Aplikasi Koreksi Kata Berbahasa Indonesia Dengan Metode *k-Nearest Neighbor* Berbasis Web

Iim Abdurrohim, Ajeng Saputri

Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Nasional PASIM
Jl. Dakota No. 8A Bandung, Indonesia
iim2loey@pasim.ac.id

Abstrak

Aplikasi Koreksi Kata Berbahasa Indonesia dibangun dengan tujuan mengkoreksi kata – kata yang tidak sesuai dengan ejaan. Terdapat beberapa metode dalam mengimplementasikan Aplikasi koreksi kata berbahasa Indonesia ini, salah satunya adalah k-Nearest Neighbor (k-NN). Penelitian Aplikasi koreksi kata menggunakan metode k-NN dengan perhitungan jarak menggunakan levenshtein distance dimana jarak yang dihitung adalah jarak antara dua string.

Pada penelitian ini dilakukan salah satu penambahan tahap preprocessing yaitu tokenizing sehingga setiap kata pada teks dapat diidentifikasi dengan mudah. Aplikasi ini dikembangkan menggunakan bahasa pemrograman berbasis web yaitu python karena memiliki fasilitas yang memadai untuk teks preprocessing dengan dilengkapi library yang menunjang aplikasi koreksi kata. Aplikasi ini menyimpan sebuah kamus dengan format dic dimana kamus tertanam di dalam aplikasi.

Kata Kunci : Aplikasi Koreksi Kata, k-NN, pra pemrosesan, penandaan, jarak *levenshtein*.

Abstract

The Indonesian Word Correction Application was built with the aim of correcting words that do not match the spelling. There are several methods in the implementation of this application. One of them is k-Nearest Neighbor (k-NN). Application Research using k-NN method with distance calculation using levenshtein distance where distance is calculated distance between two strings.

In this research, one of the preprocessing preparatory steps is tokenizing so that every word in the text can be easily. This application was developed using a web-based programming language that is python because it has adequate facilities for preprocessing text with a library equipped to support the application. This app stores a dictionary with a dicable format in which the dictionary is embedded within the application.

.

Keywords*: Word Correction Application, k-NN, preprocessing, tokenizing, levenshtein distance.*

I. PENDAHULUAN

1.1. Latar Belakang

Tulisan merupakan bentuk gagasan si penulis yang hendak disampaikan pada orang lain. Tulisan ditampilkan dalam bentuk teks yang terdiri dari kumpulan huruf atau angka yang dituliskan dalam suatu bahasa tertentu. Tulisan akan menarik untuk dibaca bila dicetak dengan tata letak dan penampilan yang baik. Namun isi dari tulisan tidak kalah pentingnya, karena isi merupakan pokok dari tulisan yang akan diterima sebagai informasi oleh pembaca, sebuah tulisan harus mudah dimengerti oleh pembaca. Dengan isi tulisan yang baik maka informasi yang disampaikan akan diterima dengan sempurna.

Kesalahan dapat terjadi dimana saja dan kapan saja termasuk seorang penulis tidak dapat terhindar dari kesalahan dalam proses penulisan naskahnya. Kesalahan tersebut umunya berupa salah ketik atau salah eja yang dapat mempengaruhi informasi yang ingin disampaikan penulis melalui

tulisannya. Hal seperti ini sering terjadi pada mahasiswa tingkat akhir yang sedang menyusun skripsi. Sebenarnya hal tersebut bisa dihindari apabila penulis melakukan proses penyuntingan yaitu pemeriksaan kembali dengan teliti tulisan yang dibuatnya. Namun hal tersebut merupakan hal yang membosankan dan menyita banyak waktu, sebab proses yang sama harus dilakukan berulang kali untuk menghasilkan tulisan yang benar – benar bebas dari kesalahan. Karena proses penyuntingan manual yang banyak memakan waktu, maka dibutuhkan suatu aplikasi koreksi yang dapat membantu seorang penulis untuk memeriksa kesalahan pada tulisan secara otomatis.

Beberapa web yang mendukung kebutuhan koreksi kata sudah tersedia namun belum ada yang secara khusus memfasilitasi koreksi kata dalam bahasa Indonesia. Ada beberapa metode yang dapat digunakan untuk pembangunan aplikasi koreksi kata ini, salah satunya dengan pendekatan *k-Nearest Neighbor* (k-NN). k-NN ini merupakan algoritma *supervised learning* dimana hasil dari instance yang baru diklasifikasikan berdasarkan mayoritas dari k-tetangga terdekat. Metode menggunakan algoritma ini bertujuan untuk mengklasifikasi objek baru berdasarkan atribut dan sample dari data training.

Dengan latar belakang yang disampaikan tersebut diatas penulis mengembangan aplikasi koreksi kata ini dengan judul penelitian "Aplikasi Koreksi Kata Berbahasa Indonesia dengan Metode *k-Nearest Neighbor* Bebasis Web".

1.2. Identifikasi Masalah

Berdasarkan latar belakang yang telah dikemukakan di atas, penulis mengidentifikasi beberapa masalah sebagai berikut:

1. Bagaimana membangun aplikasi koreksi kata untuk penulisan dalam bahasa Indonesia.
2. Bagaimana mengaplikasikan suatu metode *k-Nearest Neighbor* pada sebuah bentuk aplikasi koreksi kata.

1.3. Maksud dan Tujuan

1.3.1. Maksud penelitian

Penelitian ini dilakukan untuk membangun aplikasi yang mempermudah siapa saja dalam penyuntingan teks dengan menggunakan metode k-Nearest Neighbor yang diharapkan dapat mengkoreksi teks dengan akurasi yang baik.

1.3.2. Tujuan Penelitian

Adapun tujuan dari penelitian adalah sebagai berikut:

1. Untuk menghasilkan aplikasi berbasis web yang berguna untuk koreksi kata dalam bahasa Indonesia.
2. Untuk membangun aplikasi koreksi kata dengan mengimplementasikan metode K-Nearest Neighbor.

1.4. Batasan Masalah

Batasan masalah dalam Penelitian ini adalah:

1. Kamus tersimpan di dalam aplikasi.
2. Kata-kata yang digunakan menggunakan referensi kamus besar bahasa Indonesia (KBBI).
3. Kata yang digunakan adalah kata-kata yang umum dipakai.
4. Penambahan kata baru tidak serta merta menambah referensi kata ddidalam kamus, tetapi disimpan didalam *cookies browser user*.

II. TINJAUAN PUSTAKA

Pada bab ini dibahas mengenai definisi terkait dengan judul penelitian serta teori-teori yang akan mendasari dilakukannya penelitian berdasarkan materi yang didapatkan dari berbagai sumber yang dapat meningkatkan pemahaman dan wawasan terkait dengan topik judul.

2.1. Pengertian Aplikasi

Aplikasi adalah suatu sub kelas perangkat lunak komputer yang memanfaatkan kemampuan komputer langsung untuk melakukan suatu tugas yang diinginkan pengguna. Biasanya dibandingkan dengan perangkat lunak sistem yang mengintegrasikan berbagai kemampuan komputer, tetapi tidak secara langsung menerapkan kemampuan tersebut untuk mengerjakan suatu tugas yang menguntungkan pengguna. Contoh utama perangkat lunak aplikasi adalah pengolah kata, lembar kerja dan pemutar media. (Wikipedia)

2.2. Pengertian Koreksi Kata

Koreksi dalam kamus besar bahasa Indonesia memiliki arti pembetulan, perbaikan, pemeriksaan. Sedangkan kata menurut Wikibooks

adalah satuan bahasa terkecil yang dapat berdiri sendiri.

Koreksi kata akan dibuat menjadi sebuah aplikasi yang berbasis pada kecerdasan buatan. Aplikasi yang akan dirancang menggunakan pendekatan machine learning. Dengan machine learning, mesin dilatih untuk mengetahui dalam menilai / mengkategorikan kata.

2.3. Machine Learning

Machine learning adalah salah satu disiplin ilmu dari Computer Science yang mempelajari bagaimana membuat komputer/ mesin itu mempunyai suatu kecerdasan seperti manusia. Agar mempunyai suatu kecerdasan, komputer/ mesin harus dapat belajar. Dengan kata lain, machine learning adalah suatu bidang keilmuan yang berisi tentang pembelajaran komputer/ mesin untuk menjadi cerdas. Di sisi lain machine learning diartikan sebagai salah satu jadwal dari kecerdasan buatan yang membahas mengenai pembangunan sistem yang didapatkan berdasarkan pembelajaran data. Inti machine learning adalah representasi dan generalisasi. Arthur Samuel (1959) mendefinisikan machine learning adalah bidang studi yang memberikan kemamuan untuk belajar tanpa diprogram secara eksplisit (Simon, 2013).

