PERBANDINGAN LEVENSHTEIN, SMITH-WATERMAN DAN NEEDLEMAN-WUNSCH DALAM TYPO CHECKING

Ali Nurcahya Astawijaya[1], Ken Kinanti Purnamasari[2]

[1,2] Teknik Informatika – Universitas Komputer Indonesia
Jalan Dipatiukur 114-116 Bandung 40132
E-mail : alinurcahya04@gmail.com[1], ken.kinanti@email.unikom.ac.id[2]

ABSTRAK

Kesalahan pengetikan dapat mengakibatkan kesalahan pengertian makna dikarenakan dapat mengubah kata baku menjadi kata tidak baku. Penelitian yang membahas tentang pendeteksian perbaikan kesalahan pengetikan sendiri telah banyak dilakukan. Pada penelitian ini dilakukan pendeteksian kesalahan pengetikan teks pada dokumen bahasa Indonesia menggunakan metode *levenshtein, smith waterman & needleman wunsch.*

Sistem pendeteksian kesalahan pengetikan dapat diterapkan dengan user menginputkan dokumen pada sistem, kemudian sistem akan memberikan rekomendasi dan kata terpilih dari kata yang dianggap salah. Tahapan nya sendiri dibagi menjadi empat tahap, yaitu preprocessing, pendeteksian kesalahan kata, pengubahan kata ke dalam kode *levenshtein, smith waterman & needleman wunsch*., proses pemberian rekomendasi dan pemilihan kata. Preprocessing adalah tahapan proses untuk mendapatkan hasil akhir yaitu list setiap kata. Kata akan digunakan pada proses pendeteksian kesalahan kata untuk menentukan kata apa saja yang dianggap salah. Selanjutnya akan dilakukan proses pengubahan kata ke dalam kode sesuai aturan metode levenshtein, smith waterman & needleman wunsch. Kode tersebut untuk menentukan hasil akhir yaitu rekomendasi dan perbaikan kata dengan mencari nilai dari rekomendasi yang dihasilkan dari kata yang dianggap salah.

Dari tiga jenis pengujian yang dilakukan, yang pertama adalah pengujian akurasi dari pendeteksian kesalahan kata. Kedua adalah akurasi rekomendasi kata. Dan ketiga adalah akurasi perbaikan kata..

Kata kunci : *Typo checking, Spelling checker, Levenshtein, Smith waterman, Needleman wunsch.*

1. PENDAHULUAN

Pada era sekarang ini penulisan dokumen banyak yang menggunakan komputer, dan pada setiap komputer tidak selalu ada program yang dapat mendeteksi kesalahan penulisan [1]. Kesalahan dalam penulisan dapat disebabkan oleh kesalahan dalam pengetikan dokumen yang tidak sesuai dengan ejaan kata yang tepat yang dapat mengubah kata menjadi tidak bermakna *(non-word*) dan dapat mengubah menjadi kata yang memiliki makna berbeda (*real-word*) [2]. Apabila dalam pengetikan dokumen terdapat tulisan yang salah, maka dapat mengubah arti dan mengubah pemahaman pembaca nya.

Pada penelitian yang berkaitan dengan *typo checking* sebagai solusi dari kesalahan penulisan kata terdapat penelitian typo checking pada bahasa india. Kesimpulan yang didapat pada penelitian tersebut setelah mengambil 2 sample yaitu, algoritma *Smith Waterman* mendapat nilai akurasi terbaik, lalu algoritma *Needleman Wunsch* mendapat nilai akurasi pertengahan dan algoritma Levenshtein kurang baik untuk memperbaiki kesalahan dalam pengetikan [3]. Sedangkan dalam penelitian typo checking dalam bahasa Indonesia, kesimpulan yang didapat adalah penggunaan *Levenshtein Distance* untuk memperbaiki kesalahan dalam pengetikan sudah tepat dan dapat memberikan hasil akurasi yang cukup tinggi yaitu 86,4% [4].

Pada penelitian ini akan menggunakan algoritma *Levenshtein, Smith-Waterman, dan Needleman-Wunsch*, karena dibutuhkannya algoritma mana yang mendapatkan solusi yang optimal selain *levenshtein* sebagai pembanding. Berdasarkan uraian di atas, maka pada penelitian ini akan dilakukan typo checking dengan membandingkan algoritma *Levenshtein, Smith-Waterman, dan Needleman-Wunsch,* untuk menentukan akurasi pada kasus pendeteksian kesalahan penulisan kata pada teks berita Berbahasa Indonesia..

2. ISI PENELITIAN

2.1 Typo Checking

Typo checking adalah proses pemeriksaan kata untuk mendeteksi kata yang salah eja dan memberikan kandidat kata yang benar. *typo checking* memiliki fitur yaitu *spelling checker.*

Desain dari proses *Spelling checker* yang dilakukan yaitu:

1. Melakukan pengolahan pra proses pada teks
2. Lalu memeriksa setiap kata apakah kata salah atau tidak.
3. Selanjutnya tahapan memperbaiki kata untuk mendapatkan kata yang benar [2][9].

Typo Checking memiliki 2 cara dalam pendeteksian kesalahan yaitu.

