SISTEM KOREKSI KATA DAN PENGENALAN STRUKTUR KALIMAT BERBAHASA INDONESIA DENGAN PENDEKATAN KAMUS BERBASIS LEVENSHTEIN DISTANCE

Tria Aprilianto [1)] , Achmad Badawi [2)]

STMIK ASIA Malang

email: raptorapril@gmail.com , [1)] badawi93@gmail.com[2)]

Abstract: *Masalah yang sering terjadi dalam pembuatan karya tulis adalah kesalahan pada penulisan ejaan, dalam prakteknya seringkali disebabkan karena keteledoran pengguna. Kesalahan tersebut bias disebabkan oleh ketidaktahuan penulisan, posisi tombol keyboard dan pergerakan jari. Selain itu banyak yang belum bagitu mengenal tentang struktur kalimat dalam bahasa indonesia. Levenshtein distance digunakan untuk mencari jarak antara kata yang salah dengan kata-kata lain dalam kamus sehingga didapatkan kata-kata yang memiliki jarak terdekat. Algoritma menghasilkan saran kata yang nantinya akan dipilih oleh user sebagai pengganti kata yang salah. Penerapan algoritma levenshtein berupa nilai nilai jarak antara dua string yang dibandingkan, dengan menggunakan rumus similarity score diperoleh nilai dimana nilai yang paling tinggi yang akan dijadikan acuan dalam koreksi. Pengenalan struktur kalimat bahasa indonesia didasarkan pada ciri-ciri dari unsur kalimat. Berdasarkan hasil pengujian sistem koreksi kata dengan menginputkan beberapa kata dan mengecek berapa lama proses yang dilakukan untuk mengkoreksi inputan tersebut, dapat disimpulkan bahwa aplikasi tidak dapat memproses teks lebih dari 1000 kata atau 6 halaman. Pengujian kesalahan ejaan kata dilakukan dengan memasukkan kata yang salah dan hasil menunjukkan saran kata yang diprioritaskan adalah kata yang benar, pada pengujian kesalahn kata mempunyai nilai akurasi sebesar 86%. Sedangkan pengujian terhadap struktur bahasa indonesia dilakukan dengan memberikan kalimat dan output sistem pada penguji sebanyak 30 kalimat dengan nilai akurasi sebesar 76.66%.*

Kata kunci: *algoritma levenshtein distance, pengoreksi, ejaan, struktur kalimat.*

PENDAHULUAN

1. Latar Belakang

Salah satu manfaat komputer adalah sebagai alat bantu untuk membuat karya tulis. Berbagai aplikasi seperti *microsoft word*, *notepad*, maupun *open office word* sudah ditanamkan pada komputer untuk mempermudah pengguna. Meskipun demikian, bukan berarti penggunaannya tanpa masalah. Masalah yang dimaksud adalah adanya kesalahan pada penulisan ejaan, dalam prakteknya yang sering kali disebabkan karena keteledoran pengguna. Kesalahan tersebut biasanya disebabkan oleh ketidaktahuan penulisan, kesalahan yang berhubungan erat dengan posisi tombol *keyboard* dan pergerakan jari. Kesalahan-kesalahan yang umumnya terjadi adalah penggantian satu huruf, penyisipan satu huruf, penghilangan satu huruf, maupun penukaran dua huruf berdekatan.

Bahasa adalah salah satu komponen yang paling penting dalam kehidupan manusia. Dalam bentuk tulisan, bahasa menyimpan pengetahuan dari satu generasi ke generasi lain. Penyampaian ide atau pendapat dengan baik perlu didukung oleh penguasaan kosakata dan struktur kalimat karena semua yang hendak disampaikan harus dinyatakan melalui rangkaian struktur kata yang benar sesuai dengan aturan yang telah ada agar dapat dengan mudah dipahami oleh orang lain. Dalam penulisan sebuah karya tulis masih banyak yang belum mengenal tentang struktur kalimat dalam bahasa indonesia.

Levenshtein distance digunakan dengan mencari jarak antara kata yang salah dengan kata-kata lain dalam kamus sehingga didapatkan kata-kata yang memiliki jarak terdekat. Perhitungan levenshtein distance didapatkan dari matrik yang digunakan untuk menghitung jumlah perbedaan antara dua string. Algoritma menghasilkan saran kata yang nantinya akan dipilih oleh user sebagai pengganti kata yang salah. Berdasarkan latar belakang diatas, maka penulis melakukan penelitian dengan judul "Sistem Koreksi Ejaan Dan Pengenalan Struktur Kalimat Berbahasa Indonesia Dengan Pendekatan Kamus Berbasis *Levenshtein*

Distance".

2. Rumusan Masalah

Bagaimana membangun sistem koreksi kata dan pengenalan struktur kalimat berbahasa Indonesia dengan pendekatan kamus besar bahasa indonesia berbasis *levenshtein distance* ?

3. Batasan Masalah

Agar permasalahan yang diteliti terfokus dan tidak menyimpang, dalam pembahasan penelitian ini dibatasi pada ruang lingkup sebagai berikut :

a. Tahapan yang digunakan dalam *preprocesing* adalah *case folding*, *tokenizing*, *filtering*.
b. *Filtering* yang digunakan adalah eliminasi *commond words* dan *stop word*.
c. Tidak mengoreksi inputan teks bahasa asing.
d. Hanya mengoreksi ejaan dan susunan kalimat berdasarkan koleksi kata yang dibuat.
e. Koleksi dokumen yang digunakan berupa kamus kata berdasarkan kamus besar bahasa indonesia.
f. *Query* yang digunakan berupa kata, kalimat, paragraph dan dokumen bahasa indonesia.
g. *File* dokumen yang dapat dibaca adalah *file* .doc, .docx dan txt.

4. Tujuan dan Manfaat

Untuk membiasakan pengguna menggunakan kata-kata yang sesuai dengan kamus besar bahasa Indonesia (KBBI) dan sesuai dengan aturan dalam susunan kalimat.

