PENDETEKSIAN KESALAHAN PENGETIKAN MENGGUNAKAN METODE *DOUBLE METAPHONE* PADA TEKS BERBAHASA INDONESIA

Norfa Yoga Pratama S.[1], Ken Kinanti Purnamasari[2]

[1,2] Teknik Informatika – Universitas Komputer Indonesia
Jalan Dipatiukur 114-116 Bandung 40132
E-mail : dnorfa@yahoo.com[1], guinev_gs@yahoo.co.nz[2]

ABSTRAK

Kesalahan pengetikan dapat mengakibatkan kesalahan pengertian makna dikarenakan dapat mengubah kata baku menjadi kata tidak baku. Penelitian yang membahas tentang pengecekan perbaikan kesalahan pengetikan sendiri telah banyak dilakukan. Pada penelitian ini dilakukan pengecekan kesalahan pengetikan teks pada dokumen bahasa Indonesia berbasis kemiripan ucapan kata *(phonetic string matching)* menggunakan metode *double metaphone*.

Sistem pengecekan kesalahan pengetikan dapat diterapkan dengan *user* menginputkan dokumen pada sistem, kemudian sistem akan memberikan rekomendasi dan kata terpilih dari kata yang dianggap salah. Tahapan yang dilakukan sistem untuk mendapatkan rekomendasi dan kata perbaikan sendiri dibagi menjadi empat tahap utama, yaitu *preprocessing*, pendeteksian kesalahan kata, pengubahan kata ke dalam kode *double metaphone*, proses pemberian rekomendasi dan pemilihan kata. *Preprocessing* adalah tahapan dimana dokumen akan diproses untuk mendapatkan hasil akhir yaitu list setiap kata. Kata akan digunakan pada proses pendeteksian kesalahan kata untuk menentukan kata apa saja yang dianggap salah. Setelah mendapatkan kata yang dianggap salah, akan dilakukan proses pengubahan kata ke dalam kode sesuai aturan metode *double metaphone*. Kode tersebut untuk menentukan hasil akhir yaitu rekomendasi dan perbaikan kata dengan mencari nilai *edit distance* terkecil menggunakan teknik *approximate string matching* dengan metode *levenshtein* dari rekomendasi yang dihasilkan dari kata yang dianggap salah.

Dari tiga jenis pengujian, yang pertama adalah pengujian akurasi dari pendeteksian kesalahan kata. Kedua adalah akurasi rekomendasi kata. Dan ketiga adalah akurasi perbaikan kata dengan masing-masing 30 data uji coba. didapatkan hasil akurasi pendeteksian kesalahan kata sebesar 9.6620%, akurasi rekomendasi kata sebesar 14.3156% dan akurasi perbaikan kata sebesar 9.7195%. Akurasi yang didapatkan kecil disebabkan oleh sistem yang tidak dapat mendeteksi nama dan tempat.

Kata kunci : *Typo checking, Spelling checker, Phonetic string matching, Double metaphone.*

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.

Terdapat penelitian yang telah melakukan pengecekan kesalahan pengetikan diantaranya menggunakan morphologically analyzer, cosine similarity, approximate string matching dan phonetic string matching. Phonetic string matching sendiri memiliki banyak metode seperti soundex, metaphone, caverphone [3] dan salah satunya yaitu dengan double metaphone yang merupakan pengembangan dari metaphone [4][5] yang mencocokkan string berdasarkan kemiripan ucapan. Sebuah string dikatakan mirip jika mempunyai kode fonetis yang sama. Metode double metaphone dapat mengatasi perbedaan pengucapan dan penulisan kata pada bahasa Inggris dengan cara mentransformasikan suatu kata ke dalam kode berdasarkan bunyinya sehingga dapat digunakan dalam pemeriksaan ejaan.

