TUGAS AKHIR

IMPLEMENTASI ALGORITMA DAMERAU-LEVENSHTEIN DISTANCE UNTUK MEMPERBAIKI KESALAHAN PENULISAN *KEYWORD* PENCARIAN

OLEH :

ARIANDO

DBC 113 087

PROGRAM STUDI/JURUSAN TEKNIK INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS PALANGKA RAYA

2018

TUGAS AKHIR

IMPLEMENTASI ALGORITMA DAMERAU-LEVENSHTEIN DISTANCE UNTUK MEMPERBAIKI KESALAHAN PENULISAN KEYWORD PENCARIAN

OLEH :

ARIANDO

DBC 113 087

PROGRAM STUDI/JURUSAN TEKNIK INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS PALANGKA RAYA

2018

“IMPLEMENTASI ALGORITMA DAMERAU-LEVENSHTEIN DISTANCE UNTUK MEMPERBAIKI KESALAHAN PENULISAN KEYWORD PENCARIAN”

TUGAS AKHIR

Sebagai salah satu syarat untuk menyelesaikan Program Strata-1
pada Jurusan Teknik Informatika Fakultas Teknik Universitas Palangka Raya

Oleh:

ARIANDO
DBC 113 087

Mengetahui,

Dosen Pembimbing I,	Dosen Pembimbing II,
Enny D. Oktaviyani, ST.,M,Kom.	**Sherly Christina, S.Kom.,M,Kom.**
NIP. 19811003 200604 2 001	**NIP. 19810929 200604 2 001**

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS PALANGKA RAYA
2018

PERNYATAAN

Dengan ini saya menyatakan dengan sebenar – benarnya bahwa dalam Tugas Akhir ini tidak terdapat karya ilmiah yang pernah diajukan untuk memperoleh gelar sarjana di suatu perguruan tinggi, serta tidak terdapat karya ilmiah atau pendapat yang ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis diacu dalam Tugas Akhir ini dan disebutkan dalam daftar pustaka.

Palangkaraya, 12 Maret 2018

ARIANDO
NIM. DBC 113 087

RIWAYAT PENYUSUN

Data Diri

Nama	:	ARIANDO
NIM	:	DBC 113 087
Tempat, Tanggal Lahir	:	Anjir Kalampan, 22 April 1996
Status	:	Belum Menikah
Agama	:	Kristen
Pekerjaan	:	Mahasiswa
Alamat	:	Jl. Temanggung Kanyapi 3, Gg. Kartika Asri E No. 3, Palangka Raya
No. Telepon/HP	:	0812 5575 8606

Data Orang Tua

Nama Ayah	:	MARTHIN
Pekerjaan Ayah	:	PNS
Nama Ibu	:	ASKAYUN
Pekerjaan Ibu	:	Ibu Rumah Tangga
Alamat Orang Tua	:	Jl. Pangkalima Kapang Km.6, Anjir Kalampan, Kapuas Barat.

Riwayat Pendidikan *)

SD	:	SD Negeri 2 Anjir Kalampan	(Tahun Lulus 2007)
SMP	:	SMP Negeri 3 Anjir Kalampan	(Tahun Lulus 2010)
SMA	:	SMA Negeri 1 Kapuas Barat	(Tahun Lulus 2013)

Palangka Raya, 12 Maret 2018

ARIANDO
NIM. DBC 113 087

Keterangan:
*) Nama, Tempat, Tahun Lulus

HALAMAN PERSEMBAHAN

"Diberkatilah orang yang mengandalkan YAHWEH*, yang menaruh harapannya pada* YAHWEH*!"* (**Yeremia 17:7**)

Puji dan syukur saya panjatkan ke hadirat Tuhan Yesus Kristus, karena berkat limpahan Rahmat dan Karunia-nya sehingga saya dapat menyelesaikan tugas akhir ini dengan baik, serta tepat pada waktunya.

Laporan ini saya persembahkan dengan tulus dan terima kasih kepada:

1. Ayahanda dan Ibunda tercinta, serta kakak – kakak saya yang saya cintai, yang tidak pernah berhenti mendoakan, memberikan kasih sayang, cinta, perhatian, dan dukungan kepada saya selama ini. Terima kasih telah membimbing, mendidik, memotivasi dan mengiringi perjalanan saya hingga saat ini tanpa rasa lelah.
2. Ibu Enny Dwi Oktaviyani, ST.,M.Kom. selaku Dosen Pembimbing I dan Ibu Sherly Christina, S.Kom.,M.Kom selaku Dosen Pembimbing II yang ditengah - tengah kesibukannya telah menyediakan waktu dan memberikan bimbingan, arahan, serta motivasi kepada saya sehingga dapat menyelesaikan Tugas Akhir ini.
3. Ibu Felicia Sylviana, ST.,MM., Bapak Putu Bagus A.A.P., ST.,M.Kom. dan Ibu Licantik, S.Kom.,M.Kom. selaku Dosen Penguji, atas bimbingan, arahan dan memberikan saran serta perbaikan untuk Tugas Akhir ini sehingga dapat diselesaikan sebaik mungkin.
4. Dosen - dosen beserta staff di jurusan Teknik Informatika atas ilmu, pengalaman, dan bantuan yang telah diberikan selama ini.
5. Para sahabat saya yang luar biasa, selalu menjadi teman sekelompok dari awal kuliah, teman seperjuangan, teman dalam suka dan duka, teman berbagi informasi, ilmu, ide, pikiran dan tentunya hal – hal yang sebaiknya tidak saya sebutkan di sini.

KATA PENGANTAR

Puji dan Syukur kehadirat Tuhan Yang Maha Esa, karena berkat limpahan Rahmat dan Karunia-nya sehingga penulis dapat menyelesaikan laporan Tugas Akhir ini dengan judul "Implementasi Algoritma *Damerau-Levenhstein Distance* Untuk Memperbaiki Kesalahan Penulisan *Keyword* Pencarian" dengan baik, serta tepat pada waktunya.

Saya selaku penulis menyadari laporan ini masih jauh dari sempurna dan dalam penyelesaiannya tidak lepas dari bimbingan, arahan dan bantuan dari berbagai pihak, dalam kesempatan ini perkenankan saya untuk mengucapkan terima kasih dan penghargaan yang tinggi kepada yang terhormat Ibu Enny D. Oktaviyani, ST.,M.Kom. selaku Dosen Pembimbing I, dan Ibu Sherly Christina,S.Kom.,M.Kom., selaku Dosen Pembimbing II yang telah meluangkan waktunya untuk membimbing saya dan pihak-pihak yang telah membantu saya dalam penyusunan tugas akhir ini serta seluruh kerabat dan rekan-rekan yang telah membantu dalam pembuatan laporan ini.

Saya sangat berharap laporan Tugas Akhir ini dapat berguna dan berfungsi sebagai salah satu media yang dapat memfasilitasi proses belajar mengajar di Jurusan Teknik Informatika Universitas Palangka Raya.

Kiranya laporan tugas akhir ini dapat memberikan manfaat dan masukan bagi pembaca. Sebelumnya saya mohon maaf apabila terdapat kesalahan kata-kata yang kurang berkenan dan saya memohon kritik dan saran yang membangun demi perbaikan di masa depan.

Palangka Raya, 12 Maret 2018

Penulis

IMPLEMENTASI ALGORITMA DAMERAU-LEVENSHTEIN DISTANCE UNTUK MEMPERBAIKI KESALAHAN PENULISAN *KEYWORD* PENCARIAN

Ariando (DBC 113 087)
Jurusan Teknik Informatika Fakultas Teknik Universitas Palangka Raya
Kampus Tanjung Nyaho Jl. Yos Sudarso Palangka Raya 73112

Abstrak

Salah satu fitur yang umum dijumpai pada sebagian besar website adalah fitur pencarian data karena fitur ini mempermudah pencarian data atau halaman web. Sebagian besar fitur pencarian bekerja dengan cara menerima masukan (disebut *keyword*) dari seorang pengguna dalam bentuk frasa, kata – kata, atau bahkan kalimat, untuk kemudian dicari data yang relevan terhadap *keyword* pencarian oleh sistem di *database*nya. Oleh karena itu, hasil pencarian sangat dipengaruhi oleh keyword. Tidak jarang pengguna mengetik kata dalam *keyword* pencarian secara tidak tepat sehingga mempengaruhi hasil pencarian yang didapatkan menjadi kurang relevan atau bahkan tidak ada sama sekali. Sebuah kata bisa diketik secara tidak tepat disebabkan oleh beragam faktor, sehingga kadang mustahil untuk dihindari. Penelitian ini mencoba menyelesaikan masalah tersebut dengan cara memberikan saran perbaikan *keyword* jika terdapat kata yang diketik secara tidak tepat di dalam *keyword* tersebut. Saran perbaikan *keyword* diperoleh dengan cara menghitung jarak edit setiap kata di *keyword* dengan setiap kata yang ada pada kamus sistem berisi kata – kata yang diketik dengan benar.

Sistem yang dijadikan studi kasus dirancang dan dikembangkan menggunakan metodelogi pengembangan perangkat lunak *Waterfall*, sementara proses desain sistem dilakukan dengan pembuatan *Unified Modeling Language (UML).* Bahasa pemrograman yang digunakan yaitu PHP untuk logika program, dan MySQL untuk bagian *database.* Metode *testing* yang digunakan adalah Metode *Blackbox*, justifikasi pakar, dan perhitungan *precision* dan *recall*.