5. Metodologi Penelitian

a) Observasi (Pengamatan)
Merupakan teknik pengumpulan data melalui pengamatan secara langsung terhadap objek yang telah diteliti. Pengamatan mengenai sistem pengoreksi ejaan bahasa indonesia.

b) Studi Pustaka
Studi pustaka dilakukan dengan cara mempelajari teori-teori literatur dan buku-buku yang berhubungan dengan objek kajian sebagai dasar dalam penelitian ini, dengan tujuan memperoleh dasar teoritis gambaran dari apa yang dilakukan.

c) Perancangan dan Implementasi
Tahap disain dilakukan untuk membuat rancangan yang siap diimplementasikan berdasarkan tahap-tahap sebelumnya. Pada tahap implementasi design harus diterjemahkan dalam bentuk yang dapat dimengerti oleh mesin dengan menggunakan bahasa pemrograman.

d) Pengujian dan Evaluasi
Berdasarkan dari hasil implementasi yang telah dilakukan pengujian terhadap beberapa kasus. Setelah dilakukan pengujian maka dilakukan proses analisa sehingga bisa memecahkan masalah yang terjadi.

LANDASAN TEORI

1. *Information Retrieval*

Dalam mendefinisikan *Information Retrieval System* (Sistem temu kembali informasi) ada beberapa pendapat dari para ahli diantaranya :

Sistem temu kembali informasi pada dasarnya adalah suatu proses untuk mengidentifikasi, kemudian memanggil (*retrieval*). Suatu dokumen dari suatu simpanan (*file*), sebagai jawaban atas permintaan informasi. **(Hasugian,2003)**

Sistem temu kembali informasi adalah suatu proses pencarian dokumen dengan istilah-istilah bahasa pencarian untuk mendefenisikan dokumen sesuai dengan subjek yang diinginkan. **(Zaenab,2002)**

Information Retrieval adalah bidang di persimpangan ilmu informasi dan ilmu komputer berkutat dengan pengindeksan dan pengambilan informasi dari sumber informasi *heterogen* dan sebagian bersar tekstual. **(Mooers,2002)**.

2. *Prepocessing*

Sebelum proses *data mining* dapat dilaksanakan, perlu dilakukan proses *cleaning* pada data yang menjadi fokus *knowledge discovery in database.* Proses *cleaning* mencakup antara lain membuang duplikasi data, memeriksa data yang inkonsisten, dan memperbaiki kesalahan pada data, seperti kesalahan cetak (tipografi). Juga dilakukan proses *enrichment*, yaitu proses memperkaya data yang sudah ada dengan data atau informasi lain yang relevan dan diperlukan untuk *knowledge discovery in database*, seperti data atau informasi eksternal.

3. *Text Mining*

a. Pengertian

Text mining merupakan proses otomatis atau sebagian proses otomatis untuk teks

yang melibatkan pembentukan teks yang lebih terstruktur dan penggalian informasi yang relevan dari teks.

(Miller,2005)

Text mining adalah proses pengetahuan intensif dimana pengguna berinteraksi dan bekerja dengan sekumpulan dokumen dengan menggunakan beberapa alat analisis berupa sekumpulan dokumen dan pola menarik yang tidak ditemukan dalam bentuk *database record*, tetapi dalam bentuk data teks yang tidak terstruktur. **(Feldman dan J. Sanger,2007)**

b. Tujuan *Text Mining*

Tujuan dari *text mining* adalah untuk mendapatkan informasi yang berguana dari sekumpulan dokumen. Jadi, sumber data yang digunakan pada *text mining* adalah kumpulan teks yang memilik format yang tidak terstruktur atau minimal semi terstruktur.

4. Levenshtein Distance

Algoritma levenshtein distance dibuat oleh Vladimir Levenshtein pada tahun 1965. Algoritma levenshtein distance merupakan matrik yang digunakan untuk mengukur perbedaan jarak antara kedua sekuens. Perhitungan edit distance didapatkan dari matrik yang digunakan untuk menghitung jumlah perbedaan antara dua string. Perhitungan jarak antara dua string ini ditentukan dari jumlah minimum operasi perubahan untuk membuat string A menjadi string B. Levenshtein distance antara dua string ditentukan berdasarkan jumlah minimum perubahan / pengeditan yang dibutuhkan untuk melakukan transformasi dari satu bentuk string ke string lain (**Trevisan, 2001**).

5. Dynamic Programing Dalam Levenshtein Distance

Dalam algoritma levenshtein distance, dilakukan dynamic programing bottom-up yang bekerja dengan melibatkan suatu matriks yang berisi angka-angka yang merupakan ongkos (cost) yang dibutuhkan untuk mengubah suatu string mejadi string lain. Algoritma ini mengembalikan suatu angka yang merupakan ongkos (cost) minimum yang diperlukan untuk mengubah string awal menjadi string target yang kita kenal bersama dengan levenshtein distance **(Adiwidya, 2009:3).**

6. Batas Atas dan Batas Bawah Levenshtein Distance

Suatu hal yang menarik dari levenshtein distance adalah terdapatnya angka-angka yang menjadi batas maksimum dan minimumnya. Hal ini dapat berguna untuk berbagai macam aplikasi yang melakukan komputasi atau perbandingan terhadap angka-angka ini. Berikut beberapa diantara batas-batas tersebut :

1. Levenshtein distance minimum adalah selisih panjang string yang dibandingkan.
2. Levenshtein distance maksimum adalah panjang string yang lebih panjang.
3. Levenshtein distance adalah nol, jika kedua string identic (sama).
4. jika string yang dibandingkan adalah s dan t, levenshtein distance minimum adalah karakter yang ada di s yan ditemukan di t.