1. Pendeteksian kesalahan *non-word* yaitu pendeteksian kesalahan kata yang tidak memiliki arti. Contoh: **nasi** menjadi **nsi**. Sistem hanya mendeteksi kesalahan kata yang tidak memiliki arti.
2. Pendeteksian kesalahan *real-word* yaitu pendeteksian kesalahan kata yang memiliki makna/arti lain seperti **nasi** yang menjadi **basi**.

Pada penelitian ini hanya akan mendeteksi kesalahan *non-word* yaitu fokus pada kesalahan kata yang tidak memiliki arti seperti **nasi** menjadi **nsi**

2.2 Pencocokan String

Pencocokan string (*string matching*) secara garis besar dapat dibedakan menjadi dua yaitu *exact string matching* dan *in exact string matching* [3][10].

2.2.1. *Exact String Matching*

Merupakan pencocokan *string* secara tepat dengan susunan karakter dalam string yang dicocokkan memiliki jumlah maupun urutan karakter dalam string yang sama. Contoh : kata "baju" akan menunjukkan kecocokan hanya dengan kata "baju".

2.2.2. *In Exact String Matching*

Disebut juga *fuzzy string matching*, merupakan pencocokan *string* secara samar, maksudnya pencocokan string di mana *string* yang dicocokkan memiliki kemiripan di mana keduanya memiliki susunan karakter yang berbeda (mungkin jumlah atau urutannya) tetapi string-string tersebut memiliki kemiripan baik kemiripan tekstual/penulisan (*approximate string matching*) atau kemiripan ucapan *(phonetic string matching)*. *InExact string matching* masih dapat dibagi lagi menjadi dua yaitu:

1. Pencocokan *string* berdasarkan kemiripan penulisan *(approximate string matching)* merupakan pencocokan *string* dengan dasar kemiripan dari segi penulisannya (jumlah karakter, susunan karakter dalam dokumen). Tingkat kemiripan ditentukan dengan jauh tidaknya beda penulisan dua buah *string* yang dibandingkan tersebut dan nilai tingkat kemiripan ini ditentukan oleh pemrogram (*programmer*). Contoh: lari dengan lara, memiliki jumlah karakter yang sama tetapi ada dua karakter yang berbeda. Jika perbedaan dua karakter ini dapat ditoleransi sebagai sebuah kesalahan penulisan maka dua *string* tersebut dikatakan cocok.
2. Pembetulan bunyi (*phonetic correction*) adalah kesalahan pengejaan yang muncul karena kesalahan user dalam memasukan *query* yang memiliki bunyi seperti istilah target [3][10]. *Phonetic string matching* merupakan suatu teknik pencocokan *string* yang membandingkan suatu *string* dengan *string* yang lain berdasarkan kode fonetis masing-masing. String yang mempunyai kode fonetis yang sama bisa dikatakan mirip berdasarkan ucapan.

2.3 Pengertian Fonetik (*phonetics*)

Adalah ilmu yang menyelidiki bunyi bahasa tanpa melihat fungsi bunyi itu sebagai pembeda makna dalam suatu bahasa (*language*) [7]. Kata sifat fonetik adalah fonetis (*phonetic*). Bagian fonetik bahasa Inggris yang berkaitan erat dengan phonetic string matching adalah klasifikasi konsonan, karena klasifikasi konsonan yang memegang peranan penting dalam phonetic string matching. Konsonan didasarkan pada alat bicara yang menghasilkannya dapat dibedakan menjadi tujuh kelompok yaitu: [3] [8]

1. *Labial* atau bunyi bibir, yang dapat dibedakan lagi menjadi dua golongan yaitu :
   1.1. *Bilabial*, bunyi diartikulasi oleh dua bibir, contoh bunyi p, b, m, w.
   1.2. *Labio-dental*, bunyi diartikulasi oleh bibir bawah dan gigi atas, f,v.
2. *Dental*, bunyi diartikulasi oleh ujung lidah dengan gigi atas, contoh bunyi th (dalam kata *thin*)
3. *Alveolar*, bunyi diartikulasi oleh ujung lidah dengan punggung gigi (*teeth-ridge*), contoh bunyi d, t, n, l, r, s, z.
4. *Palato-alveolar*, bunyi yang memiliki artikulasi alveolar dikuti dengan naiknya lidah sampai pada langit-langit mulut secara simultan, contoh bunyi c, j, sh (dalam kata *show*).
5. *Palatal*, bunyi diartikulasi oleh bagian depan lidah dengan langit-langit keras (*hard palate*), contoh bunyi y.
6. *Velar*, bunyi diartikulasi oleh bagian belakang lidah dengan langit-langit lunak (*soft palate*), contoh bunyi k, g.
7. *Glottal*, bunyi diartikulasi oleh *glottis*, contoh bunyi h.

Berdasarkan cara artikulasi (dihambat), konsonan dibedakan menjadi delapan golongan yaitu : [3][8]

1. Konsonan hambat letup (*plosive*), merupakan konsonan yang terjadi dengan hambatan penuh arus udara kemudian hambatan itu dilepaskan secara tiba-tiba. Contoh : b, d, g, k, p, t.
2. Konsonan nasal atau sengau (*nasal*), merupakan konsonan yang dibentuk dengan menghambat rapat (menutup) jalan udara dari paru-paru melalui rongga mulut, bersama dengan itu langit-langit lunak beserta anak tekaknya diturunkan sehingga udara keluar melalui rongga hidung. Contoh : m, n.
3. Konsonan paduan (*affricate*), merupakan konsonan hambat jenis khusus dimana proses terjadinya dengan menghambat penuh arus udara dari paru-paru, kemudian hambatan itu dilepaskan

secara pelan-pelan. Tempat artikulasinya pada palato-alveolar. Contoh : c, j.