Pengujian dilakukan sebanyak 40 kali, dengan 10 data *keyword* pencarian dan 250 data artikel yang diambil secara acak dari internet. Hasil pengujian menunjukan bahwa algoritma *Damerau-Levenshtein Distance* mampu memberikan nilai *precision* dan *recall* sebanyak 91.24% dan 89.58% dalam memberikan saran perbaikan *keyword* berdasarkan 40 percobaan yang dijalankan. Hasil pengujian juga menunjukan bahwa *keyword* yang mengandung kata *typo* mempengaruhi hasil pencarian data yang didapatkan karena tidak ditemukan data yang mempunyai kecocokan. Setelah diperbaiki oleh sistem, baru kemudian terlihat hasil pencarian setiap uji coba meningkat dengan nilai *precision* 0.80 dan nilai *recall* 0.98 secara keseluruhan.

Kata kunci : pencarian data, *damerau-levenshtein distance*, *precision recall*, jarak edit.

IMPLEMENTATION OF DAMERAU-LEVENSHTEIN DISTANCE ALGORITHM TO FIX MISSPELLED WORDS IN A SEARCH ENGINE KEYWORD

Ariando (DBC 113 087)
Jurusan Teknik Informatika Fakultas Teknik Universitas Palangka Raya
Kampus Tanjung Nyaho Jl. Yos Sudarso Palangka Raya 73112

Abstract

One of the most common features found on a website is a search engine, because it is a great tool to ease the process of searching for specific data or web pages in a website database. Most search engines works by receiving an input (called keyword) from a human user in the form of a phrase, words, or even sentence, and then it starts searching for data in its database that are relevant to the given keyword. Thus, the returned data relies heavily on the keyword. It is not unusual that a user-typed keyword contains misspelled words, making it hard for a search engine to look for the expected relevant data by the user. A word can be misspelled due to many factors, so it is impossible to avoid it completely. This research proposes a way to solve this issue by giving a search keyword suggestion if a keyword does contains misspelled words. The search keyword suggestion obtained by calculating the edit distance of each word in the given keyword between every single word in a dictionary containing correctly spelled words, using Damerau-Levenshtein Distance Algorithm.

The software development methodology used to develop the system that is used as the case study is the *Waterfall* methodology, while the design is done using Unified Modeling Language (UML). The programming language used to build the system is PHP for the logic and MySQL for the database. The finished system is then tested using black box testing method and also calculating the precision and recall value for the search result and the search keyword suggestion. To make the search keyword suggestion test more reliable, search suggestions from the system is then compared to search keyword suggestions from the real human expert (Indonesian Language Lecturer with Doctoral degree).

The test is done 40 times, using 10 randomly picked search keyword samples from the internet and 250 articles data. The test results shows that, using Damerau-Levenshtein Distance algorithm only, the system is capable of getting 91.24% precision value and 89.58% recall value in giving the search keyword suggestions. While in finding the most relevant data to the given keyword, the system is capable of getting 0.80 precision value and 0.98 recall value. The test results also shows that misspelled word(s) in a search keyword has a huge impact on the returned search results.

Keywords : search engine, *damerau-levenshtein distance*, *precision recall*, *edit distance*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1. Latar Belakang

Seiring berjalannya waktu fitur – fitur yang dimiliki oleh website – website pada masa sekarang ini menjadi semakin canggih dan interaktif guna memenuhi kebutuhan dan tentunya mempermudah para penggunanya. Salah satu fitur yang penting dalam suatu website adalah fitur pencarian atau *searching*. Fitur ini mempermudah pengguna untuk menemukan apa yang diinginkan tanpa perlu menelusuri setiap halaman yang ada pada suatu website. Fitur pencarian ini berkerja dengan cara menerima masukan dari pengguna berupa *keyword* atau kata kunci tentang apa yang ingin dicari, kemudian *keyword* tersebut diproses dan dicocokan dengan data *searchable* yang ada di *database* website. Jika ditemukan data yang memiliki kecocokan dengan *keyword* yang dimasukan pengguna maka data dengan kecocokan tersebut akan ditampilkan ke pengguna sebagai kembaliannya, namun jika tidak ada data yang cocok maka akan ditampilkan hasil kosong atau pemberitahuan bahwa data yang dicari tidak ada. Suatu *keyword* bisa terdiri dari sebuah kata saja atau beberapa kata (frasa).

Fitur pencarian ini memang sangat memudahkan sekali bagi para pengguna website karena dapat menghemat banyak waktu dalam pencarian suatu informasi atau data. Namun, tidak jarang seorang pengguna melakukan kesalahan pengetikan atau penulisan suatu kata pada saat mengetikan *keyword* informasi yang ingin dicari. Kesalahan penulisan kata ini biasanya disebut dengan istilah *typo*. *Typo* ini bisa disebabkan oleh berbagai macam hal, seperti media pengetikan yang tidak sesuai dengan fisik pengguna (misalnya seperti pengguna berjari besar

dengan layar *touchscreen* yang kecil), kerusakan pada media pengetikan (misalnya seperti kerusakan pada *keyboard* atau *touchscreen*), minimnya kosa kata pengguna dan masih banyak lagi.

Kesalahan pengetikan suatu kata atau *typo* ini tentu akan berdampak pada hasil pencarian data. Misalnya, data *searchable* yang terdapat pada suatu *database* website adalah "Levenshtein algorithm" kemudian kata kunci pencarian yang dimasukan pengguna adalah "Levensten algoritm". Hasil proses pencarian tentu saja tidak ditemukan karena tidak ada kecocokan antara data yang ada dengan kata kunci pengguna.

Dengan adanya masalah tersebut maka diperlukan cara untuk memperbaiki kesalahan penulisan kata pada masukan kata kunci oleh pengguna. Solusinya adalah dengan membuat fitur *search suggestion* yang menampilkan *keyword* pengguna dengan kesalahan kata yang diperbaiki jika terdapat kesalahan penulisan kata pada masukan sebelumnya. Pendekatan yang digunakan adalah dengan menggunakan algoritma *Approximate String Matching Damerau-Levenshtein Distance* untuk mengecek setiap kata dari *keyword* yang dimasukan oleh pengguna untuk kemudian dibandingkan dengan kata – kata yang "benar" yang ada di *database* website. Kata – kata yang benar tersebut disimpan sebagai kamus di *database* website.

Algoritma *Damerau-Levenshtein Distance* bekerja dengan menghitung jarak antara *StringSumber* dengan *StringTarget*. Yang dimaksud dengan jarak adalah berapa kali sebuah *StringSumber* harus diubah hingga sama persis dengan *StringTarget*. Jika seandainya perbandingan antara *StringSumber* dan *StringTarget1* memiliki jarak sama dengan 2, sementara perbandingan antara

StringSumber dan *StringTarget2* memiliki jarak sama dengan 3, maka dapat disimpulkan bahwa *StringTarget1* lah kata yang mungkin ingin diketik oleh pengguna.

Dengan adanya implementasi algoritma *Damerau-Levenshtein Distance* untuk memperbaiki kesalahan penulisan *keyword* pencarian ini diharapkan ke depannya suatu website atau aplikasi dapat menampilkan hasil pencarian yang paling optimal dan relevan terhadap *keyword* yang dimasukan oleh pengguna.

1.2. Rumusan Masalah

Adapun rumusan masalah yang akan dibahas adalah :

1. Bagaimana kinerja algoritma *Damerau-levenshtein Distance* jika diimplementasikan untuk memperbaiki kesalahan penulisan *keyword* pencarian ?
2. Bagaimana pengaruh kesalahan penulisan kata dalam *keyword* pada hasil pencarian ?

1.3. Batasan Masalah

Pengimplementasian algoritma *Damerau-Levenshtein Distance* untuk memperbaiki kesalahan penulisan *keyword* pencarian ini memiliki beberapa batasan masalah, yaitu :

1. Studi kasus hanya terbatas pada fitur pencarian di website berbasis PHP.
2. *Search suggestion* yang berisi perbaikan kata – kata yang salah tidak ditampilkan secara *real time*.

3. Pengoreksian kesalahan penulisan kata hanya terbatas pada bahasa Indonesia saja, istilah – istilah atau *slang words* yang sering digunakan saat ini, dan beberapa nama organisasi, vendor, atau juga perusahaan yang terkenal saat ini. Pengoreksian kata juga tidak memperhatikan konteks atau makna dari *keyword*
4. Fitur pencarian data yang dicoba tidak menerapkan operator dan *query* khusus seperti layaknya *Google String Query* atau *Google Operators*.

1.4. Tujuan Penelitian

Tujuan dari penelitian ini adalah mengimplementasikan algoritma *Damerau-Levenshtein Distance* untuk memperbaiki kesalahan penulisan *keyword* pencarian.

1.5. Manfaat Penelitian

Adapun manfaat dari penelitian ini adalah untuk mengatasi masalah kesalahan penulisan *keyword* pencarian agar fitur pencarian dapat mendapatkan hasil yang optimal dan relevan terhadap *keyword* pengguna.

1.6. Metodologi Penelitian

A. Alat dan Bahan

1. Perangkat keras (*hardware*) yang adalah *note-book ASUS K401L* dengan spesifikasi sebagai berikut :
    1) Processor *intel core i5 5200U*;
    2) *Memory* 4 GB;

3) *Harddisk* 1 TB;

4) *VGA Intel HD Graphics dan nVidia GeForce 940m.*

2. Perangkat lunak (*software*) yang digunakan adalah:

1) Sistem Operasi (OS) : *Microsoft Windows 8.1*;

2) *Software editor* SublimeText, VS Code;

3) *Design : Miscrosoft Visio 2010;*

4) Bahasa Pemrograman : HTML, PHP, CSS, dan *Javascript*;

5) *Server Host* : XAMPP Server;

6) *Database* : MySQL.