Tipe algoritma pada machine learning:

1. Supervised Learning
   Machine learning yang bertugas menyimpulkan fungsi dari data training berlabel (Mohri, Rostamizadeh dan Talwalkar, 2012).
2. Unsepervised Learning
   Unsupervised learning adalah jenis algoritma pembelajaran mesin yang digunakan untuk menarik kesimpulan dari data set yang terdiri dari input data tanpa tanggapan berlabel (Mathworks.com, 2013).
3. Semi Supervised Learning
   Semi supervised learning adalah paradigma pembelajaran berkaitan dengan studi tentang bagaimana komputer dan sistem alam seperti manusia belajar di hadapan kedua data yang berlabel dan tidak berlabel (Zhu dan Goldberg, 2009).

Pada penelitian ini metode machine learning yang akan digunakan adalah *supervised learning*. Katena data set yang digunakan telah memiliki label / kategori.

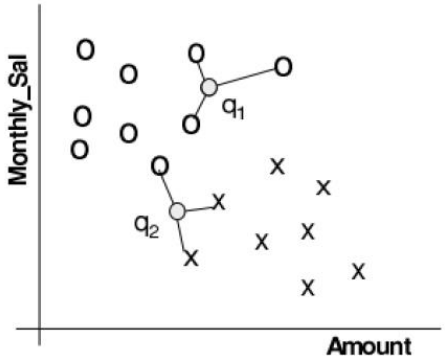
Supervised Learning menggunakan kumpulan data yang sudah berlabel / terkategorisasi yang bertujuan untuk membuat sebuah model classifier yang menghasilkan prediksi hasil pengkategorisasian data yang baru (Mathworks.com, 2013). Kumpulan data yang berlabel terus dinamakan juga sebagai training data.

Salah satu metode yang menggunakan tipe algoritma supervised learning adalah *k-Nearest Neigbor*. Maka dari itu dalam aplikasi ini metode yang akan digunakan adalah metode *k-Nearest Neigbor*.

2.4. *K-Nearest Neighbor* (k-NN)

K-Nearest Neighbor (k-NN) adalah sebuah metode untuk melakukan klasifikasi terhadap objek berdasarkan data training yang jaraknya paling dekat dengan objek tersebut. Klasifikasi k-NN memiliki dua tahap, yang pertama adalah penentuan tetangga terdekat dan yang kedua adalah penentuan kelas menggunakan tetangga masing–masing (Cunningham dan Delany, 2007).

Pada contoh ini penentuan kelas untuk q1 dapat langsung diketahui. Ketiga tetangganya berada pada kelas O sehingga q1 diklasifikasikan pada kelas O. Situasi untuk q2 sedikit lebih rumit karena memiliki dua tetangga dari kelas X dan satu tetangga dari kelas O. Hal ini dapat diselesaikan dengan voting mayoritas secara sederhana atau dengan memberikan voting berdasarkan bobot jaraknya (Cunningham dan Delany, 2007). Gambar 3 merupakan contoh untuk 3-k-NN:



Gambar 1. Contoh Sederhana 3-Nearest Neighbor Clasification

Similarity metric dan distance metric sering digunakan untuk mengukur kedekatan antara dua objek. Istilah metric memiliki arti formal dalam

matematika. Sebuah metric harus sesuai dengan empat kriteria berikut (dimana d (x, y) mengacu pada jarak antara dua objek x dan y):

1. d (x, y) ⩾ 0; non-negativity
2. d (x, y) = 0 hanya jika x = y; identity
3. d (x, y) = d (y, x); symmetry
4. d (x, z) ⩾ d (x, y) + d (y, z); triangle inequality

Pada Aplikasi ini metode yang dipakai untuk mencari jarak atau similarity adalah levenshtein distance. Tetangga terdekat dipilih berdasarkan pada distance/similarity metric. Kemudian ada berbagai cara dimana tetangga terdekat dapat digunakan untuk menentukan suatu kelas dari query. Pendekatan yang paling mudah adalah dengan menetapkan kelas mayoritas diantara tetangga yang terdekat (Cunningham dan Delany, 2007).

Pada Aplikasi ini, proses klasifikasi ditentukan menggunakan voting mayoritas. Dimana akan dipilih kategori yang muncul paling banyak pada K tetangga.

Berikut adalah gambaran umum algoritma k-NN (Satheesh dan Patel, 2010) :

Input: training set G', query xo
Output: label kelas dari x_o
P : jumlah fitur pada training set
Training set G ' , merupakan $\{xi, yi\}_1^N$ dengan $x_i \in R^p$ dan $y_i \in \{1, \dots, m\}$.
N : total jumlah data pada training set
1. Menghitung distance / similarity diantara data x_i dan x_o.
2. Sampel pelatihan diurutkan secara ascending urutan berdasarkan ukuran distance / similarity.
3. Pilih nilai dari k untuk menentukan jumlah tetangga terdekat.
Menentukan label kelas pada x_o dengan cara memilih suara terbanyak dari tetangga terdekat yang didapatkan pada langkah 3.

2.5. *Levenshtein Distance*

Algoritma Levenshtein Distance ditemukan oleh Vladimir Levenshtein, seorang ilmuan asal Rusia pada tahun 1965 (Janowski, 2010), algoritma ini sering juga disebut dengan Edit Distance (Husain,2013). Yang dimaksud dengan distance adalah jumlah modifikasi yang dibutuhkan untuk mengubah suatu bentuk string ke bentuk string yang lain, sebagai contoh hasil penggunaan algoritma ini, string "komputer" dan "computer" memiliki distance 1 karena hanya perlu dilakukan satu operasi saja untuk mengubah satu string ke string yang lain. Dalam kasus dua string di atas, string "computer" dapat menjadi "komputer" hanya dengan melakukan satu penukaran karakter 'c' menjadi 'k' (Andhika, 2010).

Algoritma Levenshtein Distance digunakan secara luas dalam berbagai bidang, misalnya mesin pencari, pengecek ejaan (spell checking), pengenal pembicaraan (speech recognition), pengucapan dialek, analisis DNA, pendeteksi pemalsuan, dan lain-lain (Adiwidya, 2009). Algoritma Levenshtein Distance bekerja dengan menghitung jumlah minimum pentranformasian suatu string menjadi string lain yang meliputi penghapusan, penyisipan, dan penukaran (Husain, 2013).

Untuk menentukan Levenshtein Distance antara dua kata kita membutuhkan persamaan matriks sebagai berikut (Pryana, Dewi, dll, 2013).

$$\mathrm{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0, \\ \min \begin{cases} \mathrm{lev}_{a,b}(i-1,j)+1 \\ \mathrm{lev}_{a,b}(i,j-1)+1 \\ \mathrm{lev}_{a,b}(i-1,j-1)+1_{(a_i \neq b_j)} \end{cases} & \text{otherwise.} \end{cases}$$

Keterangan:

$Lev_{a,b}$ = matriks $lev_{a,b}$

i = baris matriks

j = kolom matriks

Dengan $1(a_i \neq b_j)$ sebagai fungsi indikator atau cost. Cost bernilai 0 jika $a_i = b_j$ dan bernilai 1 jika $a_i \neq b_j$ dengan i sebagai karakter pertama dari string a dan j sebagai karakter pertama dari b.

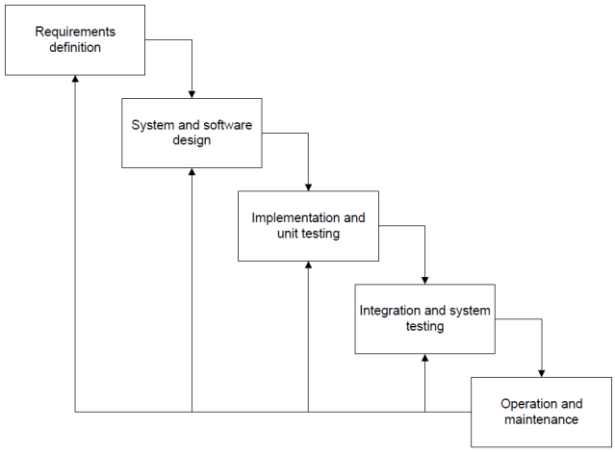
2.6. *Squential Linear/ Waterfall*

Model waterfall adalah sebuah metode pengembangan software yang bersifat sekuensial. Dalam pengembangan perangkat lunak, pendekatan ini cenderung pada pendekatan yang tidak berulang dan fleksibel, karena prosesnya mengalir dalam satu arah. (Wikipedia).