4. Konsonan sampingan (*lateral*), merupakan konsonan yang dibentuk dengan menutup arus udara di tengah rongga mulut sehingga udara keluar melalui kedua samping atau sebuah samping saja. Tempat artikulasinya pada alveolar. Contoh : l.
5. Konsonan geseran (*fricative*), merupakan konsonan yang dibentuk dengan menyempitkan jalannya arus udara yang dihembuskan dari paru-paru, sehingga jalannya udara terhalang dan keluar dengan bergeser. Contoh : f, v, r, s, z, th (dalam kata *thin*), sh (dalam kata *show*), h.
6. Konsonan getar (*rolled*), merupakan konsonan yang dibentuk dengan menghambat jalannya arus udara yang dihembuskan dari paru-paru secara berulang-ulang dan cepat dan terjadi banyak sentuhan (*tap*) yang terjadi antara ujung lidah dengan langit-langit atau gusi belakang. Contoh *rolled* r (sangat jarang ditemukan).
7. Konsonan sentuhan kuat (*flapped*), merupakan konsonan dengan proses yang hampir sama dengan konsonan rolled tetapi hanya terjadi satu sentuhan (*tap*) antara ujung lidah dengan langitlangit atau gusi belakang. Contoh flapped r (sangat jarang ditemukan). 8. Semi vokal (*semi-vowels*), bunyi yang secara praktis termasuk konsonan tetapi karena pada waktu diartikulasikan belum membentuk konsonan murni, maka bunyi-bunyi itu disebut semi-vokal. Contoh : w, y.

2.4 Analisis Masalah

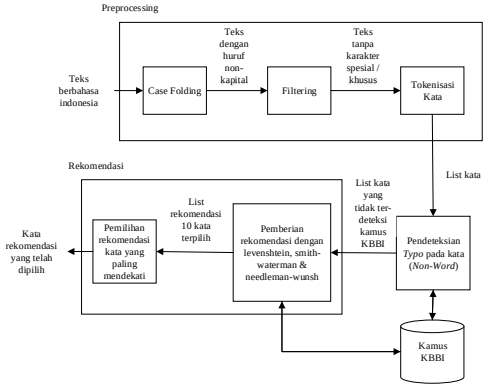
Berdasarkan pada identifikasi masalah pada penelitian ini yaitu belum adanya perbandingan pada kasus *typo checking* pada teks Berbahasa Indonesia yang menggunakan metode *levensthein, smith waterman dan needleman wunsch* untuk menghasilkan rekomendasi perbaikan kata. Dari permasalahan tersebut, maka perlu diterapkan algoritma yang dipilih untuk mendapatkan hasil yang paling baik, sehingga dapat mengetahui tepat atau tidaknya algoritma tersebut dalam typo checking kasus bahasa Indonesia. Maka sebagai solusi untuk hal tersebut, metode *levenshtein, smith waterman dan needleman wunsch* akan diterapkan pada sistem. Algoritma *levensthein, smith waterman dan needleman wunsch* untuk membantu dalam untuk merekomendasikan kata yang salah ketik dan untuk membantu dalam pemilihan perbaikan kata. Sehingga akan didapatkan hasil berupa seberapa besarkah akurasi dan performansi dari algoritma ini dalam menangani masalah yang telah dijelaskan.

Dalam typo checking ada dua jenis kesalahan yaitu yang pertama non-word adalah kesalahan yang berfokus pada kata yang terbentuk umumnya oleh kesalahan pengetikan. Kesalahan ejaan non kata ini menghasilkan kata- kata yang tidak baku. Contoh kata nasi yang menjadi nsi, nas, nai. Lalu yang kedua yaitu *real-word*, kesalahan kata yang sebenarnya menghasilkan kata sah lainnya. Contoh nasi yang menjadi basi, napi. basi dan napi sendiri memiliki makna yang berbeda dengan nasi. Pada penelitian ini penulis akan meneliti *typo checking* pada kesalahan *non-word* karena pada penelitian *typo checking* dalam bahasa Indonesia dengan kasus non-word, kesimpulan yang didapat adalah penggunaan *levenshtein distance* untuk memperbaiki kesalahan dalam pengetikan sudah tepat dan dapat memberikan hasil akurasi yang cukup tinggi yaitu 86,4% [4]. Sedangkan Pada penelitian yang berkaitan dengan *typo checking* sebagai solusi dari kesalahan penulisan kata terdapat penelitian *typo checking* pada bahasa india. Kesimpulan yang didapat pada penelitian tersebut setelah mengambil 2 sample yaitu, algoritma Smith Waterman mendapat nilai akurasi terbaik, lalu algoritma Needleman Wunsch mendapat nilai akurasi pertengahan dan algoritma Levenshtein kurang baik untuk memperbaiki kesalahan dalam pengetikan [3]. Maka pada penelitian ini akan dilakukan typo checking dengan membandingkan algoritma *Levenshtein, Smith-Waterman, dan Needleman-Wunsch*, untuk menentukan akurasi pada kasus pendeteksian kesalahan penulisan kata pada teks berita Berbahasa Indonesia.