7. Pengolahan String

String adalah representasi data dalam computer yang berupa kumpulan dari karater / huruf yang disimpan dalam suatu array / tabel yang memiliki indeks. Dengan demikian, string dapat dengan mudah diakses karakter-karakternya (dengan mangakses indeks tabel) dan membandingkannya dengan string lain (**Wahyudin, 1999).**

8. Similarity Score

Fungsi dari levenshtein distance hanya menghitung berapa banyak perubahan yang dibutuhkan untuk mengubah suatu string menjadi string yang lain, tetapi belum memeriksa kesamaan antara dua buah string. Selanjutnya digunakan *similarity score* untuk menghitung nilai kemiripan (*similarity*) dari dua buah string, dengan rumusan sebagai berikut : similarity = 1 – (jarak levenshtein distance / panjang max) **(Ilmy, 2010).**

9. Mekanisme Perolehan Nilai "Distance" Pada algoritma Levenshtein Distance

Algoritma ini digunakan delam pencarian string dengan pendekatan perkiraan (*approximate string matching*). Dengan pendekatan perkiraan ini, pencarian string target menjadi tidak harus sama persis dengan yang ada di dalam string sumber **(Ilmy, 2010)**.

10. Kalimat

Kalimat adalah satuan bahasa terkecil yang merupakan kesatuan pikiran. Dalam bahasa lisan kalimat diawali dan diakhiri dengan kesenyapan, dan dalam bahasa tulis diawali dengan huruf kapital dan diakhiri dengan tanda titik, tanda seru , atau tanda tanya. Kalimat disusun berdasarkan unsur-unsur yang berupa kata, frasa, dan atau

kausa. Jika disusun berdasarkan pengertian di atasm unsur-unsur tersebut mem-punyai fungsi dan pengertian tertentu yang disebut bagian kalimat. Ada bagian yang tidak dapat dihilangakan, ada pula bagian yang dapat dihilangkan. Bagian yang tidak dapat dihilangkan itu disebut inti kalimat, sedang bagian yang dapat dihilangkan bukan inti kalimat. Bagian inti dapat membentuk kalimat dasar, dan bagian bukan inti dapat membentuk kalimat luas. **(Widjono, 2007)**

11. Struktur Kalimat

Kalimat merupakan sarana komunikatif untuk menyampaikan pikiran atau gagasan kepada orang lain agar dapat dipahami dengan mudah. Komunikasi berlangsung baik atau benar jika menggunakan kalimat yang baik dan benar, yaitu kalimat yang dapat mengekspresikan gagasan secara jelas dan tidak menimbulkan keraguan pembaca atau pendengarnya. Untuk itu, kalimat harus disusun berdasarkan struktur yang benar, pengungkapan gagasan secara baik : singkat, cermat, tepat, jelas, maknanya, dan santun. **(Widjono, 2007)**

12. Pola Kalimat

Kalimat yang jumlah dan ragamnya begitu banyak, pada hakikatnya disusun berdasarkan pola-pola tertentu yang amat sedikit jumlahnya. Penguasaan pola kalimat akan memudahkan pemakai bahasa dalam membuat kalimat yang benar secara gramatikal. Selain itu, pola kalimat dapat menyederhanakan kalimat sehingga mudah dipahami oleh orang lain. Kemudahan itu dapat dirasakan oleh pemakai bahasa dalam mengekspresikan ide-idenya dan dalam memahami informasi yang diungkapkan oleh orang lain sehingga dapat memperkecil kesalahpahaman dalam berkomunikasi. **(Widjono, 2007)**

13. HTML

Hypertext Markup Language (HTML) adalah suatu sistem untuk menambah dokumen dengan label yang menandakan bagaimana teks di dokumen harus disajikan dan bagaimana dokumen dihubungkan bersama-sama. Hubungan *hypertext* cukup andal. Di dalam skema tambahan HTML terdapat kekuatan untuk membuat aplikasi-aplikasi client-server, multimedia, cross-platform, dan interaktif. Untaian ajektif ini bukan hanya hype; sistem semacam itu memang ada. Sistem itu dinamakan *World Wide Web* (juga dikenal WWW atau bisa disingkat Web), terdapat di internet, menyediakan organisasi ke berbagai macam sumber daya computer yang diletakkan di seluruh dunia. (**Priyanto, 2014)**

14. PHP

Menurut dokumen resmi PHP, PHP merupakan singkatan dari *PHP Hypertext Preprocessor*. Ia merupakan bahasa berbentuk skrip yang ditempatkan dalam server dan diproses di server. Hasilnyalah yang dikirim ke klien, tempat pemakai menggunakan browser. Secara khusus, PHP diracang untuk membentuk aplikasi web dinamis. Artinya, ia dapat membentuk suatu tampilan berdasarkan permintaan terkini. Misalnya, anda bisa menampilkan isi *database* ke halaman web. Pada prinsipnya PHP mempunyai fungsi yang sama dengan skrip-skrip seperti ASP (*Active Server Page*), Cold Fusion, ataupun Perl. Namun perlu diketahui bahwa PHP bisa dipakai secara *command line.* Artinya, skrip PHP dapat dijadikan tanpa melibatkan *web server* maupun *browser*. **(Abdul Kadir, 2008)**

15. MYSQL

MySQL merupakan software yang tergolong sebagai DBMS (*Database Management System*) yang bersifat *open source*. *Open source* menyatakan bahwa software ini dilengkapi dengan *source code* (kode yang dipakai untuk membuat MySQL). **(Abdul Kadir, 2008)**

16. Flowchart

Flowchart merupakan gambar atau bagan yang memperlihatkan urutan dan hubungan antar proses beserta instruksinya. Gambaran ini dinyatakan dengan symbol. Dengan demikian setiap symbol menggambarkan proses tertentu. Sedangkan hubungan antar proses digambarkan dengan garis penghubung.

Flowchart ini merupakan langkah awal pembuatan program. Dengan adanya flowchart urutan proses kegiatan menjadi lebih jelas. Jika ada penambahan proses maka dapat dilakukan lebih mudah. Setelah flowchart selesai disusun, selanjutnya pemrogram (programmer) menerjemahkannya ke bentuk program dengan bahasa pemrograman.