B. Metodologi Pelaksanaan Penelitian

Metodologi pelaksanaan penelitian yang digunakan yaitu meliputi tahapan sebagai berikut :

1. Studi Literatur

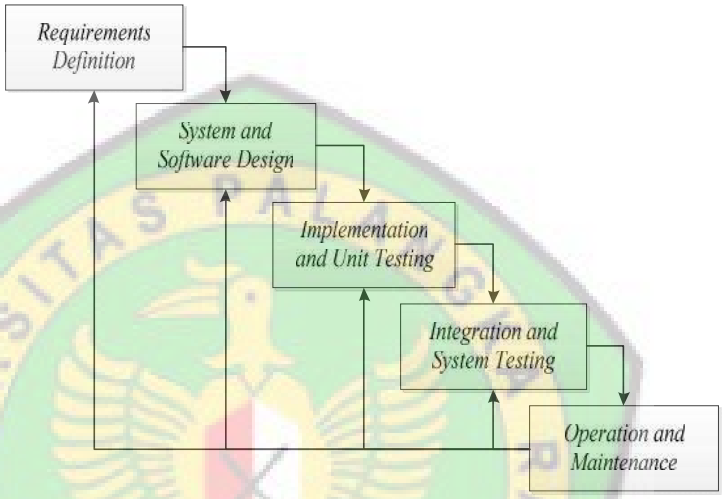
   Mempelajari buku dan informasi dari internet yang berhubungan dengan proses implementasi algoritma *Damerau-Levenshtein Distance* dan pengoreksian kata yang salah atau *spell checking*.

2. Pengumpulan Data

   Mengumpulkan data yang diperlukan untuk pengujian, seperti mengumpulkan kata – kata dalam bahasa Indonesia yang dianggap valid.

C. Metodologi Pengembangan Perangkat Lunak

Metodologi pengembangan perangkat lunak yang digunakan untuk mengembangkan *website* yang digunakan untuk penelitian adalah metode pengembangan *waterfall* menurut *Sommerville*. Model ini terbagi menjadi beberapa tahapan seperti yang terlihat pada gambar 1.1 berikut.



Gambar 1.1 Metode Pengembangan Sistem Waterfall (Sumber : Ian, Sommerville. 2009. *Software Engineering– 9th ed*)

1. *Requirements Definition* (Definisi Kebutuhan)

   Menganalisis kebutuhan yang diperlukan oleh *website* yang dijadikan studi kasus penelitian. Pada tahap analisis ini juga dilakukan pembuatan *Flowchart.*

2. *System and Software Design* (Perancangan sistem dan Perangkat Lunak)

   Pada tahap ini dilakukan desain *interface* web yang akan dibuat, rancangan akan disesuaikan dengan kebutuhan *website* yang akan

dijadikan sebagai studi kasus peneliian. Pada tahap design ini juga dilakukan pembuatan *Unified Modeling Language* (UML).

3. *Implementation and Unit Testing* (Implementasi dan pengujian unit)

   Penulisan program dengan menggunakan bahasa pemrograman PHP dan *MySQL*. Perancangan perangkat lunak direalisasikan sebagai serangkaian program atau unit program. Pengujian unit melibatkan verifikasi bahwa setiap unit telah memenuhi spesifikasinya.

4. *Integration and System Testing* (Integrasi dan Pengujian Sistem)

   Menguji sistem atau *website* yang telah dibuat, terutama menguji implementasi algoritma *Damerau-Levenshtein Distance* yang diterapkan sebagai metode *spell checking* untuk mengetahui apakah perbaikan kesalahan penulisan *keyword* pencarian yang dimasukan oleh pengguna sudah berhasil dan relevan terhadap *keyword* yang diinginkan oleh pengguna. Dengan kata lain, mengukur nilai *precision* dan *recall* algoritma yang digunakan.

1.7. Sistematika Penulisan

Berikut struktur penulisan laporan Tugas Akhir yang akan dihasilkan:

BAB I PENDAHULUAN

Pada bab ini akan diuraikan latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi dan sistematika penulisan serta jadwal pelaksanaan.

BAB II LANDASAN TEORI

Pada bab ini akan berisi teori-teori pendukung dalam peneletian, selain itu pada bab ini akan berisi materi-materi mengenai program-program pendukung yang akan digunakan.

BAB III ANALISIS DAN DESAIN

Pada bab ini akan berisi tentang analisis masalah yang sedang diteliti dan juga solusi yang ditawarkan. Pada bab ini juga membahas tentang rancangan website yang akan dijadikan bahan penelitian.

BAB IV IMPLEMENTASI DAN PENGUJIAN SISTEM

Pada bab ini akan membahas tentang pengujian sistem yang telah dikerjakan, menguji apakah sistem yang dikerjakan efektif dalam mendeteksi dan memperbaiki kesalahan penulisan kata atau *spell checking* serta menguji pengaruh kesalahan penulisan kata dalam *keyword* pencarian terhadap hasil pencarian yang didapat.

BAB V PENUTUP

Pada bab ini akan berisi kesimpulan dan saran dimana kesimpulan merupakan gagasan yang tercapai pada akhir penelitian dan saran merupakan usulan atau pendapat tentang perbaikan-perbaikan sistem yang sudah dibuat agar akan menjadi lebih baik kedepannya

1.8. Jadwal Pelaksanaan

Adapun jadwal dalam pembuatan program Mata Kuliah Tugas Akhir dapat dilihat pada Tabel 1:

Tabel 1. 1. Jadwal Pelaksanaan

RENCANA KEGIATAN	BULAN I				BULAN II				BULAN III				BULAN IV				BULAN V			
	MINGGU				MINGGU				MINGGU				MINGGU				MINGGU			
	I	II	III	IV	I	II	III	IV	I	II	III	IV	I	II	III	IV	I	II	III	IV
Penyusunan Proposal dan Seminar Proposal																				
Pembuatan Program dan Pengumpulan Data																				
Penyusunan Tugas Akhir dan Seminar Hasil																				
Pembuatan Laporan Akhir Tugas Akhir																				
Seminar Tugas Akhir																				

BAB II

LANDASAN TEORI

2.1. *Information Retrieval*

Information Retrieval (IR) adalah aktivitas untuk memperoleh sumber informasi (biasanya dokumen) dari sebuah data tidak terstruktur (biasanya teks) yang relevan atau memenuhi kebutuhan informasi dari sebuah data berukuran besar (biasanya tersimpan pada komputer) (Manning et al. 2009). IR didesain untuk memfasilitasi pengguna dalam mendapatkan kembali informasi yang behubungan dengan apa yang dibutuhkan pengguna secara efektif dan efisien (Manning et al. 2010). Di dalam sistem IR menggabungkan metode penyimpanan, indeksasi, filtering, pengorganisasian, pencarian dan menampilkan informasi. Berikut ini adalah fungsi utama sistem IR :

- Untuk mengidentifikasi sumber informasi yang relevan dengan target pengguna
- Untuk menganalisis isi dari sumber dokumen
- Untuk merepresentasikan isi dari sumber yang dianalisis sehingga dapat sesuai dengan query dari pengguna
- Untuk menganalisis query pengguna dan untuk merepresentasikannya ke dalam bentuk yang sesuai dengan database
- Untuk mencocokkan statemen pencarian dengan yang tersimpan dalam database
- Untuk mendapatkan kembali informasi yang relevan

- Untuk membuat pengaturan yang dibutuhkan dalam sistem berdasarkan feedback dari pengguna.

Ukuran efektifitas pencarian ditentukan oleh *precision* dan *recall*. *Precision* adalah rasio jumlah dokumen relevan yang ditemukan dengan total jumlah dokumen yang ditemukan (Ari Wibowo, 2011). *Recall* adalah adalah rasio jumlah dokumen relevan yang ditemukan kembali dengan total jumlah dokumen dalam kumpulan dokumen yang dianggap relevan (Ari Wibowo, 2011). *Recall* berhubungan dengan kemampuan sistem untuk memanggil dokumen yang relevan, sedangkan *precision* berkaitan dengan kemampuan sistem untuk tidak memanggil dokumen yang tidak relevan (Hasugian, 2006). Berikut adalah rumus perhitungan nilai *precision* dan *recall* menurut Hasugian.

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + TN}$$

Keterangan :

TP (*True Positive)* = Data relevan yang terambil

FP (*False Positive*) = Data tidak relevan tetapi terambil

TN (*True Negative*) = Data relevan tidak terambil

2.2. *Spelling Error*

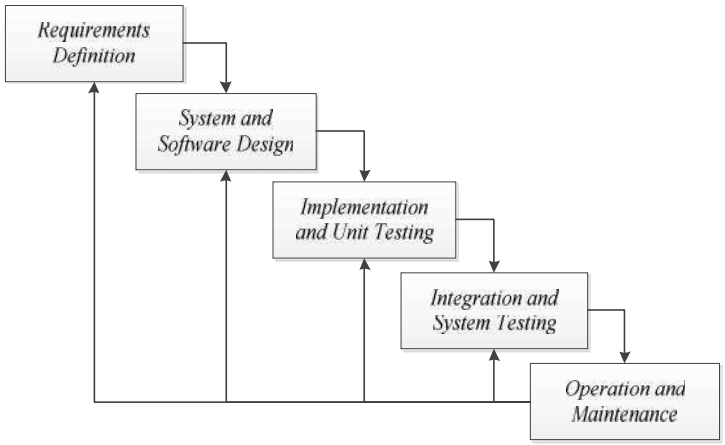
Spelling error merupakan keadaan di mana terjadi kesalahan penulisan susunan kata. Berdasarkan sejarahnya, awalnya keadaan ini berhubungan dengan kesalahan penulisan kata secara manual, namun saat ini hal tersebut juga dapat

terjadi pada proses pengetikan yang dilakukan dengan bantuan mesin ketik dan komputer. Hal tersebut dapat terjadi dikarenakan kesalahan mekanik atau keluputan tangan atau jari saat mengetik, selain itu terkadang juga disebabkan oleh ketidaktahuan seseorang tentang bagaimana pengejaan tulisan yang benar.