Oleh karena itu, pada penelitian ini menggunakan phonetic string matching dengan metode double metaphone karena pada penelitian sebelumnya mendapatkan akurasi yang cukup tinggi. Berdasarkan uraian di atas, maka pada penelitian ini akan dilakukan typo checking menggunakan metode double metaphone untuk menentukan akurasi pada kasus pendeteksian kesalahan pengetikan kata pada teks 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 Analis Masalah

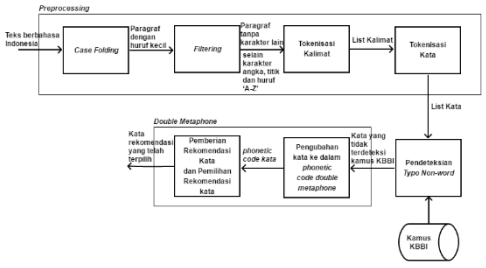
Berdasarkan pada identifikasi masalah pada penelitian ini terdapat permasalahan belum adanya akurasi pada kasus *typo checking* pada teks berbahasa Indonesia yang menggunakan metode *double metaphone* untuk menghasilkan rekomendasi perbaikan kata. Dari permasalahan tersebut, maka perlu diterapkan algoritma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

Pada penelitian lain yang berkaitan dengan metode *double metaphone* telah dilakukan sebelumnya oleh "penelitian nama ilmiah pada koleksi tesis IPB menggunakan *double metaphone*" mendapatkan persentase keberhasilan cukup tinggi sebesar 84% dalam kasus kesalahan pengetikan nama ilmiah [5]. Oleh karena itu, peneliti menggunakan metode *double metaphone* karena pada penelitian sebelumnya mendapatkan akurasi cukup tinggi dalam bahasa ilmiah. Sehingga pada penelitian *typo checking* ini, menggunakan metode *double metaphone* untuk merekomendasi kata yang salah ketik pada teks berbahasa Indonesia.

2.5 Analisis Sistem

Sistem *typo checking* yang akan dibangun dengan gambaran sistem yang tertera pada Gambar 1. Tahap sistem *typo checking* dimulai dari penerimaan input teks dokumen hingga melakukan *preprocessing* yang terdiri dari *case folding, filtering,* tokenisasi kalimat, 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 *phonetic code* nya sesuai dengan aturan *phonetic code* dari *double metaphone*, kata yang memiliki *phonetic code* yang sama akan menjadi rekomendasi perbaikan kata dan akan dilakukakan pemilihan dari kata yang dianggap salah dari rekomendasi yang akan diberikan.

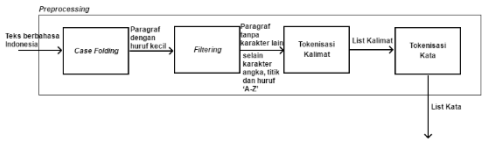
Berdasarkan analisis sistem *typo checking* yang akan dibangun, maka gambaran sistem dapat dilihat pada Gambar 1



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 dengan hasil akhir untuk mendapatkan list setiap kata. Preprocessing pada penelitian ini terdiri dari beberapa tahapan, yaitu: *case folding*, *filtering*, tokenisasi kalimat 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, kata yang tidak terdeteksi kamus KBBI dianggap kata yang salah ketik.

2.5.3 Algoritma *Double Metaphone*

Algoritma Double metaphone merupakan salah satu algoritma phonetic yang mendeteksi string nama yang mempunyai kemiripan dalam ucapan yang menghasilkan dua macam kode fonetis, yaitu panjang kode fonetis primer dan sekunder maksimal empat digit [6]. Algoritma *double metaphone* merupakan pengembangan dari algoritma *Metaphone* yang diperkenalkan oleh Lawrence Philips pada tahun 2000. Algoritma ini disebut *Double metaphone* karena algoritma ini menghasilkan dua buah kode fonetis yaitu primer dan sekunder, sedangkan pada algoritma *Metaphone* hanya dihasilkan satu buah kode fonetis. Kode fonetis sendiri maksimal terdiri dari 4 huruf. Contoh ‘MNKN’ sehingga aturan kode setelahnya diabaikan. Hal ini dilakukan untuk mengatasi masalah kesalahan pengetikan dan munculnya keanekaragaman dalam pengucapan *string* nama yang sebelumnya tidak bisa diatasi oleh algoritma *metaphone*. Dengan demikian, sebuah *string* bisa mempunyai dua kemungkinan yaitu mirip berdasarkan *phonetic code primary* atau mirip berdasarkan *phonetic code secondary.*

Aturan *phonetic code double metaphone* dapat dilihat pada tabel 2 [5].