PEMBAHASAN

1. Analisa Permasalahan

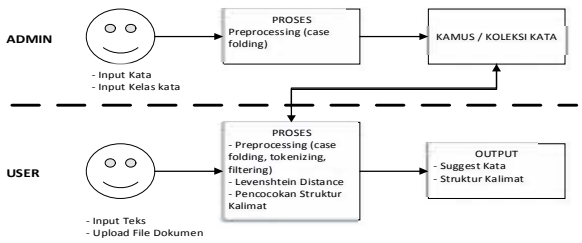
Salah satu manfaat komputer adalah sebagai alat bantu untuk membuat karya tulis. Berbagai aplikasi seperti *microsoft word*, *notepad*, maupun *open office word* sudah ditanamkan pada computer untuk mempermudah pengguna. Meskipun demikian, bukan berarti penggunaannya tanpa masalah. Masalah yang dimaksud adalah adanya

kesalahan pada penulisan ejaan, dalam prakteknya yang sering kali disebabkan karena keteledoran pengguna. Kesalahan tersebut bias disebabkan oleh ketidaktahuan penulisan, kesalahan yang berhubungan erat dengan posisi tombol *keyboard* dan pergerakan jari. Kesalahan-kesalahan yang umumnya terjadi antara lain : penggantian satu huruf, penyisipan satu huruf, penghilanngan satu huruf, maupun penukaran dua huruf berdekatan.

Untuk mengatasi masalah tersebut, diperlukan suatu metode yang dapat digunakan untuk mempermudah pengguna dalam memeriksa adanya kesalahan ejaan dalam karya tulisnya, sekaligus memberikan alternatif untuk memperbaiki kesalahan tersebut. Pengoreksian ejaan ini dapat dilakukan dengan memberikan ejaan kata yang benar, yaitu dengan memberikan usulan ejaan kata yang mirip dengan kata-kata yang ada di dalam kamus. Disamping itu dilakukan pula pegecekan terhadap struktur kalimat dalam dokumen tersebut agar user lebih mengenal tentang kalimat dalam bahasa Indonesia. Oleh Karena itu penulis menerapkan algoritma Levenshtein distance pada sistem koreksi ejaan dan pengenalan struktur kalimat berbahasa Indonesia dengan pendekatan kamus berbasis levenshtein distance. Dimana algoritma levenshtein distance digunakan untuk mencari jarak antara kata dalam dokumen dengan kata-kata dalam kamus sehingga dihasilkan usulan kata jika terdapat kata yang salah dalam dokumen, sedangkan untuk mengecekkan struktur kalimat penulis berdasarkan pada ciri-ciri setiap unsur-unsur kalimat sendiri.

2. Analisa Kebutuhan

2.1 Blok Sistem



Gambar 3.1 Block Diagram Sistem

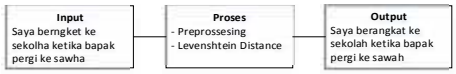
Pada gambar 3.1 diatas adalah gambar blok diagram sistem yang akan digunakan. Terdapat 2 bagian pada gambar tersebut dengan keterangan sebagai berikut :

a) Bagian admin : admin bertugas untuk menginputkan kata-kata yang akan dijadikan acuan dalam sistem ini. Pada bagian ini proses yang dilakan oleh sistem adalah preprosessing case folding yaitu merubah kata yang di inputkan oleh admin menjadi huruf kecil (*lowercase*) sehingga nantinya akan mempermudah sistem dalam pencari kata.

b) Bagian User : user dapat menginputkan kata, kalimat atau paragraph yang akan dikoreksi, selain itu user juga dapat mengunggah (*upload*) dokumen yang akan di koreksi. Proses yang dilakukan pada bagian ini adalah proses preprosecing(case folding, tokenizing, filtering), proses selanjutnya adalah proses perhitungan jarak dengan levenshtein distance dan pencocokan dengan struktur kalimat masing-masing. Hasil yang akan digapatkan oleh user adalah usulan kata yang diberikan oleh sistem sehingga user bisa memilih apakah user akan merubah kata tersebut atau tidak. Jika ada kata yang belum terdaftar dalam kamus maka user dapat memasukkannya dalam kamus.

2.2 *Input / Output* Sistem

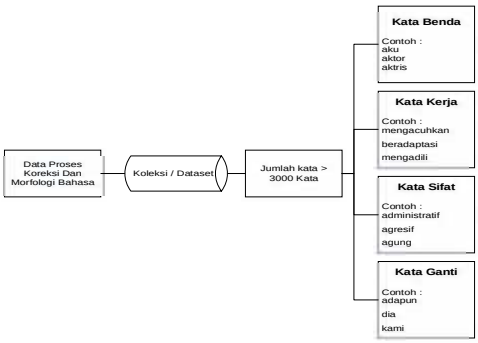
User menginputkan teks yang akan dikoreksi oleh sistem sehingga menghasilnya *suggest* kata jika terdapat kata dengan ejaan yang salah. Berikut contoh teks inputan user.



Gambar 3.2 *Input* Teks user

2.3 Analisa Data

Data yang digunakan dalam sistem ini berupa kata-kata serta kelas kata yang di inputkan oleh admin sebanyak lebih dari 3000 kata. Serta kata yang diinputkan oleh user jika ada kata yang seharusnya ada dalam kamus.