Model waterfall ini mengambil dasar aktivitas proses dari spesifikasi, pengembangan, validasi dan evolusi dan mewakili mereka sebagian terpisah fase proses, seperti spesifikasi persyaratan, perancangan perangkat lunak, implementasi, pengujian dan sebagainya. (Sommerville Ian, 2007: 65).

Model pertama yang dipublikasikan dari proses pengembangan perangkat lunak diturunkan dari proses rekayasa sistem yang lebih umum

(Royce, 1970). Ini diilustrasikan pada gambar berikut :



Gambar 2. Model Waterfall

Karena cascade dari satu tahap ke tahap lainnya, model ini dikenal sebagai model waterfall atau software life cycle. Berikut penjelasan dari tahap-tahap yang dilakukan dalam model waterfall:

a. *Requirements Definiition*. Pelayanan, batasan, dan tujuan sistem ditentukan melalui konsultasi dengan user sistem. Persyaratan ini kemudian didefinisikan secara rinci dan berfungsi sebagai spesifikasi sistem.
b. *System and Software Design*. Proses perancangan sistem membagi persyaratan dalam sistem perangat keras atau perangkat lunak. Kegiatan ini menentukan arsitektur sistem secara keseluruhan. Perancangan perangkat lunak melibatkan identifikasi dan deskripsi abstraksi sistem perangkat lunak yang mendasar dan hubungan – hubungannya.
c. *Implementasi and Unit Testing*. Pada tahap ini, perancangan perangkat lunak direalisasikan sebagai serangkaian program atau unit program. Pengujian unit melibatkan verifikasi bahwa setiap unit telah memenuhi spesifikasinya.
d. *Integration System Testing*. Unit program atau program individual diintegrasikan dan diuji sebagai sistem yang lengkap untuk menjamin bahwa persyaratan sistem telah dipenuhi. Setelah pengujian sistem, perangkat lunak dikirim kepada pelanggan.
e. *Operation and Maintenance*. Biasanya (walaupun tidak seharusnya), ini merupakan tahapan life-cycle yang paling lama. Sistem diinstal dan dipakai. Pemeliharaan mencakup koreksi dan berbagai error yang tidak ditemukan pada tahap – tahap terdahulu, perbaikan atas implementasi unit sistem dan pengembangan pelayanansistem, sementara persyaratan – persyaratan baru ditambahkan.

2.7. *Object Oriented Analysis and Design (OOAD)*

Object Oriented Analysis and Design (OOAD) adalah pendekatan teknis yang digunakan dalam analisis dan perancangan aplikasi atau sistem melalui penerapan paradigma dan konsep berorientasi objek termasuk pemodelan visual. (Wikipedia).

OOAD memiliki cakupan analisa dan desain dalam perancangannya yang biasa disebut Object Oriented Analysis (OOA) dan Object Oriented Design (OOD).

2.7.1. Object Oriented Analysis (OOA)

Analisis adalah proses penggalian kebutuhan suatu sistem dan sistem apa yang harus dilakukan untuk memenuhi kebutuhan pengguna. Tujuan OOA adalah untuk memahami letak masalah dan tanggung jawab sistem dengan cara memahami bagaimana user menggunakan atau akan menggunakan sebuah sistem. Hal ini dilakukan dengan membangun beberapa model sistem. Model ini berkonsentrasi untuk menjelaskan apa yang akan dilakukan sistem dan bagaimana melakukannya.

Proses OOA berkaitan dengan langkah – langkah berikut :

a. Mengidentifikasi actor
b. Mengembangkan model proses bisnis sederhana menggunakan diagram aktivitas UML
c. Mengembangkan Use Case
d. Mengembangkan diagram interaksi
e. Mengidentifikasi class

Tahap analisis berorientasi objek berkaitan dengan penentuan persyaratan sistem dan identifikasi class dan hubungannya dengan class lain dalam domain masalah. (Ali Bahrami, 1999)

Analisa berorientasi objek (OOA) menekankan pembangunan model dunia nyata, dengan menggunakan pandangan berorientasi objek.

Analisa berorientasi objek (OOA) merupakan metode analisis yang menguji persyaratan dari prespektif kelas dan objek yang

ditemukan dalam kosakata dari masalah domain. (Grady Booch, Robert A. Maksimchuk, Michael W. Engle, Bobbi J. Young, Jim Conallen, Kelli A. houston: 42).

2.7.2. Object Oriented Design (OOD)

Tujuan desain berorientasi objek (OOD) adalah merancang class yang diidentifikasikan selama tahap analisis dan antarmuka dengan user. Selama fase ini, digunakan untuk mengidentifikasi dan menentukan objek tambahan serta class yang mendukung untuk penerapan persyaratan.

Desain berorientasi objek dan analisis berorientasi objek adalah disiplin ilmu yang berbeda namun bisa saling terkait. Pengembangan berorientasi objek sangat incremental dengan kata lain, memulai dengan analisis berorientasi objek, memodelkannya, membuat desain berorientasi objek, kemudian melakukan beberapa hal lagi, lagi dan lagi secara bertahap menyempurnakan dan melengkapi model sistem. (Ali Bahrami, 1999 : 47).

Proses OOD terdiri dari :

a. Design classes, attributes, methods, associations, structures, dan protocols.
b. Desain access layer dan prototype user interface.
c. Kepuasan pengguna dan uji kegunaan berdasarkan penggunaan.
d. Iterasi dan memperbaiki desain.

Selain itu dapat dijelaskan bahwa desain berorientasi objek (OOD) adalah metode perancangan yang mencakup proses dekomposisi object oriented dan notasi untuk menggambarkan model logis dan fisik serta statis dan dinamis dari sistem yang sedang didesain. (Grady Booch, Robert A. Maksimchuk, Michael W. Engle, Bobbi J. Young, Jim Conallen, Kelli A. houston: 42).

2.8. *Object Oriented Programming (OOP)*

Object Oriented Programming adalah metode implementasi dimana program diorganisir sebagai objek-objek yang kooperatif, yang masing-masing mewakili sebuah instance dari beberapa class, dan yang mana class adalah semua anggota hirarki class yang disatukan melalui hubungan warisan.

Ada tiga hal penting dalam definisi OOP :

1. Logika dasarnya tidak menggunakan algoritma melainkan menggunakan objek;
2. Setiap objek adalah turunan dari beberapa class;
3. Class saling terkait satu sama lain melalui hubungan warisan.

Sebuah program mungkin tampak berorientasi objek, namun jika salah satu elemen ini hilang, maka tidak bisa disebut program berorientasi objek.

2.9. *Unified Modeling Language (UML)*

Menurut Kim Hamilton dan Russel Miles, (2006), UML adalah bahasa pemodelan standar untuk perangkat lunak dan pengembangan sistem. Sebuah Model adalah abstraksi dari hal yang sebenarnya. Ketika melakukan pemodelan, abstrak yang dibuat akan jauh dari bagian yang tidak relevan atau mungkin berpotensi menjadi hal yang membingungkan. Model adalah penyederhanaan dari sistem yang sebenarnya, sehingga memungkinan untuk dapat memahami, mengevaluasi, dan mencari celah dari sebuah desain dan sistem lebih cepat daripada menelusuri sistem yang sebenarnya.

Bahasa pemodelan dapat berupa pseudo-code, actual code, gambar, diagram atau mungkin berupa tulisan berupa deskripsi panjang; intinya apa saja yang dapat membantu untuk mendeskripsikan sistem yang akan dibuat.

Untuk memodelkan suatu sistem secara efektif, maka diperlukan suatu bahasa yang dengannya model dapat dijelaskan, salah satunya UML.

Jenis-jenis diagram pada UML menurut Kim Hamilton dan Russel Miles antara lain tersedia pada Tabel berikut ini:

Tipe Diagram	**Keterangan (Apa yang bisa dimodelkan?)**
Use Case	Interaksi antara sistem dan pengguna atau sistem eksternal lainnya.
Activity	Aktifitas sekuensial dan sejajar dalam sistem. Memodelkan proses apa saja yang terjadi pada sistem.
Class	Kelas, jenis, antarmuka, dan hubungan diantaranya.
Object	Contoh objek dari kelasyang didefinisikan dalam diagram kelas dalam konfigurasi yang penting bagi sistem. Objek – objek pada suatu sistem dan hubungan

	diantaranya.
Sequence	Interaksi antar objek dimana urutan interaksi itu penting.
Communication	Cara – cara dimana objek berinteraksi dan koneksi yang dibutuhkan untuk interaksi itu.
Timing	Interaksi antar objek dimana *timing* menjadi perhatian penting.
Interaction Overview	Digunakan untuk mengumpulkan diagram *sequence, communication,* dan *timming* bersama untuk menangkap interaksi oenting yang terjadi di dalam sistem.
Composite Structure	Bagian dalam kelas atau komponen, dan dapat menggambarkan hubungan kelas dalam konteks tertentu.
Component	Komponen penting dalam sistem dan antarmuka yang digunakan untuk berinteraksi satu sama lain.
Package	Organisasi hirarki dari kelompok kelas dan komponen.
State Machine	Menggambarkan transisi maupun perubahan keadaan suatu objek pada sistem.
Deployment	Bagaimana sistem akhirnya dikerahkan dalam situasi dunia nyata.