2.5 Analisis Sistem

Sistem typo checking yang akan dibangun dengan gambaran sistem yang tertera pada Gambar. Tahap sistem typo checking dimulai dari penerimaan input teks dokumen hingga melakukan preprocessing yang terdiri dari case folding, filtering, tokenisasi kata. Hasil dari preprocessing berupa suatu kata lalu akan diproses ke proses selanjutnya dengan mencari kesalahan ketik, kesalahan ketik berupa kata yang tidak terdeteksi dengan kamus yang terdapat di KBBI. Kata yang tidak sesuai kamus akan dicari sesuai dengan aturan string matching dari levensthein, smith waterman dan needleman wunsch, kata yang memiliki string matching yang sama akan menjadi rekomendasi perbaikan kata dan akan dilakukakan pemilihan dari kata yang dianggap salah dari rekomendasi yang akan diberikan.

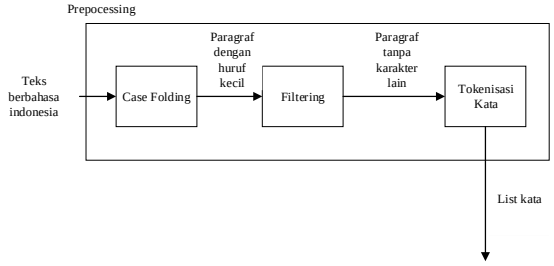
Berdasarkan analisis sistem typo checking yang akan dibangun, maka gambaran sistem dapat dilihat pada Gambar 1.



Gambar 1 Skema Proses Utama Sistem

2.5.1 Analisis Preprocessing

Tahap preprocessing merupakan tahapan untuk mempersiapkan data masukan yang akan diolah pada tahap selanjutnya. Preprocessing pada penelitian ini terdiri dari beberapa tahapan, yaitu: case folding, filtering dan tokenisasi kata.



Gambar 2 Skema Preprocessing

2.5.2 Penentuan Kesalahan Kata

Penentuan kata salah ketik yaitu dengan cara membandingkan setiap kata dengan list kata yang terdapat didalam kamus KBBI, disini kata pada jenis kesalahan ejaan non-word yang tidak terdeteksi kamus KBBI dianggap kata yang salah ketik.

2.5.3 Algoritma *Levenshtein*

Metode levenshtein mendeteksi kata yaitu dengan cara mencari nilai edit distance terkecil, edit distance didapatkan melalui perbandingan kata yang dianggap salah dengan kata rekomendasi. Contoh kata "komplek" akan dibandingkan dengan rekomendasi yang muncul yaitu "datank" dan "datang". Langkah – langkahnya dapat dilihat pada tabel 2.

Tabel 1 *Levenshtein*

Langkah	Penjelasan
1	Inisialisasi string pertama menjadi *source* (s) dengan panjang m karakter. Inisialisasi string kedua menjadi *target* (t) dengan panjang n karakter. Buat matriks yang mengandung m+1 baris dan n+1 kolom. Matriks dimulai dari baris dan kolom ke-0.
2	Isi baris ke-0 dengan 0..m. Isi kolom ke-0 dengan 0..n.
3	Periksa setiap karakter s (i dari 1 ke n).
4	Periksa setiap karakter t (j dari 1 ke m).
5	Jika s[i] = t[j], *cost* = 0. Jika s[i] ≠ t[j], *cost* = 1.
6	Isi sel matriks d[i,j] dengan nilai minimum dari sel di atasnya + 1, sel di kirinya + 1, dan sel di kiri atasnya + *cost*, dengan rumus: $d[i,j] = min\begin{cases} d[i-1,j]+1 \\ d[i,j-1]+1 \\ d[i-1,j-1]+cost \end{cases}$
7	Iterasi langkah 3, 4, 5, 6. Nilai *Levenshtein distance* ditemukan pada sel terakhir d[n,m].

Berikut adalah contoh penghitungan Levenshtein distance untuk token kata "datank". Kata "datang" akan dibandingkan dengan kata yang terdapat pada daftar kata kamus. Sebagai contoh, berikut adalah perbandingan kata "datank" dengan kata "datang" yang diambil dari daftar kata kamus.

		d	a	t	a	n	k
	0	1	2	3	4	5	6
d	1						
a	2						
t	3						
a	4						
n	5						
g	6						

Selanjutnya bandingkan setiap karakter pada matriks. Dimulai dari sel d[1,1], bandingkan karakter "d" dengan karakter "d". Pertama tentukan nilai cost. Karena "d" = "d", maka cost = 0. Lakukan untuk setiap sel matriks, hingga didapat hasil penghitungan sebagai berikut.

		d	a	t	a	n	k
	0	1	2	3	4	5	6
d	1	**0**	1	2	3	4	5
a	2	1	**0**	1	2	3	4
t	3	2	1	**0**	1	2	3
a	4	3	2	1	**0**	1	2
n	5	4	3	2	1	**0**	1
g	6	5	4	3	2	1	**1**

Hasil penghitungan dapat dilihat pada kolom terakhir baris terakhir matriks. Dari hasil penghitungan, didapat bahwa nilai Levenshtein distance dari perbandingan kata "datank" dan "datang" adalah 1.