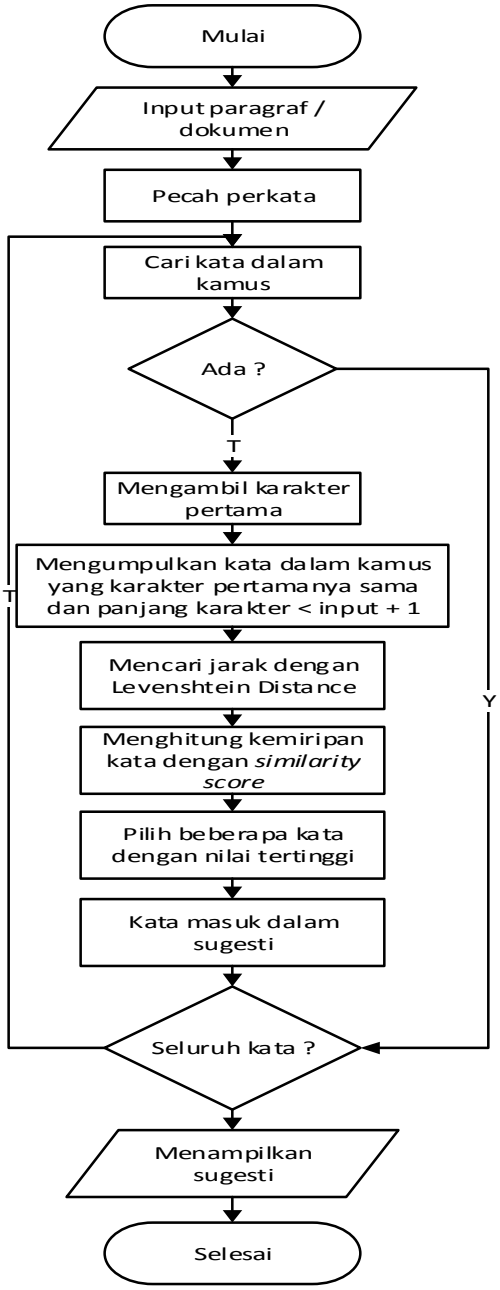
Gambar 3.3 Ilustrasi jumlah kata dalam kamus

Tabel 3.1 Kamus Struktur Kalimat

No	Kata	Keterangan
1	Terabaikan	Predikat
2	Mengabsen	Predikat
3	Saya	Subjek / objek
4	Pergi	Predikat
5	Ke sekolah	Keterangan
6	Memasak	Predikat
7	Pergi	Predikat
8	Naik	Predikat
9	Naga	Subjek / objek
10	Dengan blender	Keterangan
11	Pintu	Subjek / objek
12	Dia	Subjek / objek
13	Mereka	Subjek / objek
14	Membawakan	Predikat
15	Membelikan	Predikat
16	Mengikat	Predikat
17	Berangkat	Predikat
18	Dengan serius	Keterangan

3. Analisa algoritma

3.1 Lavenshtein Distance



Gambar 3.4 Flowchart Levenshtein Distance

Pada flowchart yang ditunjukkan pada gambar 3.4 merupakan flowchart dari proses koreksi ejaan pada sistem, dimana algoritma yang dipakai adalah levenshtein distance. Inputan yang berupa paragraf atau dokumen sebelum dilakukan proses levenshtein distance dipecah menjadi kata.

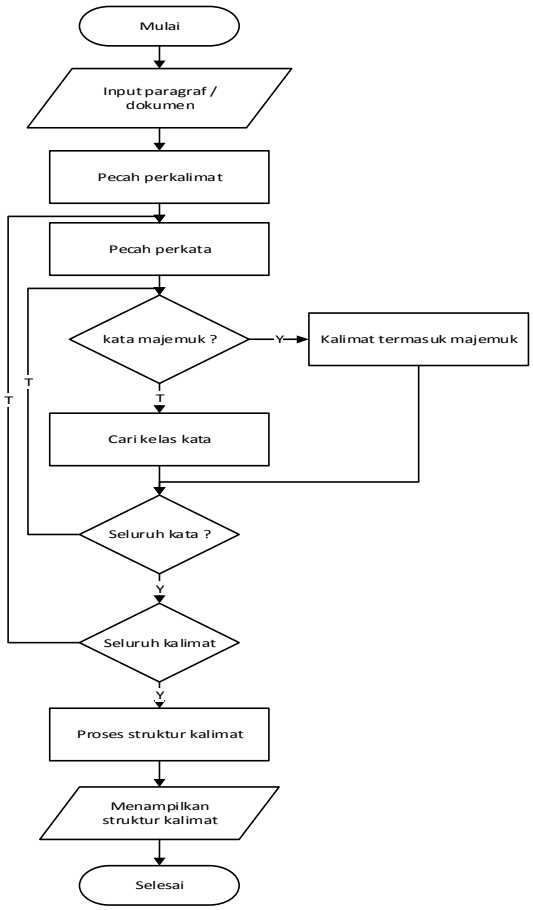
Proses selanjutnya adalah dengan mencari kata dari paragraf tersebut dalam kamus dengan tujuan jika sudah terdapat kata tersebut dalam kamus maka tidak akan mengalami proses perhitungan levenshtein distance, tetapi jika kata tersebut tidak ada dalam kamus kata maka akan menglami proses perhitunga levenshtein distance dan *similarity score*.

Ada 3 macam operasi utama yang dapat dilakukan oleh algoritma levenshtein distance yaitu :

a. Operasi Pengubahan Karakter
b. Operasi Penambahan Karakter
c. Operasi Penghapusan Karakter

Fungsi dari levenshtein distance hanya menghitung berapa banyak perubahan yang dibutuhkan untuk mengubah suatu string menjadi string yang lain, tetapi belum memeriksa kesamaan antara dua buah string. Selanjutnya digunakan *similarity score* untuk menghitung nilai kemiripan (*similarity*) dari dua buah string, dengan rumusan sebagai berikut : similarity = 1 – (jarak levenshtein distance / panjang max). Hasil dari proses lavenshtein distance dan similarity score akan diambil maksimal 3 kata dengan nilai paling tinggi yang akan dijadikan sebagai rekomendasi yang nantinya user sendiri yang akan memilih kata mana yang benar.

3.2 Pengenalan Struktur Kalimat



Gambar 3.5 Flowchart Pengenalan Struktur Kalimat

Pada gambar 3.5 menunjukkan bahwa proses pertama dari pengenalan struktur kalimat pada sistem adalah dengan melakukan pemecahan perkalimat kemudian dipecah perkata sama dengan proses pada koreksi ejaan dengan algoritma levenshtein distance. Proses selanjutnya adalah dengan melakukan pengecekkan apakah kalimat tersebut terdapat kata majemuk atau tidak, jika terdapat kata majemuk maka masuk ke dalam kategori kalimat majemuk dengan jenis majemuk sesuai dengan ciri-ciri dari kalimat majemuk itu sendiri. Setelah proses tersebut proses selanjutnya adalah pencarian kelas kata dalam kamus yang bertujuan untuk mengelompokkan kata tersebut berdasarkan kelas katanya sehingga dapat dilakukan proses pembuatan struktur kalimat.Pada pembuatan struktur kalimat sistem melakukan pengecekan kata tersebut berdasarkan kelas

katanya agar dapat dikelompokkan berdasarkan unsur-unsur kalimatnya berdasarkan ciri-ciri dari tiap unsur-unsur kalimat. Setelah semua kata telah dikelompokkan berdasarkan unsur-unsur kalimatnya makan kalimat tersebut akan ditampilkan beserta unsur-unsur dari tiap katanya.