Tabel 2 Aturan *Double Metaphone*

Huruf	Kode Fonetis		Keterangan
	Primary	*Secondary*	
A,I,U,E,O,Y	A	A	Jika diawal kata
A,I,U,E,O,Y	Dihapus	Dihapus	Diabaikan jika berada di tengah atau akhir kata (selain di awal kata)
B	B	B	Default
	P	P	Di akhir kata setelah 'M' Contoh : *dumb*
C	K	K	Jika dalam '-SCH-', ORCHES, ARCHIT, ORCHID, 'MC'-,CK,CG,CQ. Diikuti -HARAC,-HARIS,-HOR,-HYM,-HIA,-HEM,Setelah VAN, VON, MC, Sebelum 'L', 'R', 'N', 'M', 'B', 'H', 'F', 'V', 'W', default
	K	X	Jika dalam 'CHAE',
	S	S	Jika dalam '-CI-', '-CE-', 'CY', 'CAESAR','-SCI-', '-SCE-', '-SCY-'
	S	X	Jika dalam 'CZ',Diikuti '-CIA', 'CIO', 'CIE'
	X	X	Jika dalam 'CC'
	KS	KS	Jika dalam 'UCCEE', 'UCCES', dalam 'CC' diikuti 'I', 'E', 'H' (bukan 'HU'), Diawali A dan dalam 'CC',
	dihapus	dihapus	Jika dalam 'SCI, 'SCE', 'SCY'
D	J	J	Jika dalam '-DGE-', '-DGY-', 'DGI'
	T	T	Default
	TK	TK	Jika dalam 'DG'
F	F	F	Default
G	K	K	Jika dalam 'GH',default
	J	J	Jika dalam 'GH' diikuti 'I' Jika dalam '-GNED', '-GN', '-DGE-', '-DGI-"-DGY-'
	Dihapus	Dihapus	Jika dalam 'B--GH','D--GH', '-H--GH','-H---GH'
	F	F	Jika dalam '-UGH' diawali 'C', 'G', 'L', 'R', 'T'

	N	KN	Jika diikuti '-N'
	KL	L	Jika diikuti 'LI
	K	J	Diikuti 'ES', 'EP ', 'EB ', 'EL ', 'EY ', 'IB ', 'IL ', 'IN ', 'IE ', 'EI ', 'ER', di dalam –GER,-GY, -GE
H	dihapus	dihapus	Jika sesudah vokal dan tidak diikuti vokal
	H	H	Jika sebelum sebuah vokal dan tidak sesudah 'C', 'G', 'P', 'S', 'T'
J	J	J	Default
	H	H	Jika dalam 'JOSE'
K	dihapus	dihapus	Jika sesudah 'C'
	K	K	Default
	Dihapus	Dihapus	Jika diakhir kata Sebelum ‘vocal’ -LEK
L	L	L	Default
M	M	M	Default
N	N	N	Default
P	F	F	Jika sebelum 'H'
	P	P	Default
Q	K	K	Default
R	R	R	Default
S	X	X	Jika di dalam '-SIA-', '-SIO-' atau sebelum 'H'
	S	S	Default, Jika dalam 'SCI-', 'SCE-', 'SCY-'
	dihapus	dihapus	Jika di dalam '-ISL-', '-YSL-'
	X	S	Jika di dalam 'Sugar', 'SC'
	S	X	Jika diikuti 'N', 'M', 'L', 'W', 'Z'
	X	SK	Jika di dalam 'SCH-' diikuti 'ER', 'EN'
	SK	SK	Jika di dalam 'SCH-' diikuti 'OO', 'UY', 'ED', 'EM', Jika dalam 'SC-'
T	X	X	Jika dalam '-TIA-', '-TION-', 'TCH', jika sebelum 'H'
	0	0	Jika di dalam '-TCH-'
	dihapus	Dihapus	Jika diikuti 'T'
	T	T	Jika dalam TH diikuti 'OM', 'AM'
	0	T	Jika dalam 'TH','TTH'
V	F	F	Default
W	W	W	Jika sesudah vokal
	R	R	Jika di dalam 'WR'
	A	A	Jika di dalam 'WH'
	Dihapus	F	Jika di dalam 'EWSKI', 'EWSKY', 'OWSKI', 'OWSKY','SCHW-'
	TS	FX	Jika diikuti '-WICZ', '-WITZ'
X			Default
Z	S	S	Default
	J	J	Jika diikuti '-H'
	S	TS	Jika diikuti 'ZO', 'ZI', 'ZA'