Berdasarkan jenis katanya *spelling error* dapat dibedakan menjadi 2 tipe yaitu *non-word spelling error* dan *real-word spelling error*. *Non-word spelling error* merupakan kesalahan penulisan kata di mana kata tersebut tidak dapat ditemukan dalam kamus (tidak memiliki makna). Sedangkan *real-word spelling error* merupakan kesalahan penulisan kata di mana kata tersebut dapat ditemukan dalam kamus (memiliki makna) namun bukan kata yang dimaksud dalam dokumen. Dalam menangani *non-word spelling error*, dibutuhkan kandidat kata *real-word* yang mirip dengan kata yang salah dengan ketentuan memiliki nilai *edit distance* terpendek. Sedangkan untuk mengatasi *real-word spelling error* dibutuhkan kandidat kata dengan pengucapan pengejaan yang mirip.

2.3. Metodologi Penelitian

Metodologi pengembangan perangkat lunak yang digunakan untuk mengembangkan *website* yang digunakan untuk penelitian adalah metode pengembangan *waterfall* menurut *Sommerville*. Model ini terbagi menjadi beberapa tahapan seperti yang terlihat pada gambar 2.1 berikut.



Gambar 2.1 Metode Pengembangan Sistem Waterfall

(Sumber : Ian, Sommerville. 2009. *Software Engineering– 9th ed*)

a. *Requirements Definition* (Definisi Kebutuhan)

Menganalisis kebutuhan yang diperlukan oleh *website* yang dijadikan studi kasus penelitian. Pada tahap analisis ini juga dilakukan pembuatan *Flowchart.*

b. *System and Software Design* (Perancangan sistem dan Perangkat Lunak)

Pada tahap ini dilakukan desain *interface* web yang akan dibuat, rancangan akan disesuaikan dengan kebutuhan *website* yang akan dijadikan sebagai studi kasus peneliian. Pada tahap design ini juga dilakukan pembuatan *Unified Modeling Language* (UML).

c. *Implementation and Unit Testing* (Implementasi dan pengujian unit)

Penulisan program dengan menggunakan bahasa pemrograman PHP dan *MySQL*. Perancangan perangkat lunak direalisasikan sebagai serangkaian program atau unit program. Pengujian unit

melibatkan verifikasi bahwa setiap unit telah memenuhi spesifikasinya.

d. *Integration and System Testing* (Integrasi dan Pengujian Sistem)

Menguji sistem atau *website* yang telah dibuat, terutama menguji implementasi algoritma *Damerau-Levenshtein Distance* yang diterapkan sebagai metode *spell checking* untuk mengetahui apakah perbaikan kesalahan penulisan *keyword* pencarian yang dimasukan oleh pengguna sudah berhasil dan relevan terhadap *keyword* yang diinginkan oleh pengguna.

e. *Operation and Maintenance* (Operasi dan Pemeliharaan)

Mengoperasikan program dilingkungannya, sesuai dengan kebutuhan *user* dan melakukan *maintenance* atau pemeliharaan. Biasanya merupakan fase siklus yg paling lama (walaupun tidak seharusnya). Tahap ini tidak dilakukan dalam penelitian ini karena hasil produk atau sistem yang diselesaikan tidak digunakan dalam tahap produksi.

2.4. Perangkat Analisis Sistem

Analisis sistem memiliki pengertian dari ahlinya, yaitu menurut Drs. Komarudin, Analisis Sistem adalah susunan yang teratur dari kegiatan yang berhubungan satu sama lainnya serta prosedur – prosedur yang berkaitan untuk melaksanakan dan memudahkan pelaksanaan kegiatan dari suatu organisasi.

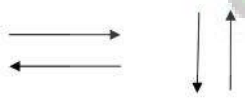
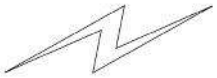
A. *Flowchart*

Flowchart adalah representasi grafis dari langkah – langkah yang harus diikuti dalam menyelesaikan suatu permasalahan yang terdiri atas sekumpulan simbol, dimana masing – masing simbol merepresentasikan kegiatan tertentu. *Flowchart* diawali dengan penerimaan input dan diakhiri dengan penampilan *output*. Bagan alir program (*program flowchart*) merupakan bagian yang menjelaskan secara rinci langkah-langkah dari proses program. (Ema Utami dan Sukrisno, 2005)

Simbol-simbol yang di pakai dalam *flowchart* dibagi menjadi 3 kelompok :

1. *Flow Direction Symbols*, digunakan untuk menghubungkan simbol satu dengan yang lain. Simbol ini disebut juga *connecting line*. Simbol-simbol tersebut adalah sebagai berikut :

Tabel 2.1. *Flow Directions Symbols*

Simbol	Keterangan
	Simbol arus / *flow*, yaitu menyatakan jalannya arus suatu proses
	Simbol *communication link*, yaitu menyatakan transmisi data atau informasi dari satu lokasi ke lokasi lain
	Simbol *connector*, berfungsi menyatakan sambungan dari proses ke proses lainnya dalam halaman atau lembar yang sama

Tabel 2.1. *Flow Directions Symbols* (Lanjutan)

Simbol	Keterangan
	Simbol *offline connector*, menyatakan sambungan dari proses ke proses lainnya dalam halaman atau lembar yang berbeda

2. *Processing Symbols*, menunjukan jenis operasi pengolahan dalam suatu proses atau prosedur.

Tabel 2.2. *Processing Symbols*

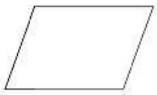
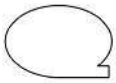
Simbol	Keterangan
	Simbol *process*, yaitu menyatakan suatu tindakan (proses) yang dilakukan oleh komputer
	Simbol *manual*, yaitu menyatakan suatu tindakan (proses) yang tidak dilakukan oleh komputer (manual)
	Simbol *decision*, yaitu menujukkan suatu kondisi tertentu yang akan menghasilkan dua kemungkinan jawaban : ya / tidak
	Simbol predefined process, yaitu menyatakan penyediaan tempat penyimpanan suatu pengolahan untuk memberi harga awal.
	Simbol *terminal*, yaitu menyatakan permulaan atau akhir suatu program

Tabel 2.2. *Processing Symbols* (Lanjutan)

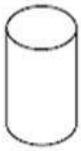
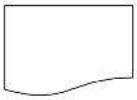
Simbol	Keterangan
	Simbol *keying operation*, Menyatakan segal jenis operasi yang diproses dengan menggunakan suatu mesin yang mempunyai *keyboard*
	Simbol *offline-storage*, menunjukkan bahwa data dalam simbol ini akan disimpan ke suatu media tertentu
	Simbol *manual input*, memasukkan data secara manual dengan menggunakan online keyboard

3. *Input/Output Symbols*, menunjukkan jenis peralatan yang digunakan sebagai media input atau output.

Tabel 2.3. *Input / Output Symbols*

Simbol	Keterangan
	Simbol *input/output*, menyatakan proses input atau output tanpa tergantung jenis peralatannya
	Simbol *punched card*, menyatakan input berasal dari kartu atau output ditulis ke kartu
	Simbol *magnetic tape*, menyatakan input berasal dari pita magnetis atau output disimpan ke pita magnetis

Tabel 2.3. *Input / Output Symbols* (Lanjutan)

Simbol	Keterangan
	Simbol *magnetic disk*, menyatakan input berasal dari disk magnetis atau output disimpan ke disk magnetis
	Simbol *document*, mencetak keluaran dalam bentuk dokumen (melalui printer)
	Simbol *disk storage*, menyatakan input berasal dari dari disk atau output disimpan ke disk
	Simbol *display*, mencetak keluaran dalam layar monitor

B. *Unified Modeling Language* (UML)

Menurut Widodo, (2011:6), "UML adalah bahasa pemodelan standar yang memiliki sintak dan semantik". Menurut Nugroho (2010:6), "UML (*Unified Modeling Language*) adalah bahasa pemodelan untuk sistem atau perangkat lunak yang berparadigma (berorientasi objek)." Pemodelan (*modeling*) sesungguhnya digunakan untuk penyederhanaan permasalahan-permasalahan yang kompleks sedemikian rupa sehingga lebih mudah dipelajari dan dipahami. Jenis – jenis diagram UML yang digunakan adalah sebagai berikut :

1. *Usecase Diagram*

Menurut Rosa A. S dan M. Shalahuddin (2013:155), *usecase diagram*, atau diagram *usecase* merupakan pemodelan untuk kelakukan (*behavior*) sistem informasi yang akan dibuat. Usecase

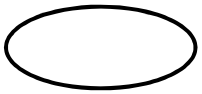
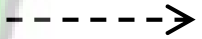
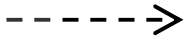
mendeskripsikan sebuah interaksi antara saru atau lebih actor dengan sestem informasi yang akan dibuat. Secara kasar, *usecase* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu.

Diagram *Use case* menyajikan interaksi antara use case dan aktor dalam sistem yang akan dikembangkan. *Use case* adalah fungsionalitas atau persyaratan-persyaratan sistem yang harus dipenuhi oleh sistem yang akan dikembangkan menurut pandangan pemai sistem. Sedangkan aktor bisa berupa orang, peralatan, atau sistem lain yang berinteraksi terhadap sistem yang akan dibangun.