3.3 Study Kasus

User menginputkan kalimat "Saya berangket ke sekolha ketika bapak pergi ke sawha", proses selanjutnya adalah preprocessing sebagai berikut :

1. *Case Folding* : Merupakan tahapan yang mengubah semua huruf dalam dokumen menjadi huruf kecil. Hanya huruf 'a' sampai 'z' yang diterima. Karakter selain huruf dihilangkan dan dianggap delimiter (pembatas). Di misalkan user menginputkan teks Saya berangket ke sekolha ketika bapak pergi ke sawha. Contho dari tahapan ini ditunjukkan pada Tabel 3.1 berikut :

Tabel 3.2 Proses Case Folding

Teks Input	Hasil Case Folding
Saya	saya
berangket	berangket
ke	ke
sekolha	sekolha
Ketika	ketika
bapak	bapak
pergi	pergi
ke	ke
sawha	sawha

2. *Tokenizing / Parsing* : Tahap pemotongan string input berdasarkan tiap kata yang menyusunnya. Selain itu, spasi digunakan untuk memisahkan antar kata tersebut. Contoh dari tahapan ini sebagai berikut yang ditunjukkan pada Tabel 3.2 :

Tabel 3.3 Proses Tokenizing/Parsing

Teks Input	Hasil Tokenizing
saya berangket ke sekolha ketika bapak pergi ke sawha	saya
	berangket
	ke
	sekolha
	ketika
	bapak
	pergi
	ke
	sawha

3. *Filtering* : mengambil kata-kata penting dari hasil tokenizing. Proses filtering dapat menggunakan algoritma *stoplist* (membuang kata yang kurang penting) atau *wordlist* (menyimpan kata yang penting). *Stoplist / stopword* adalah kata-kata yang tidak deskriptif yang dapat dibuang dalam pendekatan *bag-of-words*. Contoh dari tahapan ini adalah sebagai berikut yang ditunjukkan pada Tabel 3.3.

Tabel 3.4 Proses *Filtering*

Teks Input	Hasil Filtering
Saya	saya
berangket	berangket
ke	sekolha
sekolha	bapak
ketika	pergi
bapak	sawha
pergi	
ke	
sawha	

3.3.1 Penerapan Algoritma Levenstein Distance Perbandingan Kata Saya

Tabel 3.5 Kamus Kata

Kamus Kata
sama
saya
sayap

Sistem melakukan perbandingan kata saya dengan tiap kata yang telah terpilih pada tabel 3.4 menggunakan algoritma levenshtein distance.

- Kata yang dibandingkan : saya ↔ sama

Tabel 3.6 Matrik Levenshtein Distance saya ↔ sama

		s	a	m	a
	0	1	2	3	4
s	1	0	1	2	3
a	2	1	0	1	2
y	3	2	1	1	2
a	4	3	2	1	1

Dari matrik diatas diperoleh nilai levenshtein distance yaitu nilai dari kolom terkhir dari baris terkhir berupa nilai 1. Setelah itu menghitung nilai kesamaan kedua string sebagai berikut :

$$\text{Sim(saya,sama)} = 1 - \frac{LD(saya,sama)}{\max(|saya|,|sama|)}$$

$$= 1 - \frac{1}{\max(4,4)}$$

$$= 1 - \frac{1}{4}$$

$$= 0{,}75$$

Dari matrik diatas dapat dilakukan proses perhitungan kesamaan dua string yang dibandingkan dengan nilai levenshtein distance 0 sebagai berikut :

$$\text{Sim(saya,saya)} = 1 - \frac{LD(saya,saya)}{\max(saya,saya)}$$

$$= 1 - \frac{0}{4}$$

$$= 1$$

Proses selanjutnya adalah membandingkan kata saya dengan kata sayap sebagai berikut :

Kata yang dibandingkan : saya ↔ sayap

$$\text{Sim(saya,sayap)} = 1 - \frac{LD(saya,sayap)}{\max(saya,sayap)}$$

$$= 1 - \frac{1}{5}$$

$$= 0{,}8$$

Setelah proses perbandingan kata saya dengan beberapa kata dalam kamus telah selesai maka dilakukan proses pemilihan rekomendasi kata sebagai berikut :

Perbandingan kata saya ↔ sama = 0,75

Perbandingan kata saya ↔ saya = 1

Perbandingan kata saya ↔ sayap = 0,8

Dari nilai diatas dipilih kata dengan hasil perbandingan 1 atau dengan hasil perbandingan paling besar, dalam kasus ini terpilih kata saya. Dikarenakan kata inputan sama dengan kata dalam kamus maka tidak masuk dalam rekomendasi kata.

Perbandingan Kata Berangket

Pada proses perbandingan ini kata yang dibandingkan adalah kata berangket dengan kata dalam kamus yang tersedia sebagai berikut :

Tabel 3.7 Kamus Kata

Kamus Kata
beranjak
berangkat

$$Sim(berangket,beranjak) = 1 - \frac{LD(berangket,beranjak)}{\max(berangket,beranjak)}$$

$$= 1 - \frac{4}{9}$$

$$= 0{,}56$$

Perbandingan kata berangket ↔ berangkat

$$Sim(berangket,berangkat) = 1 - \frac{LD(berangket,berangkat)}{\max(berangket,berangkat)}$$

$$= 1 - \frac{1}{9}$$

$$= 0{,}89$$

Dari proses perbandingan kata berangket dengan beberapa kata dalam kamus, selanjutnya proses pemilihan rekomendasi kata. Sebagai berikut :

Perbandingan kata berangket ↔ beranjak = 0,56

Perbandingan kata berangket ↔ berangkat = 0,89

Dari nilai diatas dikarenakan tidak ada yang bernilai 1 maka dipilih nilai perbandingan yang terbesar yaitu 0,89. Dalam kasus ini kata inputan tidak sama dengan kata dalam kamus yang tersedia sehingga kata dengan perbandingan yang paling besar masuk dalam rekomendasi kata.