2.5.4 *Smith-Waterman*

Algoritma Smith-Waterman dalam sistem pendeteksi kesamaan dokumen. Algoritma ini sudah dikenal luas dalam bidang bioteknologi untuk pendeteksian kesamaan DNA. Algoritma ini akan membandingkan

dua dokumen dalam sebuah matriks. Dari matriks ini, nantinya akan dilakukan traceback untuk mencari letak kesamaan dari dokumen aslinya, Langkahnya dapat dilihat pada tabel 2.

Tabel 2 Pembuatan Tabel Matriks

Langkah	Penjelasan
1	Inisialisasi i dan j dari string A dan B
2	Buat matriks S_{ij}
3	Inisialisasi nilai 0 pada metrik (0,0),(0,j),(i,0)
4	lalu, lakukan perhitungan pada metrik H(1,1) dengan rumus $S_{ij} =$ $S_{i-1,j-1} + h\ jika\ String\ A = B$ $maks$ $\{0, S_{i-1,j} - d, S_{i,j-1} - d, S_{i-1,j-1} - r\}$ $jika\ A \neq B$ i = panjang string A+1 j= panjang string B+1 H= matriks dari string A dan B h=d=r=1
5	Hitung semua metrik H_{ij} dari kiri ke kanan, atas ke bawah, bila perhitungan mendapatkan nilai negatife maka dibulatkan menjadi 0. Jika nilai positif nilai tertinggi digunakan
6	Melakukan traceback dimulai dari skor tertinggi, lalu lakukan traceback bedasarkan sumber masing-masing skor secara rekursif sampai 0 di temui.
7	Pembobotan local *similarities*

Berikut adalah contoh perhitungan smith-waterman untuk token "datank". Kata "datang" akan dibandingkan dengan kata yang berada pada kamus berdasarkan aturan-aturan algoritma smith-waterman. Lakukan untuk setiap sel matriks, hingga didapatkan perhitungan sebagai berikut.

		-	d	a	t	a	n	k
	0	1	2	3	4	5	6	7
-	1	0	0	0	0	0	0	0
d	2	0						
a	3	0						
t	4	0						
a	5	0						
n	6	0						
g	7	0						

Dimulai dari sel $H_{1,1}$, bandingkan karakter "d" dengan karakter "d". Karena "d" = "d", Selanjutnya untuk H_2,1, bandingkan karakter "a" dengan "d" karena "a" ≠ "d" maka gunakan rumus dimana nilai h=d=r=1 [11]. Lalu Hitung nilai maksimum.

	-	d	a	t	a	n	k
0	1	2	3	4	5	6	7

-	1	0	0	0	0	0	0	0
d	2	0	1	0	0	0	0	0
a	3	0	0	1	0	1	0	0
t	4	0	0	0	1	0	0	0
a	5	0	0	1	0	1	0	0
n	6	0	0	0	0	0	1	0
g	7	0	0	0	0	0	0	0

2.5.5 ***Needleman-Wunsch***

Needleman-Wunsch merupakan metode pertama yang menerapkan Dynamic Programming dalam Sequence Alignment sedangkan Smith-Waterman memodifikasi metode dari Needleman-Wunsch untuk mencari Local Alignment. Dynamic Programming sangat optimal untuk mencari kecocokan antara 2 sequences dan dapat diperluas untuk lebih dari 2 sequences. Metode ini sangat lambat untuk sequences yang sangat panjang dan perhitungannya dapat dilihat pada tabel 3.

Tabel 3 Pembuatan Tabel Matriks

Langkah	Penjelasan
1	Misalkan *sequence* A memiliki panjang M dan *sequence* B memiliki panjang N, maka buat m atriks nilai berukuran (M+1) × (N+1)
2	Kemudian isi baris pertama dan kolom pertama matriks dengan nilai dari *gap penalty*
3	Misalkan matriks nilai disebut matriks D, maka rumus untuk elemen dari matriks D, *D(i,j)* adalah: $D[i,j]$ $= max \begin{cases} D(i-1,j-1) + s(x_i, y_i) \\ D(i-1,j) + g_x \\ D(i,j-1) + g_y \end{cases}$
4	$s(x_i, y_i)$: elemen matriks substitusi di residu *i* pada *sequence* x dan residu j di *sequence* y
5	$D(i-1,j-1)$: elemen matriks D di diagonal kiri atas. $D(i,j-1)$: elemen matriks D di kiri $D(i,j)$ $D(i-1,j)$: elemen matriks D di atas $D(i,j)$
6	g_x : *gap penalty* untuk *sequence* x g_y : *gap penalty* untuk *sequence* y
7	Setelah matriks nilai berukuran (M+1) × (N+1) telah terisi

	sepenuhnya, maka alignment score maksimum dari dua buah sequence adalah nilai dari elemen paling kanan bawah dari matriks nilai, $D(M+1, N+1)$

Berikut adalah contoh penghitungan Needleman Unsch untuk token kata "hinga". Kata "hinga" akan dibandingkan dengan kata yang terdapat pada daftar kata kamus. Sebagai contoh, berikut adalah perbandingan kata "satang" dengan kata "datang" yang diambil dari daftar kata kamus.