Tabel 2.4. Simbol – simbol diagram *Usecase*

Elemen *Use case*	**Uraian**	**Simbol Elemen**
Aktor	Aktor adalah para pengguna (*user*) dari sebuah sistem. Aktor adalah seseorang atau sesuatu yang harus berinteraksi / dikembangkan dengan sistem.	

Tabel 2.4. Simbol – simbol diagram *Usecase* (Lanjutan)

Elemen *Use case*	Uraian	Simbol Elemen
Use case	Gambar *use case* menggunakan lingkaran berbentuk bulat telur (*ovals*) yang diberi naam dengan kata kerja (*verbs*) yang menggambarkan fungsi-fungsi sistem	 *Use case*
Generalization	*Generalization* merupakan hubungan yang menyatakan bahwa elemen spesial (anak) dapat digantikan oleh objek *general* (orangtua)	 Generalization
include	Menghubungkan antara 2 atau lebih use case untuk menunjukan *use case* tersebut merupakan bagian dari base *use case*.	 <<include>>
extend	Menghubungkan antara dua atau lebih *use case* yang merupakan tambahan dari base *use case* yang biasanya untuk mengatasi kasus pengecualian.	 <<extend>>

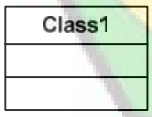
2. *Class Diagram*

Menurut Rosa A. S dan M. Shalahuddin(2013:141), Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun

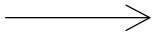
sistem, kelas memiliki apa yang disebut atribut dan metode atau operasi. *Atribut* merupakan *variable-variabel* yang dimiliki oeh sesuatu kelas, sementara Operasi atau metode adalah fungsi-fungsi uang dimiliki suatu kelas.

Diagram kelas dibuat agar pembuat program membuat kelas-kelas sesuai rancangan di dalam diagram kelas agar dokumentasi perancangan dan perangkat lunak singkton. Banyak berbagai kasus, perancangan kelas yang dibuat tidak sesuai dengan kelas-kelas yang dibuat pada perangkat lunak, sehingga tidaklah ada gunanya lagi sebuah perancangan karena apa yang dirancang dan hasilnya jadinya tidak sesuai.

Tabel 2.5. Simbol - simbol *Class Diagram*

Simbol	**Deskripsi**
kelas 	Kelas pada struktur sistem
Antar muka/ *interface* Nama_*interface* 	Sama dengan konsep interface dalam pemrograman berorientasi objek
Asosiasi/*association* _____	Relasi antarkelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity*

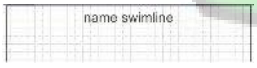
Tabel 2.5. Simbol - simbol Class Diagram

Simbol	Deskripsi
Asosiasi berarah / *directed association* 	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity*
Generalisasi	Relasi antar kelas dengan makna generalisasi-spesialisasi (umum khusus)
Ketergantungan / *dependency*	Relasi antar kelas dengan makna kebergantungan antar kelas
Agregasi / *aggregation*	Relasi antar kelas dengan makna semua bagian (*whole-part*)

3. *Activity Diagram*

Menurut Rosa A. S dan M. Shalahuddin (2013:161), Diagram aktifitas menggambarkan workflow (aliran kerja) atau aktifitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan actor, jadi aktivitas yang dapat dilakukan oleh sistem.

Tabel 2.6. Simbol - simbol *Activity* Diagram

Simbol	Keterangan
Status awal	Status awal aktifitas sistem, sebuah diagram aktifitas memiliki sebuah status awal
Status akhir	Status akhir aktifitas sistem, sebuah diagram aktifitas memiliki sebuah status akhir
aktifitas	Aktifitas yang dilakukan sistem, aktifitas biasanya diawalu dengan kata kerja
Percabangan / *decision*	Asosiasi percabangan dimana jika ada pilihan aktifitas lebih dari satu
Penggabungan / *join*	Asosiasi penggabungan dimana lebih dari satu aktifitas digabungkan menjadi satu.
Transition 	State transition menunjukan kegiatan apa berikutnya setelah suatu kegiatan sebelumnya
Swimline 	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktifitas yang terjadi.

2.5. *Text Preprocessing*

Tahap proses awal terhadap teks untuk mempersiapkan teks menjadi data yang akan diolah lebih lanjut. Sekumpulan karakter yang bersambungan (teks) harus dipecah-pecah menjadi unsur yang lebih berarti. Hal ini dapat dilakukan dalam beberapa tingkatan yang berbeda.

A. *Case Folding*

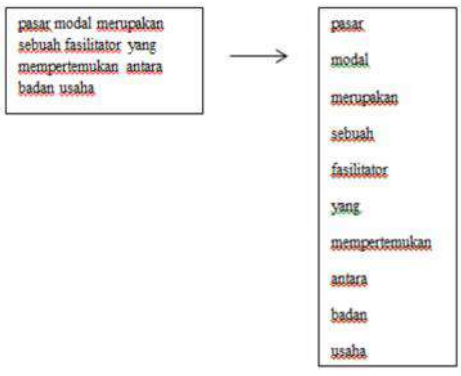
Merupakan tahapan untuk merubah semua huruf di dalam teks menjadi *lowercase*. Misalkan, terdapat kalimat seperti berikut :

"Algoritma *Approximate String Matching*" akan diubah menjadi "algoritma *approximate string matching"*.

B. *Tokenization*

Tokenization adalah proses memecah teks menjadi kalimat dan kata/token (Feldman, R. & Sanger, J.). Fitur ini terdiri dari tipe kapitalisasi, keberadaan digit, tanda baca, karakter spesial dan lain sebagainya. Karakter petik tunggal ('), titik (.), semikolon (;), titik dua (:) atau lainnya, juga dapat memiliki peran yang cukup banyak sebagai pemisah kata.

Sebuah titik (.) biasanya untuk tanda akhir kalimat, tapi dapat juga muncul dalam singkatan, inisial orang, alamat internet, dan lain-lain. Kemudian tanda *hypen* (-) biasanya muncul untuk menggabungkan dua token yang berbeda untuk membentuk token tunggal. Tapi dapat pula ditemukan untuk menyatakan rentang nilai, kata berulang, dan sebagainya. Atau karakter *slash* (/) sebagai pemisah file atau direktori atau url ataupun untuk menyatakan "dan atau".



Gambar 2.2. Proses *Tokenization*

2.6. *Damerau-Levenshtein Distance*

Algoritma *Damerau-Levenshtein Distance* merupakan algoritma pengembangan dari algoritma *Levenshtein Distance*. *Damerau-Levenshtein Distance* menentukan jumlah minimum operasi yang dibutuhkan untuk mengubah satu string menjadi string lain, di mana operasi yang digunakan sama dengan *Levenshtein Distance* yaitu *insertion* (penambahan karakter), *deletion* (penghapusan karakter), *substitution* (pergantian karakter) namun dengan penambahan operasi *transposition* (penukaran posisi karakter) diantara dua karakter berdempetan (Damerau dalam Jupin, Shi, & Obradovic, 2013).

Setiap kesalahan berupa hilangnya karakter huruf, kelebihan karakter huruf, atau kesalahan urutan huruf dari dua karakter huruf yang berbeda (contoh: seharusnya tertulis “ka”, diketik dengan “ak”) dianggap sebagai 1 kesalahan.

```
algorithm damerau-levenshtein distance is
    input: strings a[1..length(a)], b[1..length(b)]
    output: distance, integer
    let d[0..length(a), 0..length(b)] be a 2-d array of integers, dimensions length(a)+1, length(b)+1
    for i := 0 to length(a) inclusive do
        d[i, 0] := i
    for j := 0 to length(b) inclusive do
        d[0, j] := j
    for i := 1 to length(a) inclusive do
        for j := 1 to length(b) inclusive do
            if a[i] = b[j] then
                cost := 0
            else
                cost := 1
            d[i, j] := minimum(d[i-1, j] + 1,     // deletion
                               d[i, j-1] + 1,     // insertion
                               d[i-1, j-1] + cost)  // substitution
            if i > 1 and j > 1 and a[i] = b[j-1] and a[i-1] = b[j] then
                d[i, j] := minimum(d[i, j],
                                   d[i-2, j-2] + cost)  // transposition
    return d[length(a), length(b)]
```

Gambar 2.3. Algoritma *Damerau-Levenshtein Distance*

Berikut adalah **contoh** perbandingan antara *Damerau-Levenshtein Distance* dan *Levenshtein Distance* dalam menghitung jarak perbedaan antara kata “***twittre***” (*string* sumber) dan kata “***twitter***” (*string* target).

1. Menggunakan algoritma *Damerau-Levenshtein Distance*

			String Target						
			t	w	i	T	t	e	r
		0	1	2	3	4	5	6	7
String Sumber	t	1	0	1	2	3	4	5	6
	w	2	1	0	1	2	3	4	5
	i	3	2	1	0	1	2	3	4
	t	4	3	2	1	0	1	2	3
	t	5	4	3	2	1	0	1	2
	r	6	5	4	3	2	1	1	1
	e	7	6	5	4	3	2	1	1

Dengan menggunakan algoritma *Damerau-Levenshtein Distance* jarak yang dihasilkan adalah 1, karena perubahan yang diperlukan adalah menukar posisi karakter “r” dan “e” dengan *cost* sama dengan 1.