Perbandingan Kata Sekolha

Pada proses perbandingan kata sekolha terdapat beberapa kata yang terpilih dari kamus kata, kata-kata yang terpilih sebagai berikut :

Tabel 3.8 Kamus Kata

Kamus Kata
Sekolah
Sekongkol
Seksama

Selanjutnya sistem akan melakukan proses perbandingan dengan kata yang terpilih dari kamus kata yang tersedia tersebut.

Dari ketiga proses perbandingan tersebut diperoleh hasil sebagai berikut :

- Perbandingan sekolha ↔ sekolah = 0,72
- Perbandingan sekolha ↔ sekongkol = 0,44
- Perbandingan sekolha ↔ seksama = 0,57

Maka kata sekolah masuk dalam kategori rekomendasi kata dari kata inputan sekolha.

Perbandingan kata Bapak

Pada proses perbandingan kata bapak terdapat beberapa kata yang terpilih dari kamus kata yang tersedia, kata-kata yang terpilih sebagai berikut :

Tabel 3.9 Kamus Kata

Kamus Kata
Bapak

Dari kata dalam kamus tersebut dilakukan proses perbandingan kata bapak pada inputan dengan kata bapak dalam kamus

$$Sim(bapak,bapak) = 1 - \frac{LD(bapak,bapak)}{\max(bapak,bapak)}$$

$$= 1 - \frac{0}{5}$$

$$= 1$$

Dari hasil perhitungan perbandingan antara kata inputan bapak dengan kata dalam kamus berupa kata bapak diperoleh hasil 1, sehingga dua kata tersebut dianggap sama dan tidak masuk dalam kategori rekomendasi kata.

Perbandingan kata Pergi

Pada proses perbandingan kata pergi terdapat beberapa kata yang terpilih dari kamus kata yang tersedia, kata-kata yang terpilih sebagai berikut :

Tabel 3.10 Kamus Kata

Kamus Kata
pergi

$$Sim(pergi,pergi) = 1 - \frac{LD(pergi,pergi)}{\max(pergi,pergi)}$$

$$= 1 - \frac{0}{5}$$

$$= 1$$

Dari hasil perhitungan perbandingan antara kata inputan pergi dengan kata dalam kamus berupa kata pergi diperoleh hasil 1, sehingga dua kata tersebut dianggap sama dan tidak masuk dalam kategori rekomendasi kata.

Perbandingan Kata Sawha

Pada proses perbandingan kata sawha terdapat beberapa kata yang terpilih dari kamus kata yang tersedia, kata-kata yang terpilih sebagai berikut :

Tabel 3.11 Kamus Kata

Kamus kata
satwa
sawi
sawah

Dari ketiga proses perbandingan tersebut diperoleh hasil sebagai berikut :

- Perbandingan sawha ↔ satwa = 0,6
- Perbandingan sawha ↔ sawi = 0,6
- Perbandingan sawha ↔ sawah = 0,6

Dalam kasus ini tidak terdapat nilai 1 dalam hasil perbadingan maka diambil nilai yang paling besar pada perbandingan tersebut, dikarenakan dalam kasus ini nilai yang paling besar lebih dari satu maka nilai yang sama tersebut dimasukkan kedalam kategori rekomendasi kata. Sehingga nantinya user yang akan memilih kata mana yang cocok.

Pengenalan Struktur Bahasa Indonesia

Pada proses ini dalam satu kalimat diambil perkata untuk dikelompokkan sesuai dengan kelas katanya masing-masing. Dalam kasus ini pengelompokan kata dapat dilihat pada tabel 3.12 berikut :

Tabel 3.12 Kelas Kata

Kata	Kelas Kata
saya	Kata ganti
berangkat	Kata kerja
ke	Kata depan
sekolah	Kata benda
ketika	Kata benda
bapak	Kata benda
pergi	Kata berja
ke	Kata depan
sawah	Kata benda

Setelah proses pengempokan kata berdasarkan kelas kata, proses selanjutnya adalah pengecekan terhadap struktur dari kalimat tersebut berdasarkan ciri-ciri dari unsur kalimat.

Dan setelah kalimat tersebut digabungkan maka termasuk dalam kalimat majemuk bertingkat dengan anak kalimat keterangan waktu dengan pemisah kata ketika. Sehingga dalam dituliskan sebagai berikut :

Saya berangkat ke sekolah ketika bapak pergi ke sawah

S P K(tempat) S P K(tempat)

IMPLEMENTASI DAN PENGUJIAN

1. Implementasi Sistem

Implementasi adalah tahap sebuah perancangan selesai dilakukan dan selanjutnya di implementasikan pada bahasa pemrogrman yang digunakan. Implementasi system ini merupakan tahap melaksanakan sistem sehingga siap untuk dioperasikan. Implementasi bertujuan untuk mengkonfirmasi modul-modul perancangan,

sehingga pengguna dapat memberi masukan kepada pengembang sistem.

1.1 Implementasi Antar Muka

Menggambarkan rancangan bentuk tampilan grafis dari sistem. Berikut adalah tampilan grafis :

Gambar 4.1 Halaman Utama

Gambar 4.2 Halaman Upload

Gambar 4.3 Halaman Koreksi dan Rekomendasi Kata

Gambar 4.4 Daftar Saran kata

Gambar 4.5 Matrik Levenshtein Distance

Gambar 4.6 Daftar Kalimat

Gambar 4.7 Struktur Kalimat

Gambar 4.8 Hasil Preprocessing

Gambar 4.9 Halaman Riwayat

Gambar4.10 Halaman Login

2. Pengujian

2.1 Pengujian

Pengujian terhadap aplikasi dilakukan dengan 3 tahap yaitu tahap pengujian waktu, tahap pengujian kata, dan tahap pengujian kalimat.