III. OBJEK PENELITIAN DAN ANALISIS

3.1. Objek Penelitian

Pada penelitian ini dibutuhkan suatu kamus / data dengan jumlah yang cukup. Data kamus diperoleh dengan mengambil kumpulan kata yang telah disiapkan oleh kbbi.kemdikbud.go.id.

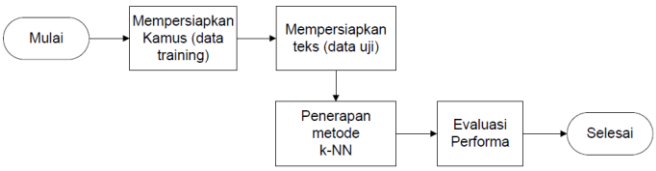
Aplikasi yang dibuat merupakan aplikasi berbasis web dengan menggunakan bahasa pemrograman *phyton*.

3.2. Analisis Sistem

Analisis sistem dilakukan untuk mempelajari algoritma pengklasifikasian metode *k-nearest neighbor* secara manual sehingga dapat diterapkan pada aplikasi yang akan dibuat.

3.2.1. Alur Proses Aplikasi Koresi Kata

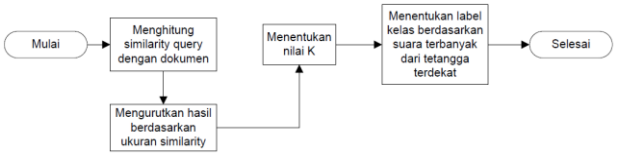
Proses aplikasi koreksi kata berbahasa Indonesia, melalui beberapa tahap seperti terlihat pada gambar yang menunjukkan alur prosesnya. Diasumsikan bahwa semua data uji (teks yang akan dikoreksi) telah melalui tahap preprocessing.



Gambar 3. Alur Proses Aplikasi Koreksi Kata

3.2.2. Penerapan Metode k-NN

Pada penerapan metode k-NN setiap kata pada teks tulisan akan dijadikan sebagai fitur / parameter penilaian. Namun untuk menentukan fitur diperlukan tahap untuk memisahkan satu kata dengan kata lainnya yang disebut proses preprocessing. Fitur kata pada teks yang diinput setiap kata direpresentasikan dalam bentuk metric dengan cara menentukan bobot dari setiap kata menggunakan leventhstein. Gambar 6 menjelaskan mengenai gambaran umum alur proses dari penerapan metode k-NN.



Gambar 4. Alur Proses Algoritma k-NN

Bagian yang diperhatikan dalam penerapan metode k-NN adalah sebagai berikut :

a. Analisis input
   Input yang dibutuhkan aplikasi adalah data training dan data uji dalam bentuk teks. Data input diperoleh dari kamus sebagai data training dan teks sebagai data uji. Untuk data training / kamus diasumsikan sudah tersimpan di dalam aplikasi. Dan untuk data uji / teks baru berupa kumpulan kata yang belum melalui tahap preprocessing.
b. Analisis data preprocessing
   Data preprocessing dibutuhkan dalam aplikasi ini untuk mengambil fitur / parameter. Maksudnya adalah memisahkan satu kata dengan kata lainnya. Data preprocessing yang akan diterapkan ke

dalam aplikasi adalah tokenizing yaitu mengambil term-term yang terdapat pada teks. Tokenizing merupakan aktivitas mengekstrak kata – kata dalam teks yang digunakan sebagai fitur/parameter dalam aplikasi koreksi kata. Pada tahap ini, tanda baca pada teks tidak akan diambil. Perubahan sebuah teks melalui tahap tokenizing seperti terlihat pada contoh.

Contoh teks:

Komputer adalah alat yang dipakai untuk mengolah data menurut prosedur yang telah dirumuskan. Kata computer pada awalnya dipergunakan untuk menggambarkan orang yang pekerjaannya menlakukan perhitungan aritmetika, dengan atau tanpa alat bantu, tetapi arti kata ini kemudian dipindahkan kepada mesin itu sendiri.

Setelah proses tokenizing :

[‘Komputer’, ‘adalah’, ‘alat’, ‘yang’, ‘dipakai untuk’, ‘mengolah’, ‘data’, ‘menurut’, ‘prosedur’, ‘yang’, ‘telah’, ‘dirumuskan’, ‘Kata’, ‘computer’, ‘pada’, ‘awalnya’, ‘dipergunakan’, ‘untuk’, ‘menggambarkan’, ‘orang’, ‘yang’, ‘pekerjaannya’, ‘melakukan’, ‘perhitungan’, ‘aritmetika, dengan’, ‘atau’, ‘tanpa’, ‘alat’, ‘bantu’, ‘tetapi’, ‘arti’, ‘kata’, ‘ini’, ‘kemudian’, ‘dipindahkan’, ‘kepada’, ‘mesin’, ‘itu’, ‘sendiri’]

c. Analisis tahap kategorisasi

Setelah proses preprocessing selesai berikutnya adalah proses kategorisasi, dimana dalam proses ini adalah mengklasifikasikan kata ke dalam kategori match atau mismatch menggunakan levenshtein distance.

Algoritma levenshtein distance yaitu algoritma yang mengukur kesamaan antara 2 string, nantinya akan lebih dikenal dengan string source/sumber (s) dengan string target (t), contoh :

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

Algoritma *levenshtein* bekerja sebagai berikut :

1. Langkah pertama
   - Set variable N yang menyimpan panjang string source (s)
   - Set variable M yang menyimpan panjang string target (t)
   - Jika N = 0 atau M = 0 maka exit
   - Buat matriks ber-ordo [0 – N][0 – M]
2. Langkah kedua
   - Inisialisasi baris 0 – N
   - Inisialisasi kolom 0 – M
3. Langkah ketiga
   - Periksa setiap karakter dari string s (looping dari 1 ke N → variable i)
4. Langkah keempat
   - Periksa setiap karakter dari string t (looping dari 1 ke M → variable j)
5. Langkah kelima
   - Jika s[i] = t[i] => cost = 0
   - Jika s[i] != t[i] => cost = 1
6. Langkah keenam
   - Set value D[i,j] yang diambil dari minimum jumlah :
     - D[i-1,j] + 1
     - D[i,j-1] + 1
     - D[i-1,j-1] + cost
7. Langkah ketujuh
   - Setelah langkah 3, 4, 5, 6 selesai (tidak ada looping lagi), hasilnya akan ketemu di elemen D[N,M]

Contoh kasus untuk mengimplementasikan algoritma di atas adalah :

Jika ada 2 buah kata x = BARU dengan y = BATU, maka ekspektasi dari outputnya adalah 1 dimana 1 merupakan perbedaan huruf dari kedua kata tersebut.

Langkah 1 & 2

		B	A	R	U
	0	1	2	3	4
B	1				
A	2				
T	3				
U	4				

Langkah 3 – 6, 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

		B	A	R	U
	0	1	2	3	4
B	1	0			
A	2				
T	3				
U	4				

Lakukan langkah tersebut sampai akhirnya mendapatkan hasil :

		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 bisa kelihatan perbedaan antara kedua kata :

X : BARU
Y : BATU
Adalah 1 huruf.