2. Menggunakan algoritma *Levenshtein Distance*

			String Target						
			t	w	i	t	t	e	r
		0	1	2	3	4	5	6	7
String Sumber	t	1	0	1	2	3	4	5	6
	w	2	1	0	1	2	3	4	5
	i	3	2	1	0	1	2	3	4
	t	4	3	2	1	0	1	2	3
	t	5	4	3	2	1	0	1	2
	r	6	5	4	3	2	1	1	3
	e	7	6	5	4	3	2	1	2

Dengan menggunakan algoritma *Levenshtein Distance* jarak yang dihasilkan adalah 2, karena perubahan yang diperlukan adalah dengan mensubtitusi karakter “r” dan “e” kemudian mensubtitusi karakter “e” dan “r” dengan *cost* masing – masing subtitusi adalah 1.

2.7. Tinjauan Pustaka

Dalam penelitian yang berjudul "Implementasi Algoritma *Damerau-Levenshtein Distance* Untuk Memperbaiki Kesalahan Penulisan *Keyword* Pencarian" ini akan dilakukan analisis sebagai pembanding dengan penelitian yang telah ada sebelumnya seperti pada Tabel 2.1 :

Tabel 2.7. Tinjauan Pustaka

	I	II
Judul	**Koreksi Ejaan Query Bahasa Indonesia Menggunakan Algoritma Damerau-Levenshtein**	**Deteksi Kesalahan Ejaan dan Penentuan Rekomendasi Koreksi Kata yang Tepat Pada Dokumen Jurnal JTIIK Menggunakan Dictionary Lookup dan Damerau-Levenshtein Distance**
(Nama, Tahun)	Utis Sutisna, Julio Adisantoso, 2010	Tusty Nadia Maghfira, Imam Cholissodin, Agus Wahyu Widodo, 2017
Inti Sari	Penelitian ini menjelaskan tentang pengoreksian ejaan Bahasa Indonesia pada dokumen teks dengan menerapkan algoritma *Damerau-Levenshtein Distance*	Penelitian ini dilakukan untuk mendeteksi kesalahan ejaan dan menentukan rekomendasi koreksi kata yang tepat pada dokumen jurnal dengan menggunakan metode

	untuk mengukur jarak antar *string sumber* dan *string target*. Penelitian ini juga membandingkan nilai *precision* dan *recall* antara algoritma *Levenshtein Distance* dan algoritma *Damerau-Levenshtein Distance*. Berdasarkan hasil penelitian ini didapatkan bahwa algoritma *Damerau-Levenshtein Distance* memiliki nilai *precision* dan *recall* lebih baik dibandingkan dengan algoritma *Levenshtein Distance.*	*Dictionary Lookup* dan *Damerau-Levenshtein Distance*. Teks di dalam dokumen jurnal diekstrak dengan metode *Text-Mining* untuk kemudian dideteksi ejaan kata atau kalimat yang terdapat pada teks tersebut dengan menggunakan algoritma *Damerau-Levenshtein Distance*. Penelitian ini menyimpulkan bahwa algoritma *Damerau-Levenshtein Distance* dapat diimplementasikan dengan baik pada proses deteksi dan koreksi kesalahan ejaan kata pada suatu dokumen jurnal

BAB III

ANALISIS DAN DESAIN

3.1. Analisis

Pada tahap ini dilakukan analisis system yang akan dijadikan studi kasus untuk mengimplementasikan algoritma *Damerau-levenshtein Distance* untuk memperbaiki kesalahan penulisan *keyword* pencarian.

A. Algoritma Damerau-Levenshtein Distance

Algoritma *Damerau-Levenshtein Distance* yang diimplementasikan tidak berbeda dengan versi aslinya yang mendukung empat operasi *edit*, yaitu pengurangan, penambahan, subtitusi dan transposisi seperti pada gambar berikut.

```
algorithm damerau-levenshtein distance is
    input: strings a[1..length(a)], b[1..length(b)]
    output: distance, integer
    let d[0..length(a), 0..length(b)] be a 2-d array of integers, dimensions length(a)+1, length(b)+1
    for i := 0 to length(a) inclusive do
        d[i, 0] := i
    for j := 0 to length(b) inclusive do
        d[0, j] := j
    for i := 1 to length(a) inclusive do
        for j := 1 to length(b) inclusive do
            if a[i] = b[j] then
                cost := 0
            else
                cost := 1
            d[i, j] := minimum(d[i-1, j] + 1,     // deletion
                               d[i, j-1] + 1,     // insertion
                               d[i-1, j-1] + cost)  // substitution
            if i > 1 and j > 1 and a[i] = b[j-1] and a[i-1] = b[j] then
                d[i, j] := minimum(d[i, j],
                                   d[i-2, j-2] + cost)  // transposition
    return d[length(a), length(b)]
```

Gambar 3.1 Algoritma Damerau-Levenshtein Distance

Algoritma *Damerau-Levenshtein Distance* di atas akan menghasilkan jarak edit antara *string* target dan *string* sumber. Jarak edit inilah yang dibutuhkan oleh system untuk menentukan apakah terdapat kesalahan penulisan pada suatu kata.

B. Kamus Kata Pembanding

Kata – kata yang dijadikan sebagai pembanding disimpan sebagai kamus di *database*. Kata – kata pembanding ini terbatas pada bahasa Indonesia yang sesuai dengan KBBI (Kamus Besar Bahasa Indonesia) saja dan beberapa kata lainnya seperti nama orang.

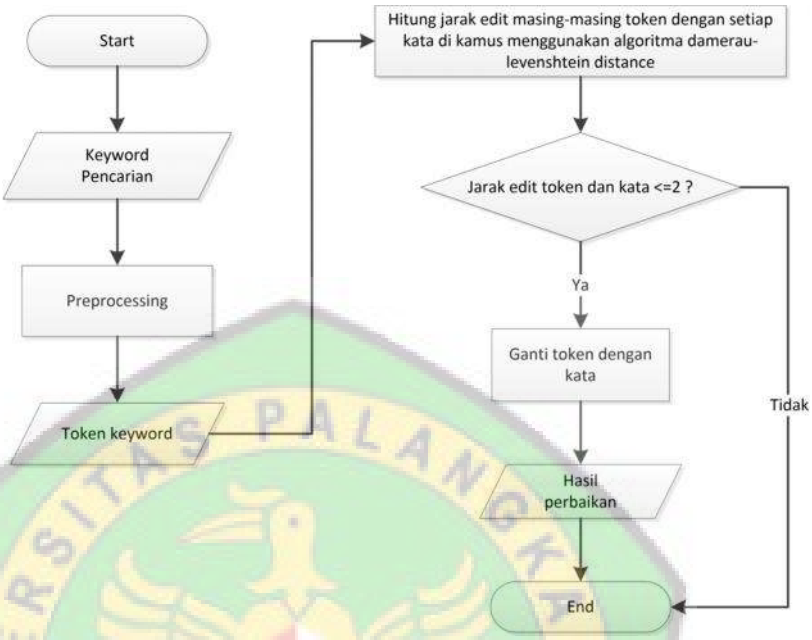
Tabel 3.1 Struktur table Kamus

word (kata)	first_l (huruf depan)
a	a
aba	a
abad	a
abadi	a
abadiah	a
abdi	a
abdomen	a
abduksi	a
abduktor	a
aberasi	a
abisal	a
abiseka	a
dst	

Jumlah kata yang disimpan untuk pengujian adalah sebanyak 20.000 kata. Setiap kata yang ada pada *keyword* pencarian pengguna akan dihitung jarak *edit*nya dengan setiap kata yang ada pada kamus.

C. Metode Pengecekan dan Perbaikan *Keyword*

Flowchart berikut ini menggambarkan tahapan pengecekan dan perbaikan *keyword* pencarian yang dimasukan.



Gambar 3.2 Metode pengecekan dan perbaikan *keyword*

Deskripsi *flowchart* pada gambar 3.2 :

1. Menerima *keyword* pencarian
2. Melakukan *preprocessing keyword* pencarian dengan langkah – langkah sebagai berikut :
    1) *Case folding*, yaitu tahapan merubah semua huruf di *keyword* pencarian menjadi *lowercase.*
    2) *Tokenization*, yaitu tahapan menghapus semua tanda baca dari *keyword* jika terdapat tanda baca dan kemudian memisahkan deretan kata dalam *keyword* menjadi *token* atau potongan kata tunggal jika *keyword* berbentuk frase.
3. *Token – token keyword* dihasilkan dari tahap *preprocessing*

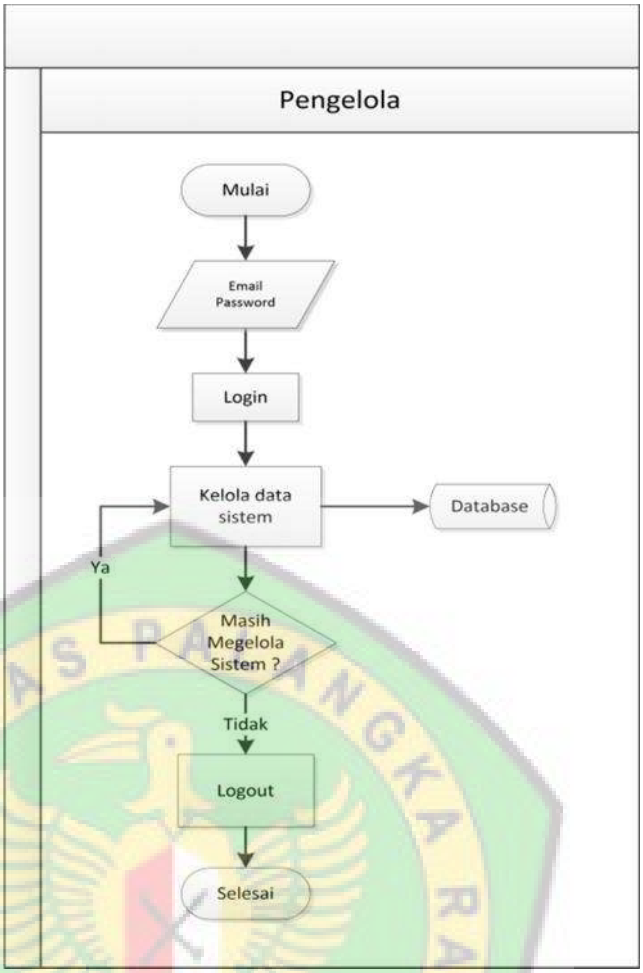
4. Menghitung jarak edit masing – masing *token* dengan setiap kata yang ada di kamus menggunakan algoritma *Damerau-Levenshtein Distance*.
5. Menentukan apakah jarak edit antara satu *token* dengan kata pembanding yang ada di kamus berjumlah <= 2. Jika jarak edit <= 2 maka proses akan berlanjut ke langkah berikutnya, jika tidak maka proses berakhir.
6. Mengganti *token* yang memiliki jarak edit <= 2 dengan kata pembanding.
7. Hasil perbaikan didapatkan. Selesai.