		d	a	t	a	n	k
	0	-2	-4	-6	-8	-10	-12
d	-2						
a	-4						
t	-6						
a	-8						
n	-10						
g	-12						

Panjang sequence A: 6, Panjang sequence B : 6. Buat matriks nilai berukuran 6 × 6, indeks dimulai dari 0, isi dengan gap penalties. Lakukan untuk setiap sel matriks, hingga didapat hasil penghitungan sebagai berikut.

		d	a	t	a	n	k
	0	-2	-4	-6	-8	-10	-12
d	-2	5	3	1	-1	-3	-5
a	-4	3	10	8	6	4	2
t	-6	1	8	15	13	11	9
a	-8	-1	6	13	20	18	16
n	-10	-3	4	11	18	25	23
g	-12	-5	2	9	16	23	26

Hasil penghitungan dapat dilihat pada kolom terakhir baris terakhir matriks. Dari hasil penghitungan, didapat bahwa nilai Needleman-wunsch dari perbandingan kata "datank" dan "datang lalu di traceback"

2.6 Pengujian Sistem

Tahap pengujian sistem bertujuan untuk menemukan kesalahan-kesalahan atau kekurangan-kekurangan pada sistem yang diuji. Pengujian bermaksud untuk mengetahui apakah prototype yang dibuat sudah sesuai dengan tujuan penelitian.

2.6.1 Pengujian *Black Box*

Tabel 4 Pengujian *Black Box*

No	Nama Proses	Point Pengujian	Jenis Pengujian
1	Memilih input	Memilih data masukan	*Black box*
2	*Case Folding*	Menekan menu "*case folding"* yaitu proses pengubahan karakter menjadi huruf kecil	*Black box*
3	*Filtering*	Menekan menu "*filtering*" yaitu proses menghapus karakter yang dianggap tidak valid	*Black box*
4	Tokenisasi Kata	Menekasn menu "tokenisasi" yaitu proses menampilkan list kata	*Black box*
5	*Typo Checking*	Menekan menu "Kata Terpilih" yaitu proses menentukan kata yang dianggap salah ketik	*Black box*
6	Rekomendasi Kata	Proses pemberian rekomendasi kata yang dianggap salah dan pemilihan kata dari rekomendasi yang tersedia.	*Black box*

2.7 Pengujian Akurasi

2.7.1 Pengujian Akurasi *Levenshtein*

Pengujian ini bertujuan untuk menentukan akurasi yang dilakukan dalam penelitian ini. Dalam penelitian ini menggunakan pengujian pendeteksian kesalahan kata, pengujian akurasi perbaikan dan rekomendasi.

Tujuan dari pengujian pendeteksian kesalahan kata yaitu untuk melihat berapa akurasi pendeteksian kesalahan kata. pengukuran akurasi pendeteksian kesalahan kata yang biasa digunakan dalam penelitian pengecekan kesalahan pengetikan yaitu [12]:

$$Akurasi\ Pendeteksian\ Kesalahan\ Kata = \left(\frac{Jumlah\ Kata\ Sesuai}{Jumlah\ Kata\ Terdeteksi\ Salah}\right) x\ 100\%$$

dimana,

Jumlah kata salah sesuai : yaitu jumlah kata yang dianggap salah

Jumlah kata terdeteksi salah : yaitu jumlah total kata yang terdeteksi salah.

Tabel 5 Akurasi *Levenshtein*

No Dokumen	Jumlah Kata Sesuai	Jumlah Kata Terdeteksi Salah	Akurasi
1	6	61	9.8360
2	5	45	11.1111
3	5	81	6.1728
4	8	59	13.5593
5	5	62	8.0645
6	4	57	7.0175
7	5	23	21.7391
8	4	44	9.0909
9	8	56	14.2857
10	8	49	16.3265
11	5	32	15.625
12	5	42	11.9047
13	9	61	9.8360
14	5	42	11.9047
15	7	97	7.2164
16	9	54	9.2592
17	5	110	4.5454
18	5	47	10.6382
19	5	85	5.8823
20	6	57	10.5263
21	5	123	4.0650
22	4	42	9.5238
23	5	110	4.5454
24	5	97	5.1546
25	4	107	3.7383
26	4	45	11.1111
27	5	38	13.1578
28	5	44	11.3636
29	5	89	5.6179
30	5	71	7.0422
Rata-rata	5.5333	64.3333	9.6620

Dari Tabel 6 analisis hasil pengujian pendeteksian kesalahan kata menggunakan kamus KBBI edisi ke-4 (empat) menghasilkan akurasi sebesar 9.6620%. Berikut ini adalah penjelasan dari penyebab akurasi rekomendasi *typo checking* yang dihasilkan sistem kecil yaitu.

1. List kata yang kurang lengkap pada kamus KBBI sehingga banyak kata yang tidak terdeteksi.
2. Nama dan tempat dianggap kesalahan kata dikarenakan tidak terdapat dalam kamus KBBI.

2.7.2 Pengujian *Smith-Waterman*

Tujuan dari pengujian ini untuk melihat berapa banyak kata yang rekomendasinya tepat. Berdasarkan skenario yang dibuat Pengujian akurasi rekomendasi pada penelitian ini menggunakan pengujian akurasi. Pengukuran akurasi perbaikan yang biasa digunakan dalam penelitian pengecekan kesalahan pengetikan yaitu [12]:

$$Akurasi\ Rekomendasi = \left(\frac{Jumlah\ Kata\ Sesuai}{Jumlah\ Kata\ Terdeteksi\ Salah}\right) x\ 100\ \%$$

dimana,

Jumlah rekomendasi sesuai : yaitu jumlah total kata yang memiliki rekomendasi yang dianggap sesuai.