3.2.3. Analisis Kebutuhan Data

Untuk mengembangkan aplikasi pada penelitian ini maka dibutuhkan data untuk menjalankan algoritma metode klasifikasi k-NN. Berikut data yang diperlukan pada aplikasi yang dibuat:

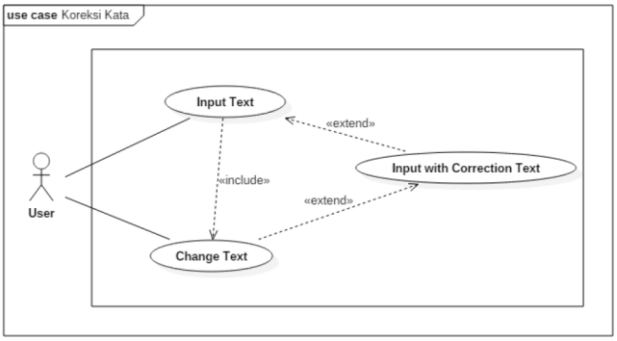
1. Data training
   Data training yaitu kamus yang diperlukan untuk proses menjalankan algoritma klasifikasi k-NN sebagai kata pembanding.
2. Data uji
   Data uji yaitu teks yang diperlukan untuk proses menjalankan algoritma klasifikasi k-NN. Dari tulisan ini akan diekstrak term/kata yang muncul pada teks menggunakan data preprocessing.
3. Term / fitur
   Term yang muncul pada teks berupa kata. Kata-kata yang muncul pada teks akan menentukan proses pengkategorian dengan menggunakan algoritma k-NN.
4. Kategori / nilai
   Kategori / nilai merupakan label yang diberikan untuk kata pada teks sesuai dengan ketentuan penilaian terhadap kata di dalam kamus. Nilai disini dapat menentukan sebuah kata benar atau salah.

IV. PERANCANGAN SISTEM

Pada tahap ini dilakukan perancangan aplikasi berdasarkan requirement yang dihasilkan. Perancangan yang dilakukan meliputi perancangan perilaku sistem, perancangan kelas, perancangan interaksi antar objek, dan perancangan tampilan antarmuka/ user interface. Berikut penjelasan masin-masing perancangan tersebut.

4.1. Perancangan Perilaku Sistem

Perancangan perilaku sistem digambarkan menggunakan diagram use case. Selanjutnya untuk memperjelas interaksi antara user dengan sistem dibuat use case specification yang berisi deskripsi yang menjelaskan setiap interaksi pada diagram use case. Berikut use case diagram dari aplikasi yang dibuat seperti ditunjukkan pada gambar berikut:



Gambar 5. Use Case Koreksi Kata

Berikut skenario use case koreksi kata tersebut:

Tabel 1. Skenario Use Case Koreksi Kata

Use Case ID	UC.1
Use Case Name	Input Text
No. Req	REQ. 1, REQ. 2, REQ. 3, REQ. 4
Primary Actor	User
Stakeholder and Interest	User memasukkan teks yang akan dikoreksi.
Precondition	Telah terdapat kamus dalam aplikasi (data training).
Postcondition	Sistem menampilkan hasil koreksi.
Main Scenario	
Actor	*System*
1. User menginput text di textarea “Masukkan Teks” yang telah disediakan.	2. Sistem menampilkan hasil koreksi di textarea “Hasil Koreksi” bahwa yang diinput user sudah sesuai dengan kamus, ditandai dengan tidak adanya perubahan warna pada text.
Alternative Scenario	
Actor	*System*
1. User menginput text di textarea “Masukkan Teks” yang telah disediakan.	2. Sistem menampilkan hasil koreksi di textarea “Hasil Koreksi” bahwa terdapat kata yang tidak sesuai dengan kamus, ditandai dengan berubahnya warna pada kata yang salah menjadi warna biru.

Tabel 2. Skenario Use Case Change Text

Use Case ID	UC.2
Use Case Name	Change Text
No. Req	REQ. 1, REQ. 2, REQ. 3, REQ. 4, REQ. 5
Primary Actor	User
Stakeholder and Interest	User merubah text yang telah dikoreksi sistem.
Precondition	Telah terdapat kamus dalam aplikasi (data training).
Postcondition	User menetapkan hasil koreksi
Main Scenario	
Actor	*System*
1. User menginput text di textarea “Masukkan Teks” yang telah disediakan 3. User mengklik kata yang telah dikoreksi oleh sistem di textarea “Hasil Koreksi” yang berwarna biru. 5. User memilih kata berwarna biru yang dianggap tepat dari listarea untuk menggantikan kata yang salah.	2. Sistem menampilkan hasil koreksi di textarea “Hasil Koreksi” bahwa terdapat kata yang tidak sesuai dengan kamus, ditandai dengan berubahnya warna pada kata yang salah menjadi warna biru. 4. Di listarea sistem menampilkan kata asal yang diklik user (ditandai dengan warna merah) dan kumpulan kata berwarna biru yang mendekati kata yang diklik oleh user. 6. Sistem merubah kata pada textarea “Hasil Koreksi” yang dipilih User sesuai dengan kata pilihan User di listarea.
Alternative Scenario	
Actor	*System*
1. User menginput text di textarea “Masukkan Teks” yang telah disediakan.	2. Sistem menampilkan hasil koreksi di textarea “Hasil Koreksi” bahwa yang diinput user sudah sesuai dengan kamus, ditandai dengan tidak adanya perubahan warna pada text.

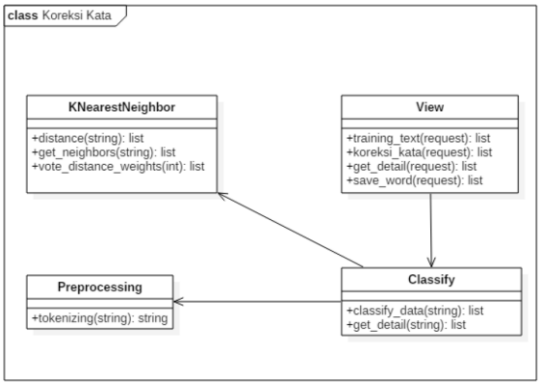
Tabel 3. Skenario Use Case Input With Corection Text

Use Case ID	UC.3
Use Case Name	Input with Correction Text
No. Req	REQ. 1, REQ. 2, REQ. 3, REQ. 4, REQ. 5, REQ.6
Primary Actor	User
Stakeholder and Interest	User memasukan kata yang telah dikoreksi.
Precondition	Sistem telah menampilkan hasil koreksi dengan adanya kata yang tidak sesuai kamus (ditandai perubahan warna pada kata menjadi biru)
Postcondition	Sistem menyimpan kata

	berwarna merah menjadi kata yang tidak perlu dikoreksi lagi.
Main Scenario	
Actor	*System*
1. User mengklik kata yang telah dikoreksi oleh sistem di textarea "Hasil Koreksi" yang berwarna biru. 3. User meng-klik kata berwarna merah untuk menyimpan kata tersebut dan agar kata tersebut tidak dikoreksi sistem saat akan input teks lagi.	2. Di listarea sistem menampilkan kata asal yang diklik user (ditandai dengan warna merah) dan kumpulan kata berwarna biru yang mendekati kata yang diklik oleh user. 4. Sistem merubah kata di listarea yang asalnya berwarna merah menjadi warna biru yang artinya sistem telah menyimpan kata tersebut agar tidak dikoreksi. 5. Sistem merubah kata pada textarea "Hasil Koreksi" yang dipilih User sesuai dengan kata yang pada listarea yang asalnya berwarna merah.
Alternative Scenario	
Actor	*System*
1. User mengklik kata yang telah dikoreksi oleh sistem di textarea "Hasil Koreksi" yang berwarna biru. 3. User memilih kata berwarna biru yang dianggap tepat dari listarea untuk menggantikan kata yang salah.	2. Di listarea sistem menampilkan kata asal yang diklik user (ditandai dengan warna merah) dan kumpulan kata berwarna biru yang mendekati kata yang diklik oleh user. 4. Sistem merubah kata pada textarea "Hasil Koreksi" yang dipilih User sesuai dengan kata pilihan User di listarea.