2.2 Tahap Pengujian Waktu

Tahap pengujian waktu dilakukan dengan menginputkan beberapa teks kemudian di cek waktu dari proses aplikasi dalam menentukan rekomendasi kata. Tahap ini dapat dilihat pada tabel 4.1 berikut :

Tabel 4.1 Tabel Pengujian Kata

No	Jumlah Kata Atau Halaman	Waktu
1	63 Kata	1 detik
2	280 Kata / 1 halaman	4 detik
3	980 kata / 6 halaman	12 detik
4	1608 Kata / 10 halaman	24 detik

Dari beberapa pengujian yang telah dilakukan aplikasi ini dapat mengoreksi kata dengan dengan memberikan rekomendasi kata yang tepat untuk pengguna, dengan jumlah kata maksimal 980 kata / 10 halaman. Tidak disarankan untuk mengreksi kata diatas jumlah tersebut.

2.3 Tahap Pengujian Kata

Tahap pengujian kata dilakukan dengan menginputkan beberapa kata yang salah kemudian dicek apakah saran kata yang benar terdapat pada urutan pertama atau teratas jika saran kata yang berada diurutan teratas maka masuk dalam kata benar dalam pengujian ini. Setelah dilakukan pengujian sebanyak tiga tahap pada pengujian kata dimana setiap tahap kata yang diinputkan sebanyak 100 kata salah diperoleh nilai keakuratan sebesar 86%. Pengujian ini dapat dilihat pada tabel 423 berikut :

Tabel 4.2 Pengujian Kata

No	Tahap	Rata-rata
1	Tahap 1	0.86
2	Tahap 2	0.87
3	Tahap 3	0.85

2.4 Tahap Pngujian Kalimat

Pengenalan struktur kalimat Bahasa Indonesia diuji dengan memberikan kalimat dan output dari sistem kepada 6 penguji dimana setiap 1 orang penguji mendapatkan 5 kalimat sehingga jumlah seluruh kalimat yang diuji adalah 30 kalimat dan diperoleh nilai keakuratan sebesar 76.66%.

PENUTUP

1. Kesimpulan

Berdasarkan pembahasan dan penelitian yang telah dipaparkan pada bab sebelumnya, maka dapat diambil kesimpulan berikut :

1. Aplikasi ini bukan untuk menentukan hasil akhir, melainkan hanya rekomendasi saja.
2. Aplikasi hanya dapat mengkoreksi dokumen dengan jumlah halaman sebanyak 6 lembar.
3. Untuk pengkoreksi ejaan dapat dalam aplikasi ini manggunakan metode levenshtein distance mempunyai tingkat akurasi sebesar 86%.
4. Untuk mengenalan struktur kalimat aplikasi ini mempunyai tingkat akurasi sebesar 76.66%.

2. Saran

Beberapa saran yang perlu disampaikan untuk pengembangan sistem koreksi ini adalah :

1. Dapat ditambahkan data kamus lain sebagai perbandingan tingkat akurasi dari sistem koreksi ini.
2. Dapat ditambahkan jenis dokumen yang bias diupload.
3. Dalam penguraian struktur kalimat berdasarkan cirinya bukan satu-satunya cara yang dapat digunakan, masih ada beberapa metode yang lebih akurat dalam penguraian struktur kalimat.

DAFTAR PUSTAKA

Adiwidya. Algoritma Levenshtein Distance Dalam Pendekatan Approximate String Matching. Bandung. Departemen Teknik Informatika ITB. 2009

Al-Bahra. Analisa dan Desain Sistem Informasi. Jakarta. Elek Media Komutindo. 2005

Artrianto, Detha. Aplikasi Penentu Struktur Kalimat Bahasa Indonesia. Jarakta. Fakultas Ilmu Komputer Dan Teknologi Informasi Universitas Gunadarma. 2011

Bing Liu. Web Data Mining. Chicago.

Departement Of Computer Science. University Of Illinois. 2007

Fan, Weiguo. dkk. Tapping Into The Power Of Text Mining. Communication Of ACM. Blacksburg. Pamplin Hall. 2005

Feldman, R., and J. Sanger. The Text Mining Handbook : Advenced Approaches In Analizing Unstruktured Data. England. Cambridge University Press. 2007

Hasugian, Jonner. Penggunaan Bahasa Ilmiah dan Kosa Kata Terkendali dalam Sistem Temu Balik Informasi Berbasis Teks. Medan. Jurnal Studi Pustaka Perpustakaan dan Informasi. 2006

Ilmy. Penerapan Algoritma Levenshtein Distance Untuk Mengetahui Kesalahan Pengerjaan Pada Editor Teks. Bandung. Departemen Teknik Informatika ITB. 2010

Kadir, abdul. Pemrograman Web Dinamis Menggunakan php (revisi). Yogyakarta. Andi. 2008

Miller, Thomas. Data and Text Mining A Business Applications Approach. Jersey. Rentice Hall. 2005

Mooers, C.N. Zatocoding Applied to Mechanical Organization of Knowledge. USA. American Documentation. 2002

Nazief, B.A.A., and M. Andriani. Confix-Stripping. Approach to Stemming Algorithm for Bahasa Indonesia. Jakarta. University Of Indonesia.1996

Hidayatullah, priyanto. Pemrograman Web. Bandung. Informatika. 2014

Trevisan. Edit Distance. http://people.eecs.berkeley.edu/. 2001 (diakses 12 April 2017. 03:53)

Wahyudin. Algoritma Trigram Untuk Koreksi Ejaan. Bogor. Departemen Ilmu Komputer IPB. 1999

Widjono. Hs. Bahasa Indonesia Mata Kuliah Pengembangan Kepribadian di Perguruan Tinggi. Jakarta. PT. Grasindo. 2007

Zaenab, Ratu Siti. Efektifitas Temu Kembali Informasi Dengan Bahasa Alami. Sumatra Utara. Jurnal Pustakawan. 2002