Jumlah kata terdeteksi salah : yaitu jumlah total kata yang terdeteksi salah.

Tabel 6 Akurasi *Smith-Waterman*

No Dokumen	Jumlah Kata Sesuai	Jumlah Kata Terdeteksi Salah	Akurasi
1	13	61	21.314
2	6	45	13.3333
3	6	81	7.4074
4	6	59	10.1694
5	13	62	20.9677
6	4	57	7.0175
7	7	23	30.4348
8	7	44	15.9090
9	6	56	10.7142
10	13	49	20.4082
11	5	32	15.625
12	6	42	14.2857
13	10	61	16.3934
14	11	42	26.1905
15	18	97	18.5567
16	10	54	18.5185
17	4	110	3.6363
18	2	47	4.2553
19	6	85	7.0588
20	13	57	22.8070
21	13	123	10.5691
22	9	42	21.4286
23	6	110	5.4545
24	9	97	9.2784
25	8	107	7.4766
26	7	45	15.5556
27	6	38	15.7895
28	5	44	11.3636
29	17	89	19.1011
30	6	71	8.4507
Rata-rata	8.4	64.3333	14.3156

Dari Tabel 7 analisis hasil pengujian rekomendasi kata menggunakan *Smith-Waterman* menghasilkan akurasi sebesar 14.3156%. Berikut ini adalah penjelasan dari penyebab akurasi rekomendasi *typo checking* yang dihasilkan sistem kecil yaitu.

1. List kata yang kurang lengkap pada kamus KBBI sehingga banyak kata tidak memiliki rekomendasi yang benar.
2. Kesalahan kata mempengaruhi pengubahan sehingga yang dihasilkan tidak benar pada rekomendasi yang seharusnya tidak muncul.

2.7.3. Pengujian *Needleman-Wunsch*

Tujuan dari pengujian ini untuk melihat berapa banyak kata yang memiliki perbaikan tepat

pengukuran akurasi perbaikan yang biasa digunakan dalam penelitian pengecekan kesalahan pengetikan yaitu [12]:

$$Akurasi\ Perbaikan = \left(\frac{Jumlah\ Kata\ Sesuai}{Jumlah\ Kata\ Terdeteksi\ Salah} \right) x\ 100\%$$

dimana,

Jumlah perbaikan benar : yaitu jumlah total kata yang memiliki perbaikan yang dianggap sesuai.

Jumlah kata terdeteksi salah : yaitu jumlah total kata yang terdeteksi salah.

Tabel 7 Akurasi *Needleman-Wunsch*

No Dokumen	Jumlah Perbaikan Kata Benar	Jumlah Kata Terdeteksi Salah	Akurasi
1	13	61	21.314
2	5	45	11.1111
3	5	81	6.1728
4	6	59	10.1694
5	10	62	16.129
6	2	57	3.5088
7	6	23	26.087
8	5	44	11.3636
9	4	56	7.1428
10	13	49	18.367
11	4	32	12.5
12	5	42	11.905
13	3	61	4.918
14	7	42	16.667
15	8	97	8.2474
16	4	54	7.4074
17	1	110	0.9090
18	1	47	2.1277
19	3	85	3.5294
20	9	57	15.7894
21	9	123	7.3171
22	7	42	16.6667
23	3	110	2.7273
24	6	97	6.1856
25	1	107	0.9346
26	4	45	8.8889
27	5	38	13.1579
28	4	44	9.0909
29	5	89	5.618
30	4	71	5.6338
Rata-rata	5.4	64.3333	9.7195

Dari Tabel 8 analisis hasil pengujian penrbaikan kata menggunakan metode *needleman-wunsch* menghasilkan rata-rata akurasi sebesar 9.7195%. Berikut ini adalah penjelasan dari penyebab akurasi dari perbaikan kata *typo checking* yang dihasilkan sistem kecil.

2.8 Analisis Hasil Pengujian

Pengujian sistem yang telah dilakukan menunjukkan bahwa sistem yang yang dibuat dari proses preprocessing, pendeteksian kesalahan kata dengan metode levensthein, smith-waterman. needleman, sudah memenuhi persyaratan fungsional.