4.2. Perancangan Class Diagram

Untuk menggambarkan arsitektur aplikasi yang dibuat, digunakan Class Diagram. Seperti gambar berikut:

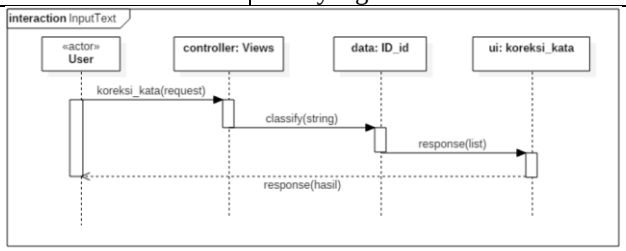


Gambar 6. Arsitektur Aplikasi Koreksi Kata

4.3. Perancangan Interaksi Antar Objek

Tabel 4. Interaksi Antar Objek: Input Text

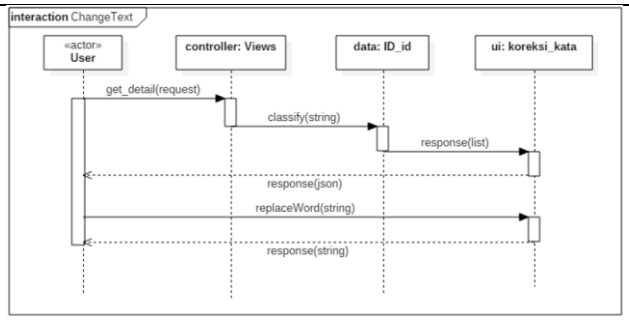
Secuence diagram ID	SD. 1
Secuence diagram name	Input Text
Use Case ID	UC. 1
No. Req	REQ. 1, REQ. 2, REQ. 3, REQ. 4
Nama kelas	*Contoller: Views* *View: koreksi_kata* *Model: ID_id*
Deskripsi	Proses input text yang menghasilkan text tanpa adanya kata yang dikoreksi



Gambar 7. Sequence Diagram Input Text

Tabel 5. Interaksi Antar Objek: Change Text

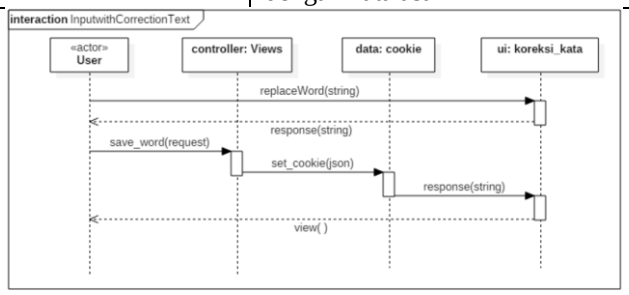
Secuence diagram ID	SD. 2
Secuence diagram name	*Change Text*
Use Case ID	UC. 2
No. Req	REQ. 1, REQ. 2, REQ. 3, REQ. 4, REQ. 5
Nama kelas	*Contoller: Views* *View: koreksi_kata* *Model: ID_id*
Deskripsi	Proses mengubah hasil koreksi sistem dengan keinginan user



Gambar 8. Sequence Diagram Change Text

Tabel 6. Interaksi Antar Objek: Input with Correction Text

Secuence diagram ID	SD. 3
Secuence diagram name	*Input with Correction Text*
Use Case ID	UC. 3
No. Req	REQ. 1, REQ. 2, REQ. 3, REQ. 4, REQ. 5, REQ.6
Nama kelas	*Contoller: Views* *View: koreksi_kata* *Model: cookie*
Deskripsi	Proses menyimpan kata ke dalam cookie dan mengkoreksi kata dengan kata asal



Gambar 9. Sequence Diagram Input with Correction Text

4.4. Perancangan User Interface

Berikut rancangan user interface pada aplikasi Koreksi Kata disajikan pada tabel berikut:

Tabel 7. Perancangan Antarmuka

No UI	UI.1
Nama UI	Koreksi kata
Deskripsi UI	Tampilan dimana user memasukkan tulisan untuk dikoreksi

Gambar 10. User Interface Koreksi Kata

Keterangan	1. Setelah user memasukan tulisan ke dalam textarea Masukkan Kata dan menekan tombol koreksi maka tulisan kata yang salah akan muncul, dan hasil koreksi kata akan muncul secara otomatis di textarea Hasil Koreksi. 2. User bebas memilih kata yang diinginkan dari listarea kumpulan kata yang mendekati kata yang salah. 3. Jika ternyata kata yang dianggap salah oleh sistem merupakan kata yang benar namun terdeteksi salah, kemungkinan kata tersebut belum ada di kamus, dan user bisa menyimpan kata tersebut dengan cara klik kata yang dimaksud lalu klik simpan. 4. Jika kumpulan kata sudah dirasa benar maka user bisa langsung klik salin untuk mengambil teks tersebut

V. IMPLEMENTASI DAN PENGUJIAN

Aplikasi ini dapat dijalankan pada berbagai platform sistem operasi dan perangkat keras, namun implementasi dan pengujian sepenuhnya hanya dilakukan pada perangkat keras PC/Laptop dengan sistem operasi Windows.

4.1. Kebutuhan Perangkat Lunak

Dalam menerapan rancangan yang telah dibuat, dibutuhkan beberapa software untuk membuat Aplikasi Koreksi Kata Berbahasa Indonesia dengan Metode k-NN Berbasis Web, yaitu :

Tabel 8. Perangkat Lunak Pendukung

Perangkat Lunak(*Software*)	**Keterangan**
Windows 10	Sebagai sistem operasi.
Google Chrome	Sebagai *browser* atau sebagai alat untuk mengakses aplikasi, karena aplikasi yang di bangun berbasiskan *web*.
Eclipse	Sebagai IDE (*Integrated Development Environment*) untuk penulisan kode aplikasi.
Java version 1.4.0 – 5.0	Untuk menunjang berjalannya Eclipse IDE.

4.2. Kebutuhan Perangkat Lunak

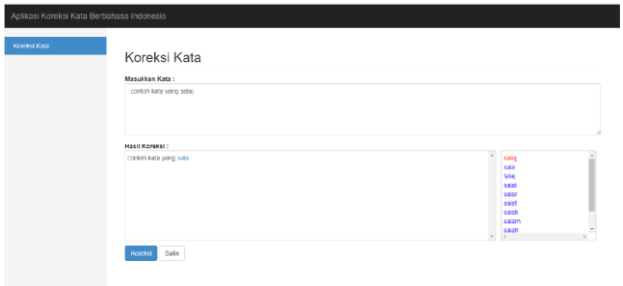
Dalam penerapan Aplikasi Koreksi Kata Berbahasa Indonesia dengan Metode k-NN Berbasis Web, dibutuhkan beberapa perangkat keras untuk menyajikan aplikasi ini. Adapun yang dibutuhkan adalah :

Tabel 9. Perangkat Keras Pendukung

Perangkat Keras	**Minimal**	**Direkomendasikan**
Processor	800 Mhz	1.5 Ghz or faster
RAM	512 MB	2 GB or more
Memory	512 MB	1 GB or more
Hard Disk	300 MB	1 GB or more
Display	1024 x 768 16bit	1366 x 768 16bit

4.3. Tampilan Aplikasi Koreksi Kata

Implementasi halaman koreksi kata dibuat bentuk file program dengan berekstensi html yang terkoneksi dengan file berekstensi py.



Gambar 11. Tampilan Aplikasi Koreksi Kata

4.4. Pengujian Sistem

Pengujian bermaksud untuk mengetahui perangkat lunak yang dibuat sudah memenuhi kriteria yang sesuai dengan tujuan perancangan perangkat lunak tersebut.

Pengujian yang dilakukan yaitu pengujian Black box dan White box. Pengujian black box digunakan untuk menguji fungsi – fungsi khusus dari perangkat lunak yang dirancang. Pengujian white box digunakan untuk meyakinkan semua perintah dan kondisi dieksekusi secara minimal.

4.4.1. Pengujian Black Box

Berdasarkan rencana pengujian, maka dapat dilakukan pengujian Black Box pada Aplikasi Koreksi Kata Berbahasa Indonesia dengan Metode k-NN Berbasis Web sebagai berikut :

Tabel 10. Pengujian Input Text

Identifikasi	Pengujian dan Hasil Uji Aplikasi (1)		
Nama Butir Uji	Input Text		
Tujuan	Menampilkan hasil koreksi teks		
Kondisi Awal	Sudah terdapat kamus di aplikasi		
Skenario			
1. Input teks yang akan dikoresi ke textarea **Masukkan Kata**			
2. Klik tombol koreksi			
Hasil			
Data Masukkan	**Yang Diharapkan**	**Pengamatan**	**Kesimpulan**
Teks yang akan dikoreksi	Sistem menampilkan hasil koreksi dengan membedakan kata yang tidak ada di dalam kamus dengan perubahan warna pada kata menjadi	Sistem telah menampilkan hasil koreksi dengan kata berwarna biru untuk kata yang tidak ada di dalam kamus	OK

	warna biru		

Tabel 11. Pengujian Change Text

Identifikasi			Pengujian dan Hasil Uji Aplikasi (2)
Nama Butir Uji			Change Text
Tujuan			Mengubah kata sudah dikoreksi sistem dengan kata yang dianggap lebih sesuai
Kondisi Awal			Sistem sudah menampilkan hasil koreksi dengan kata berwarna biru sebagai kata yang tidak ada di kamus
Skenario			
1. User klik kata berwarna biru di text area **Hasil Koreksi** yang dianggap kurang sesuai dengan keinginan User			
2. Sistem menampilkan asal kata yang diklik oleh User dengan warna merah dan menampilkan beberapa kata yang mendekati kata yang aan dikoreksi dengan warna biru di kolom kecil			
Hasil			
Data Masukkan	**Yang Diharapkan**	**Pengamatan**	**Kesimpulan**
Memilih satu kata di textarea **Hasil koreksi**	Dapat menampilkan asal kata yang diklik oleh user dengan warna merah dan menampilkan beberapa kata yang mendekati kata yang akan dikoreksi dengan warna biru di list area	Telah menampilkan asal kata yang diklik oleh user dengan warna merah dan menampilkan beberapa kata yang mendekati kata yang akan dikoreksi dengan warna biru di list area	OK
Memilih satu kata di list area	Dapat mengubah kata di textarea **Hasil koreksi** dengan kata berwarna biru yang dipilih oleh user yang ada di list area	Kata di textarea **Hasil koreksi** diganti dengan kata yang dipilih oleh user yang ada di list area	OK