D. Proses Bisnis Sistem

Sistem yang dijadikan sebagai studi kasus penelitian ini adalah system atau *web* yang menyimpan data *searchable* yang dalam hal ini adalah data artikel (tidak terbatas pada suatu topik apapun). Layaknya sebuah system pada umumnya, terdapat pengelola dan pengguna atau pengunjung yang terlibat di dalam system.

1. Pengelola Sistem

Proses bisnis berikut ini tujuannya adalah untuk menggambarkan bagaimana proses yang terjadi pada saat pengelola system mengelola data yang ada pada system.

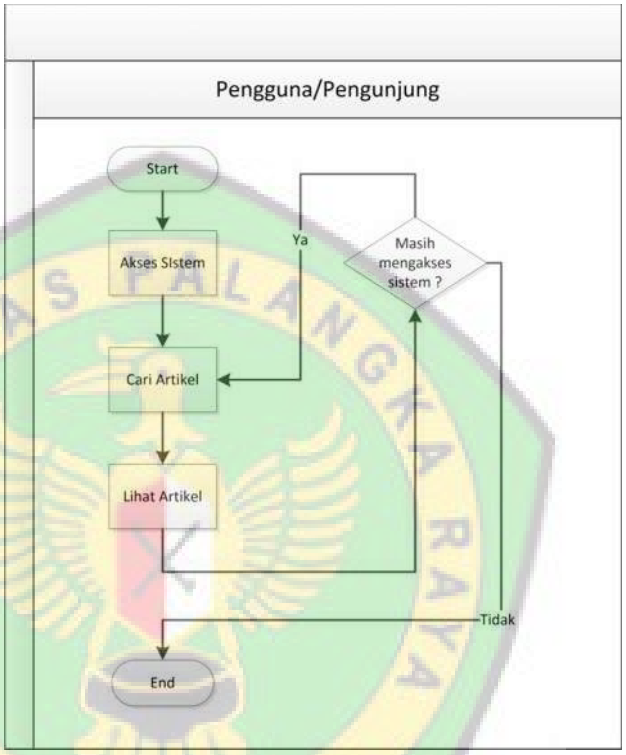


Gambar 3.3 Proses bisnis pengelola system

1) Pengelola memasukan *email* dan *password.*
2) Sistem meloginkan pengelola.
3) Pengelola kemudian mengelola data – data yang ada pada system.
4) Jika pengelola masih mengelola system, maka kembali ke tahap 4. Jika sudah selesai maka lanjut ke tahap 6.
5) Pengelola keluar dari system. Selesai.

2. Pengguna/Pengunjung

Proses bisnis berikut ini tujuannya adalah untuk menggambarkan bagaimana proses yang terjadi pada saat pengguna mengakses system untuk memperoleh artikel yang diinginkan.



Gambar 3.4 Proses bisnis pengguna/pengunjung

1) Pengguna mengakses system
2) Pengguna mencari artikel yang diinginkan
3) Pengguna melihat artikel yang diinginkan
4) Pengguna kemudian menentukan apakah masih mengakses system atau tidak. Jika iya, maka kembali ke langkah 3 atau 4. Selesai.

E. Fungsionalitas

Pada tahap ini akan dibahas fungsionalitas yang terdapat pada system yang dirancang sebagai studi kasus. Yang dimaksud dengan fungsionalitas adalah fitur – fitur yang ada pada system.

1. Pengelola Sistem

1) Login

Sebelum memulai mengelola data yang ada pada system, pengelola harus *login* terlebih dahulu dengan memasukan kresendisial berupa *email* dan *password*.

2) Kelola Akun

Pengelola dapat mengelola akunnya sendiri yang digunakan seperti misalnya mengganti kredensial akun.

3) Kelola Artikel

Pengelola dapat mengelola data artikel di dalam system. Mulai dari menambahkan artikel baru, mengedit, melihat dan menghapus artikel yang sudah ada.

4) Kelola Kamus

Pengelola dapat mengelola data kata – kata yang dijadikan sebagai pembanding dengan kata – kata yang terdapat di *keyword* pencarian. Kata – kata ini disimpan sebagai kamus di *database*. Pengelolaan meliputi menambah kata baru, mengedit, melihat dan menghapus kata yang sudah ada.

5) Cari Artikel

Pengelola dapat melakukan pencarian artikel dengan cara memasukan *keyword* pencarian di *form* pencarian data.

2. Pengguna/Pengunjung

1) Lihat Artikel

Pengguna/pengunjung dapat melihat artikel yang ada di dalam system.

2) Cari Artikel

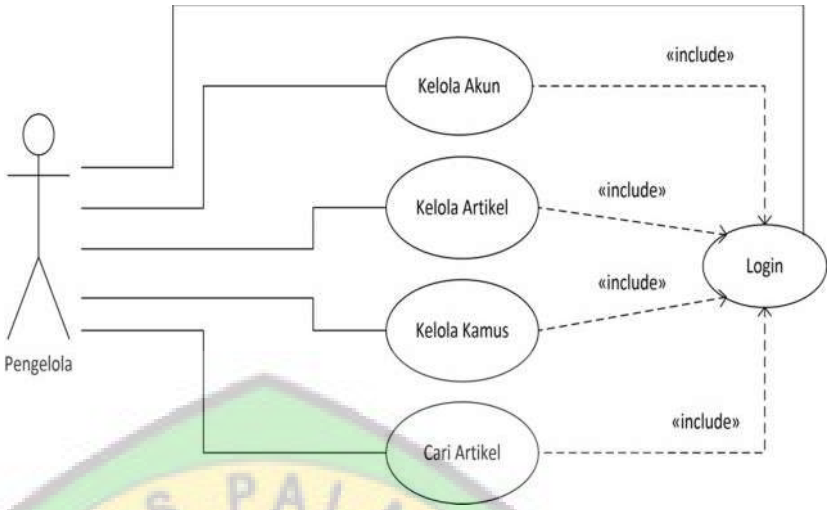
Pengguna/pengunjung dapat melakukan pencarian artikel dengan cara memasukan *keyword* pencarian di *form* pencarian data.

3.2. *Unified Modeling Languange* (UML)

Pada tahap ini akan dijelaskan *Unified Modeling Language* (UML) untuk menggambarkan proses – proses yang terdapat di dalam system. Diagram UML yang digunakan meliputi : *Usecase diagram, Activity diagram,* dan *Class diagram.*

A. *Usecase Diagram*

1. Pengelola Sistem



Gambar 3.5 *Usecase diagram* pengelola system

Berikut adalah penjelasan dari *usecase diagram* pengelola system pada gambar di atas :

1) Login

Proses di mana pengelola system melakukan *login* untuk dapat mengelola sistem. Proses ini merupakan proses autentikasi standar pada suatu aplikasi web.

Tabel 3.2 *Usecase* Login

Nama *usecase*	Login
Aktor	Pengelola

Tabel 3.3 *Usecase Login* (lanjutan)

Pre condition	Sistem menyediakan *form login*
Post condition	Sistem menampilkan halaman *web* dengan hak akses pengelola
Tujuan	Login untuk dapat mengelola system
Skenario	Pengelola memasukan data berupa *email* dan *password* di *form login*

2) Kelola Akun

Proses ini merupaka proses di mana pengelola dapat mengubah data akunnya sendiri.

Tabel 3.4 *Usecase* Kelola Akun

Nama *usecase*	Kelola Akun
Aktor	Pengelola
Pre condition	Sistem menyediakan menu Kelola Akun
Post condition	Sistem menampilkan halaman pengelolaan akun
Tujuan	Mengelola data akun milik aktor yang bersangkutan
Skenario	Pengelola melakukan proses perubahan data akun berupa nama, *email*, dan *password*.

3) Kelola Artikel

Proses ini merupakan proses di mana pengelola dapat mengelola data artikel di dalam system.

Tabel 3.5 *Usecase* Kelola Artikel

Nama *usecase*	Kelola Artikel
Aktor	Pengelola
Pre condition	Sistem menyediakan menu kelola artikel
Post condition	Sistem menampilkan halaman pengelolaan artikel
Tujuan	Mengelola data artikel yang ada di system
Skenario	Pengelola menambah data artikel baru, melihat, mengedit dan menghapus yang sudah ada.

4) Kelola Kamus

Proses ini merupakan proses di mana pengelola dapat mengelola data kamus di dalam system.