Tabel 9 Hasil Analisis Pengujian

No Dokumen	*Levenshtein*	*Smith-Waterman*	*Needleman-Wunsch*
1	9.8360%	21.314%	21.314%
2	11.1111%	13.3333%	11.1111%
3	6.1728%	7.4074%	6.1728%
4	13.5593%	10.1694%	10.1694%
5	8.0645%	20.9677%	16.129%
6	7.0175%	7.0175%	3.5088%
7	21.7391%	30.4348%	26.087%
8	9.0909%	15.9090%	11.3636%
9	14.2857%	10.7142%	7.1428%
10	16.3265%	20.4082%	18.367%
11	15.625%	15.625%	12.5%
12	11.9047%	14.2857%	11.905%
13	9.8360%	16.3934%	4.918%
14	11.9047%	26.1905%	16.667%
15	7.2164%	18.5567%	8.2474%
16	9.2592%	18.5185%	7.4074%
17	4.5454%	3.6363%	0.9090%
18	10.6382%	4.2553%	2.1277%
19	5.8823%	7.0588%	3.5294%
20	10.5263%	22.8070%	15.7894%
21	4.0650%	10.5691%	7.3171%
22	9.5238%	21.4286%	16.6667%
23	4.5454%	5.4545%	2.7273%
24	5.1546%	9.2784%	6.1856%
25	3.7383%	7.4766%	0.9346%
26	11.1111%	15.5556%	8.8889%
27	13.1578%	15.7895%	13.1579%
28	11.3636%	11.3636%	9.0909%
29	5.6179%	19.1011%	5.618%
30	7.0422%	8.4507%	5.6338%
Rata-rata	9.6620%	14.3156%	9.7195%

3. PENUTUP

3.1 Kesimpulan

Berdasarkan hasil dari 3 (tiga) pengujian yang telah dilakukan, dapat diketahui bahwa typo checking memiliki akurasi yang sedikit kecil dari hasil 30 uji coba dokumen berita dengan rata-rata akurasi pendeteksian kesalahan kata pada *levensthein* sebesar 9.66%, rata-rata akurasi pendeteksian kesalahan kata pada *smith waterman* sebesar 14.31% dan rata-rata akurasi pendeteksian kesalahan kata *pada needleman wunsch sebesar* 9.71% di setiap dokumen dengan kesalahan kata yang dibuat manual rata-rata disetiap dokumen sebanyak 2 kesalahan kata, kesalahan kata yang terdeteksi belum tentu murni kesalahan ketik dikarenakan sistem yang dibangun belum dapat mendeteksi nama dan tempat. Dikarenakan kesalahan kata ditentukan oleh kata yang tidak terdapat dalam kamus data sehingga kamus data sangat bergantung dalam menentukan kesalahan kata dan rekomendasi kata.

3.2 Saran

Pada penelitian ini dengan menggunakan kamus KBBI sebagai sumber daya dalam menentukan kata yang salah ketik, memiliki akurasi pendeteksian kesalahan kata yang kurang baik dikarenakan tidak dapat mendeteksi nama dan tempat. Sehingga penelitian selanjutnya disarankan untuk mengganti sumber daya kamus dengan kamus yang lebih banyak dan lebih lengkap dalam inventori kata, atau penambahan untuk pendeteksian nama dan tempat seperti *NER (named entity recognition)* sehingga dapat mendeteksi nama dan tempat untuk meningkatkan pendeteksian kesalahan kata, akurasi pemberian rekomendasi dan akurasi perbaikan kata yang lebih baik.

DAFTAR PUSTAKA

[1] T. A. Hariawan, Implementasi Spell Checker Dengan Menggunakan Algoritma Metaphone Pada Kata Berbahasa Indonesia, Daerah Istimewa Yogyakarta: Sinta, 2005.

[2] M. Fachrurrozi and A. A. Manik, Perbaikan Ejaan Kata pada Dokumen Bahasa Indonesia dengan Metode Cosine Similarity, Palembang: Teknik Informatika Universitas Sriwijaya, 2015.

[3] M. Syaroni and R. Munir, Pencocokan String Berdasarkan Kemiripan Ucapan (Phonetic String Matching) dalam Bahasa Inggris, Bandung: Departemen Teknik Informatika ITB, 2004.

[4] N. UzZaman and M. Khan, A Encoding for Bangla and its Application in Spelling Checker, Bangladesh: Center for Research on Bangla Language Processing BRAC University, 2005.

[5] H. Cahyadi, Pencarian Nama Ilmiah pada Koleksi Tesis Perpustakaan IPB, Bogor: Departemen Ilmu Komputer IPB, 2010.

[6] S. Rahadian, R. W. Ciptasari and Adiwijaya, Perbandingan Pencocokan String Nama dalam Bahasa Indonesia Berdasarkan Kemiripan Ucapan (Phonetic String Matching) Needleman dan Smith-Waterman, Bandung: Fakultas Teknik Informatika Universitas Telkom, 2007.

[7] Anonim, "Fonetik," 11 Mei 2017. [Online]. Available: https://kbbi.web.id/fonetik. [Accessed 15 Mei 2017].

[8] D. Jones, The Pronouncation of English 4th edition, Cambridge: Great Britain at the University Press, 1972.

[9] M. Y. Soleh and A. Purwanti, A Non Word Error Spell Checker for Indonesian using Morphological Analyzer and HMM, Bandung: Department of Informatics ITB, 2011.

[10] Binstock and Rex, Practical Algorithms for Programmers, Addison Wesley, 1995.

[11] A. Pahdi, Koreksi Ejaan Istilah Komputer Berbasis Kombinasi Algoritma Damerau-Levenshtein dan Algortima Soundex, Banjarbaru: Journal Speed, 2016.

[12] M. F. Wahyudipraja, Implementasi Algoritma Cocke -Younger-Kasami (CYK) Dan Levenshtein Untuk Merekomendasikan Perbaikan Struktur Kalimat Dan Kesalahan Pengetikkan Bahasa Indonesia, Bandung: JBPTUNIKOMPP, 2015.