Kata yang dianggap salah akan dicari kode *phonetic* nya lalu dibandingkan dengan kamus data. Kode yang mirip akan menjadi rekomendasi kata.

2.5.4 Algoritma *Levenshtein*

Algoritma ini dikenal dengan *Levenshtein Distance*. Algoritma ini ditemukan oleh Vladimir Losifovich Levenshtein yang merupakan ilmuan dari Rusia pada tahun 1965. Algoritma ini mengukur kesamaan antara 2 string untuk menentukan nilai terkecil atau disebut dengan nilai *edit distance*, 2 string yang akan dibandingkan nantinya akan lebih dikenal dengan string sumber (s) dengan string target (t) [11], contoh :

1. Jika (s) = “coba”, dan (t) = “coba” ==> maka *Levenshtein*(s,t) = 0, karena kedua string tersebut sama / tidak ada perbedaan
2. Jika (s) =”coba”, dan (t) = “cona” ==> maka *Levenstein*(s,t) =1 , karena adanya perbedaan antara kedua string tersebut di huruf yang ketiga yakni ‘b’ dengan ‘n’.

Contohnya jika ada 2 buah kata x = BARU dengan y = BATU, berapakah nilai *edit distance* perbedaan huruf dari kedua kata tersebut.

1. Pembuatan Tabel Matriks

Tabel 2 Pembuatan Tabel Matriks

d[i,j]		B	A	R	U
	0	1	2	3	4
B	1				
A	2				
T	3				
U	4				

a. Pengisian tabel matriks

ketika i = 1, dan j = 1

*x[i] = B , y[i] = B, cost = 0

kemudian ambil nilai minimal dari

- d[i-1,j] + 1 ===> d[1-1,1] + 1 = 1
- d[i,j-1] + 1 ===> d[1,1-1] + 1 = 2
- d[i-1,j-1] + cost ===> d[1-1,1-1] + 0 = 0

2. Pengisian tabel matriks

Tabel 3 Pengisian Tabel Matriks

d[i,j]		B	A	R	U
	0	1	2	3	4
B	1	**0**			
A	2				
T	3				
U	4				

3. Hasil pengisian matriks.

Tabel 4 Hasil *Edit Distance*

d[i,j]		B	A	R	U
	0	1	2	3	4
B	1	0	1	2	3
A	2	1	0	1	2
T	3	2	1	1	2
U	4	3	2	2	**1**

Dari sini dapat kita ketahui nilai *edit distance* kedua kata BARU dan BATU bernilai 1 (satu). Dari kata yang dianggap salah akan menjadi string sumber yang akan dibandingkan oleh setiap rekomendasi yang dihasilkan sistem. Rekomendasi kata yang memiliki nilai edit distance terkecil akan dipilih sebagai perbaikan kata yang dianggap salah

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 5 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 Kalimat	Menekan menu "tokenisasi kalimat" yaitu proses menampilkan daftar kalimat	*Black box*
5	Tokenisasi Kata	Menekasn menu "tokenisasi kata" yaitu proses menampilkan list kata	*Black box*
6	*Typo Checking*	Menekan menu "*typo checking*" yaitu proses menentukan kata yang dianggap salah ketik	*Black box*

7	*Double Metaphone*	Menekan menu "*double metaphone*" yaitu proses pengubahan kata ke dalam *phonetic code* sesuai aturan metode *double metaphone*	*Black box*
		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 Pendeteksian Kesalahan Kata

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]:

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

dimana,

Jumlah kata salah sesuai : yaitu jumlah kata yang dianggap salah

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

Tabel 6 Akurasi Pendeteksian Kesalahan Kata

No Dokumen	Jumlah Kata Salah 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 Akurasi Rekomendasi Kata

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]:

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

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 7 Akurasi Rekomendasi Kata

No Dokumen	Jumlah Rekomendasi 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 *double metaphone* 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 *phonetic code* sehingga menyebabkan *phonetic code* yang dihasilkan tidak sesuai *phonetic code* yang benar sehingga rekomendasi yang seharusnya tidak muncul.