Tabel 12. Pengujian Input with Corecction Text

Identifikasi			Pengujian dan Hasil Uji Aplikasi (3)
Nama Butir Uji			Input with correction text
Tujuan			Menyimpan kata yang belum ada di dalam kamus ke dalam cookie
Kondisi Awal			Sistem telah menampilkan asal kata yang di klik oleh user dedngan warna merah dan menampilkan beberapa kata yang mendekati kata yang akan dikoreksi dengan warna biru di list area
Skenario			
1. User klik kata berwarna merah di listarea			
2. Sistem menampilkan konfirmasi penyimpanan kata			
3. Klik tombol simpan			
Hasil			
Data Masukka	**Yang Diharapka**	**Pengamatan**	**Kesimpulan**

n	**n**		
Klik kata berwarna merah di listarea	Dapat menampilkan konfirmasi "Apakah anda yakin akan menyimpan kata?"	Menampilkan konfirmasi "Apakah anda yakin akan menyimpan kata?"	OK
	Dapat menyimpan kata di cookie	Kata tersimpan di cookie	OK
	Dapat mengubah kata berwarna merah jadi warna biru di listarea	Mengubah kata berwarna merah jadi warna biru di listarea	OK
	Dapat mengubah kata yang ditunjuk di textarea **Hasil koreksi** dengan kata yang baru tersimpan	Mengubah kata yang ditunjuk di textarea **Hasil koreksi** dengan kata yang baru tersimpan	OK

4.4.2. Pengujian White Box

Pengujian White Box pada Aplikasi Koreksi Kata Berbahasa Indonesia dengan Metode k-NN Berbasis Web sebagai berikut :

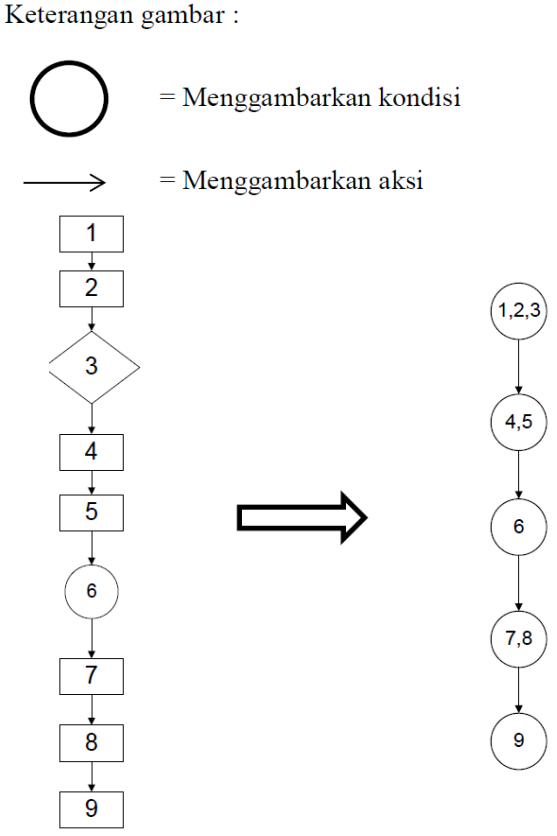
4.4.2.1. Pengujian Algoritma k-Nearest Neighbor

Pengubahan code Algoritma k-Nearest Neighbor menjadi flowchart kemudian menjadi flowgraph.

Berikut code algoritma k-Nearest Neighbor yang digunakan :

```
1. def get_neighbors(self, training_set,
     labels,
     test_instance,
     k,
     distance=distance):
2. distances = [ ]
3. for index in range(len(training_set)):
4. dist= distance(test_instance,
training_set[index])
5. distances.append((training_set[index],dist,
labels[index]))
6. end
7. distances.sort(key=lambda x: x[1])
8. neighbors = distances[:k]
9. return neighbors
```

a. Pengubahan flow chart menjadi flow graph algoritma k-Nearest neighbor yang digunakan dapat dilihat pada gambar berikut:



Gambar 12. Flow Chart menjadi Flow Graph

Dari gambar tersebut dapat dihitung cluclomatic complexity sebagai berikut :

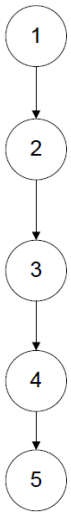
V(G) = E – N + 2
V(G) = 4 – 5 + 2
V(G) = (-1) + 2
V(G) = 1
Di mana :

E = jumlah edge pada grafik alir
N = jumlah node pada grafik alir

Jadi, cyclomatic complexity untuk gambar 13 adalah 1. Berdasarkan cylclomatic complexity tersebut, maka terdapat 1 path yaitu :

Path 1	1, 2, 3, 4, 5, 6, 7, 8, 9

b. Penomoran ulang flow graph



Gambar 13. Penomoran ulang Flow Graph algoritma k-NN

c. Graph matrix

N	1	2	3	4	5	
1		1				0
2			1			0
3				1		0
4					1	0
5						0
Jumlah + 1						1

Gambar 14. Graph Matriks Algoritma k-NN

d. Pengujian path algoritma k-NN

Tabel 13. Pengujian path algoritma k-NN

No	Kasus Uji	Hasil Yang Diharapkan	Hasil Uji Kasus	Keterangan

1	Memberikan nilai K= 3, training_set=[aku, akua, akun, kamu, dia, siapa, mereka, adalah, alibi, aneh, ampul], labels=[aku, akua, akun, kamu, dia, siapa, mereka, adalah, alibi, aneh, ampul], test_instance=”akuuu”, distance=levenshtein, self=yang ditentukan oleh python
	`distances = []`
	`for index in range(len(training_set)):`
	`dist = distance(test_instance, training_set[index])`
	`distances.append((training_set[index], dist, labels[index]))`
	`distances.sort(key=lambda x: x[1])`
	`neighbors = distances[:k]`
	`return neighbors`
		Memasukkan nilai K = 3 yang artinya mengambil kata dari kamus yang mendekati test_instance = “akuuu”	Menghasilkan 3 kata yang mendekati kata “akuuu” yaitu : 1. aku 2. akua 3. akun	OK

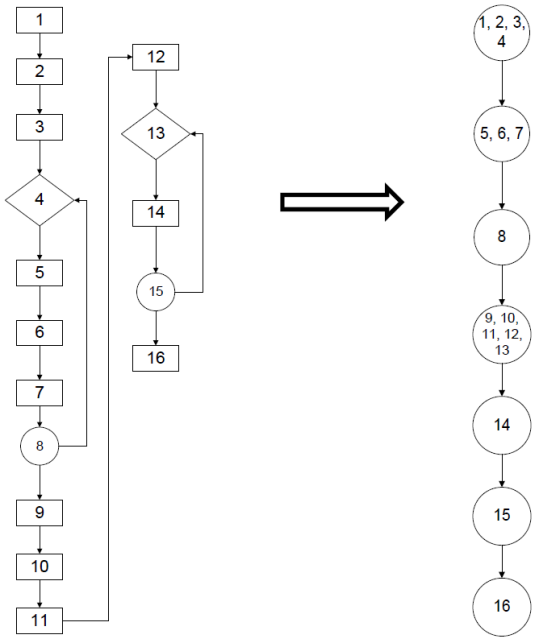
4.4.2.2. Pengujian Algoritma Vote Distance Weights

a. Pengubahan code Algoritma Vote Distance Weights menjadi flowchart kemudian menjadi flowgraph. Berikut code algoritma Vote Distance Weights yang digunakan :

```
1. def vote_distance_weights(self, neighbors, all_results=True):
2. class_counter = Counter()
3. number_of_neighbors = len(neighbors)
4. for index in range(number_of_neighbors):
5. dist = neighbors[index][1]
6. label = neighbors[index][2]
7. class_counter[label] += 1 / (dist**2 + 1)
8. end
9. labels, votes = zip(*class_counter.most_common())
10. winner = class_counter.most_common(1)[0][0]
11. votes4winner = class_counter.most_common(1)[0][1]
```

```
12. total = sum(class_counter.values(), 0.0)
13. for key in class_counter:
14. class_counter[key] /= total
15. end
16. return winner, class_counter.most_common()
```

b. Pengubahan flow chart menjadi flow graph algoritma Vote Distance Weights yang digunakan dapat dilihat pada gambar berikut:

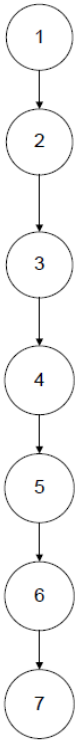


Gambar 15. Flow Chart menjadi Flow Graph

Dari gambar tersebut diatas dapat dihitung *cluclomatic complexity* sebagai berikut :
V(G) = E – N + 2
V(G) = 6 – 7 + 2
V(G) = (-1) + 2
V(G) = 1
Di mana :
E = jumlah edge pada grafik alir
N = jumlah node pada grafik alir
Jadi, cyclomatic complexity untuk gambar 16 adalah 2. Berdasarkan cylclomatic complexity tersebut, maka terdapat 2 path yaitu :

Path 1	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16

c. Penomoran ulang flow graph



Gambar 16. Penomoran ulang Flow Graph algoritma

d. Graph matrix

N	1	2	3	4	5	6	7	
1		1						0
2			1					0
3				1				0
4					1			0
5						1		0
6							1	0
7								0
Jumlah + 1								1

Gambar 17. Graph Matriks Algoritma Vote Distance Weights

e. Pengujian path algoritma Vote Distance Weights

Tabel 14. Pengujian path algoritma Vote Distance Weights

No	Kasus Uji	Hasil Yang Diharapkan	Hasil Uji Kasus	Keterangan

1	Memberikan nilai self=yang ditentukan oleh python, neighbor= [‘aku’, ‘akua’, ‘akun’]
	`def vote_distance_weights(self, neighbors, all_results=True):`
	`class_counter= Counter()`
	`number_of_neighbors = len(neighbors)`
	`for index in range(number_of_neighbors):`
	`dist=neighbors[index][1]`
	`label=neighbors[index][2]`
	`class_counter[label] += 1 / (dist**2 + 1)`
	`end`
	`labels,votes=zip(*class_counter.most_common())`
	`winner=class_counter.most_common(1)[0][0]`
	`votes4winner=class_counter.most_common(1)[0][1]`
	`total=sum(class_counter.values(), 0.0)`
	`for key in class_counter:`
	`class_counter[key] /= total`
	`end`
	`return winner, class_counter.most_common()`
		Memasukkan nilai neighbor = [‘aku’, ‘akua’, ‘akun’] yang didapat dari proses algoritma *k-NN*, lalu aplikasi menampilkan salah satu kata dari 3 kata tersebut yang paling mendekati kata “akuuu”	Menghasilkan 1 kata yang paling mendekati kata “akuuu” yaitu “aku”	OK

VI. KESIMPULAN DAN SARAN

5.1. Kesimpulan

Secara umum Aplikasi koreksi kata sudah mampu memenuhi kebutuhan seperti yang tercantum pada identifikasi masalah. Berdasarkan hasil yang didapat maka dapat ditarik kesimpulan sebagai berikut :

1. Aplikasi koreksi kata ini dapat berjalan sesuai dengan tujuannya yaitu mengkoreksi kata berbahasa Indonesia berbasis web.
2. Metode k-NN telah diterapkan pada Aplikasi Koreksi Kata dengan perhitungan jarak menggunakan levenshtein distance.

5.2. Saran

Kamus pada aplikasi ini masih hanya meliputi kata – kata yang umum digunakan. Oleh karena itu saran untuk pengembangan Aplikasi Koreksi Kata kedepannya agar lebih baik lagi adalah menambah kamus dan fasilitas yang belum tersedia pada Aplikasi, karena ada beberapa fasilitas yang seharusnya diterapkan di dalam Aplikasi tersebut seperti :

1. Penambahan kamus nama orang Indonesia.
2. Penambahan kamus organisasi.
3. Penambahan kamus akronim (singkatan).
4. Fasilitas susunan pada teks yang tidak berubah seperti font tulisan.

VII. DAFTAR PUSTAKA

[1] Adiwidya, B. M. D. 2009. Algorima levenshtein dalam pendekatan approximate string matching. Skripsi. Institut Teknologi Bandung.

[2] Andika, F. R. 2010. Penerapan string suggestion dengan algoritma levenshtein distance dan alternatif algoritma lain dalam aplikasi. Skripsi. Institut Teknologi Bandung.

[3] Andri, Sunda Ariana, Margareta Andriani. 2014. Aplikasi Koreksi Kesalahan Berbasis Pada Tulisan Berbahasa Indonesia Untuk Meningkatkan Kualitas Penulisan Karya Ilmiah. Yogyakarta.

[4] Badanbahasa.kemendikbud.go.id. Sekilas Tentang Sejarah Bahasa Indonesia [online]. Available http://badanbahasa.kemdikbud.go.id/lamanbahasa/petunjuk_praktis/627/Sekilas%20Tentang%20Sejarah%20Bahasa%20Indonesia

[5] Bahrami, Ali. 1999. Object Oriented System Development, Singapore: Irwin – McGraw-Hill.

[6] Booch, Grady., A. Maksimchuk, Robbert., W. Engle, Michael., J. Young Ph.D, Bobbi., Conallen Jim. and A. Houston, Kelli. 2007. Object-Oriented Analysis And Design With Applications – Third Edition, California: Addison-Wesley.

[7] Booch, Grady. 1998. Object Oriented Analysis And Design – Second Edition, California: Addison-Wesley.

[8] Cunningham, P. and Delany, S. 2007. K-Nearest Neighbor Classifier.

[9] Dhanta, R. 2009. Pengantar Ilmu Kompter, Surabaya: Indah.

[10] Hamilton, Kim., Miles Russ. 2006. Learning UML 2.0, United State of America: O’Reilly.

[11] Husain, S. W. 2013. Penerapan algoritma binary search dan metode approximate string matching pada aplikasi kamus bahasa Indonesia – Mongondow berbasis mobile. Skripsi. Universitas Negeri Gorontalo.

[12] Janowski, T. and Mohanti, H. 2010. Distributed Computing and Internet Technology. India: Springer.

[13] Kamusilmiah.com. Sejarah World Wide Web [online]. Available https://www.kamusilmiah.com/it/sejarah-world-wide-web

[14] Mathworks.com. Unsupervised Learning – MATLAB [online]. Available https://www.mathworks.com/discovery/unsupervised-learning.html
[15] Mohri, Mehryar, Afshin Rostamizadeh, and Ameet Talwalkar. 2012. Foundations of Machine Learning. The MIT Press.
[16] Pryana, Dewi Rokhmah, Suprapto, Suharsono, Aswin. 2013. Program Aplikasi Editor Kata Bahasa Indonesia Menggunakan Metode Approximate String Matching Dengan Algoritma Levenshtein Distance Berbasis Java, Malang : Universitas Brawijaya.
[17] Satheesh, Ajita., dan Ravindra Patel. 2010. Dynamic Nearest Neighbours Classifier For Integrated Data Using Object Oriented Concept Generalization. International Journal of Simulation, Science and Technology, vol. 11, 35-40.
[18] Simon, P. 2013. To big to ignore the business case for big data, Hoboken, NJ: Wiley.
[19] Sommerville, Ian. 2007. Software Engineering – Eight Edition, Addison Wesley : Massachussets.
[20] Tim E-Media Solusindo, Membengun Komunitas Online, Ibid.
[21] en.wikipedia.org. Levenshtein Distance [online]. Available https://en.wikipedia.org/wiki/Levenshtein_distance
[22] en.wikipedia.org. Object – oriented analysis and design [online]. Available https://en.wikipedia.org/wiki/Object-oriented_analysis_and_design
[23] id.wikipedia.org. Aplikasi [online]. Available https://id.wikipedia.org/wiki/Aplikasi
[24] Zhu, Xiaojin and Andrew B. Goldberg. 2009. Introduction to Semi-Supervised Learning Synthesis Lectures on Artifical Intelligence and Machine Learning. University of Wisconsin.