2.7.3. Pengujian Akurasi Perbaikan Kata

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]:

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

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 8 Akurasi Perbaikan Kata

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 *levenshtein* menghasilkan rata-rata akurasi sebesar 9.7195%. Berikut ini adalah penjelasan dari penyebab akurasi dari perbaikan kata *typo checking* yang dihasilkan sistem kecil yaitu.

1. Tidak terdapat rekomendasi yang benar
2. KBBI kurang lengkap sehingga berpengaruh pada rekomendasi yang sedikit
3. Kata yang benar memiliki *edit distance* yang besar sehingga tidak terpilih

4. Akibat variasi kesalahan ketik *phonetic code* menjadi berbeda dan tidak menghasilkan rekomendasi yang seharusnya

2.8 Analisis Hasil Pengujian

Pengujian sistem yang telah dilakukan menunjukkan bahwa sistem yang yang dibuat untuk pendeteksian kesalahan, pemberian rekomendasi dan perbaikan kata dari proses *preprocessing*, pendeteksian kesalahan kata, pengubahan kata ke dalam *phonetic code double metaphone* dan pemberian rekomendasi dan perbaikan kata dengan metode *double metaphone*, sudah memenuhi persyaratan fungsional.

Tabel 9 Hasil Analisis Pengujian

No Dokumen	Akurasi Pendeteksian Kesalahan Kata	Akurasi Rekomendasi Kata	Akurasi Perbaikan Kata
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 penelitian yang telah dilakukan, dapat diketahui bahwa *typo checking* menggunakan metode *double metaphone* memiliki pendeteksian kesalahan kata yang kurang baik dari hasil 30 uji coba dokumen berita dengan rata-rata akurasi pendeteksian kesalahan kata sebesar 9.6620% di setiap dokumen sedangkan kesalahan kata yang dibuat manual rata-rata disetiap dokumen sebanyak 5 kesalahan kata, kesalahan kata yang terdeteksi belum tentu murni kesalahan ketik dikarenakan sistem yang dibangun tidak 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. Namun dari rata-rata dari 5 kata per dokumen yang sengaja dibuat salah. Sistem dengan baik dapat mendeteksi ,memberikan rekomendasi dan memberikan perbaikan yang benar dikarenakan kata tersebut terdapat di dalam kamus data. Namun letak kesalahan ketik huruf sendiri dapat mempengaruhi pengubahan ke dalam *phonetic code* sehingga terkadang pula kata yang murni salah ketik tidak mendapatkan rekomendasi dan perbaikan yang sesuai seperti kesalahan ketik huruf konsonan di awal kata atau di tengah kata. Jika kesalahan ketik di akhir kata kebanyakan tidak mempengaruhi secara signifikan jika kata tersebut panjang atau memiliki suku kata yang panjang dikarenakan *phonetic code* maksimal hanya terdiri dari 4 huruf sehingga jika sudah mendapatkan 4 huruf kata selanjutnya akan diabaikan

Rata-rata akurasi pendeteksian kesalahan kata sebesar 9.6620%, akurasi rekomendasi kata sebesar 14.3156% serta akurasi perbaikan kata yang kecil dengan rata-rata sebesar 9.7195%.

Hasil pendeteksian kesalahan kata bergantung pada kelengkapan kata pada kamus data yang digunakan, dan rekomendasi kata bergantung pada *phonetic code* yang sama dihasilkan dari kamus data, melalui metode *double metaphone* dan pemilihan kata tergantung pada nilai *edit distance* terkecil dari rekomendasi yang dihasilkan.

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 Double Metaphone 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 Menggunakan Algoritme Double Metaphone, 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) Soundex dan Double Metaphone, 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.