Tabel 3.6 *Usecase* Kelola Kamus

Nama *usecase*	Kelola Kamus
Aktor	Pengelola

Tabel 3.7 *Usecase* Kelola Kamus (lanjutan)

Pre condition	Sistem menyediakan menu kelola kamus
Post condition	Sistem menampilkan halaman pengelolaan kamus
Tujuan	Mengelola data kata – kata yang disimpan sebagai kamus di system
Skenario	Pengelola menambah data kata baru, melihat, mengedit dan menghapus yang sudah ada.

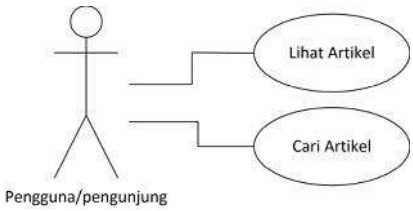
5) Cari Artikel

Proses ini merupakan proses di mana pengelola dapat mencari data artikel yang ada di system.

Tabel 3.8 *Usecase* Cari Artikel

Nama *usecase*	Cari Artikel
Aktor	Pengelola
Pre condition	Sistem menyediakan *form* pencarian
Post condition	Sistem menampilkan hasil pencarian
Tujuan	Mencari data artikel yang ada di system
Skenario	Pengelola memasukan *keyword* pencarian di *form* pencarian

2. Pengguna/Pengunjung



Gambar 3.6 *Usecase diagram* pengguna/pengunjung system

Berikut adalah penjelasan dari *usecase diagram* pengguna/pengunjung system pada gambar di atas :

1) Lihat Artikel

Proses ini adalah proses di mana pengguna/pengunjung system dapat melihat data artikel yang ada.

Tabel 3.9 *Usecase* Lihat Artikel

Nama *usecase*	Lihat Artikel
Aktor	Pengguna/pengunjung
Pre condition	Sistem menyediakan halaman yang berisi data – data artikel
Post condition	Sistem menampilkan artikel yang dipilih oleh pengguna
Tujuan	Melihat artikel yang ada di system
Skenario	Pengguna/pengunjung memilih dan mengklik artikel yang ingin dilihat

2) Cari Artikel

Proses ini adalah proses di mana pengguna/pengunjung system dapat mencari data artikel yang ada di system.

Tabel 3.10 *Usecase* Cari Artikel (Pengguna/pengunjung)

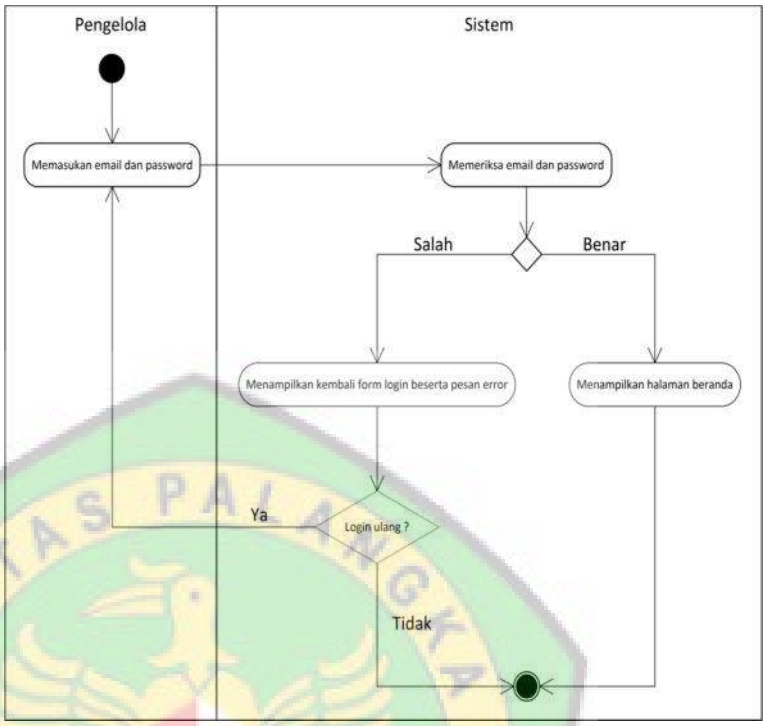
Nama *usecase*	Cari Artikel
Aktor	Pengguna/pengunjung
Pre condition	Sistem menyediakan *form* pencarian
Post condition	Sistem menampilkan hasil pencarian
Tujuan	Mencari data artikel yang ada di system
Skenario	Pengguna/pengunjung memasukan *keyword* pencarian di *form* pencarian

B. *Activity Diagram*

Berdasarkan *usecase diagram* yang telah digambarkan sebelumnya, maka dapat dimodelkan alur kerja (*workflow*) setiap proses dalam bentuk *activity diagram* untuk menggambarkan urutan aktifitas dalam tiap – tiap proses yang ada.

1. Pengelola Sistem

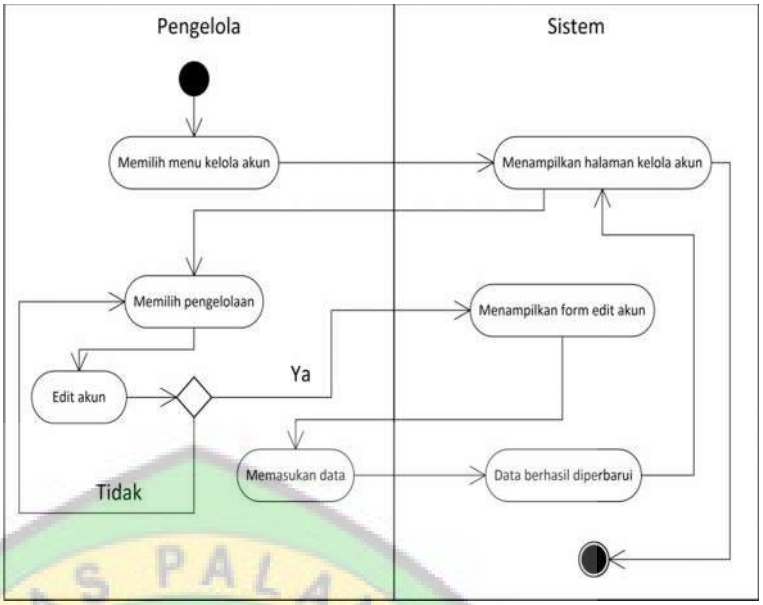
1) Login



Gambar 3.7 ***Activity Diagram Login***

Pengelola memasukan kredensial *login* berupa *email* dan *password*. Kredensial tersebut kemudian divalidasi oleh system, jika kredensial tersebut benar dan terdaftar di dalam system maka pengelola akan dialihkan ke halaman beranda pengelola. Namun jika kredensial yang dimasukan salah atau tidak terdaftar di dalam system makan pengelola akan dialihkan kembali ke halaman *login.*

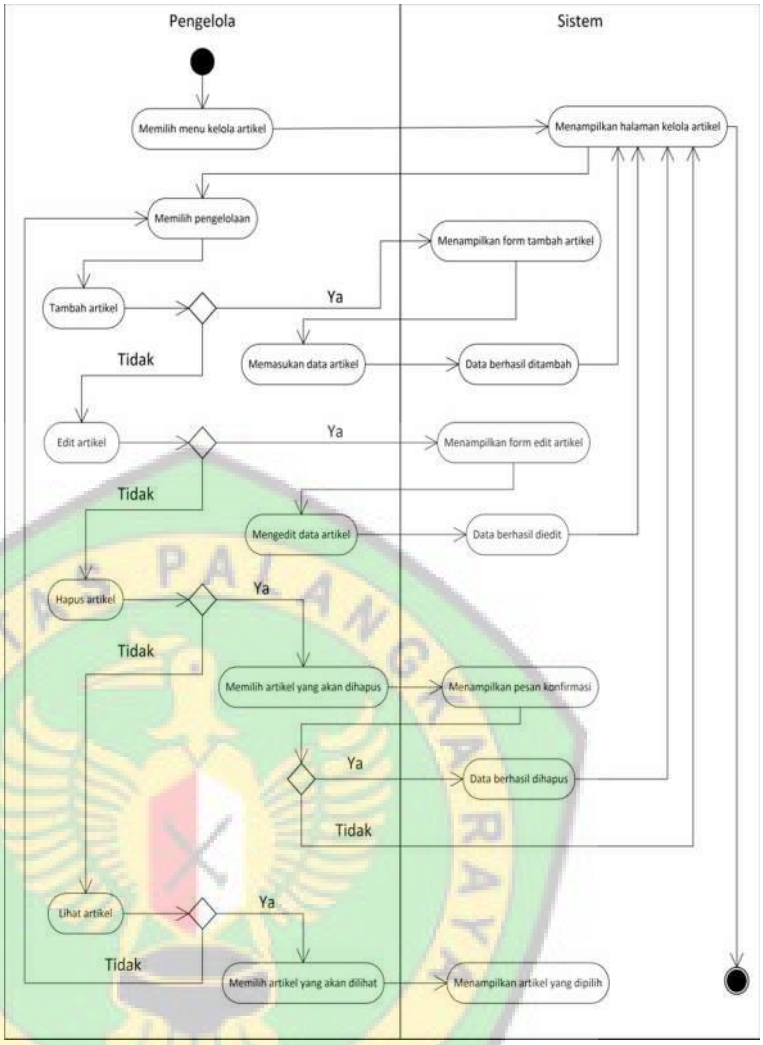
2) Kelola Akun



Gambar 3.8 *Activity Diagram* Kelola Akun

Pengelola memilih menu kelola akun, system kemudian menampilkan halaman kelola akun. Di halaman ini pengelola dapat membuka *form* untuk mengubah data akunnya.

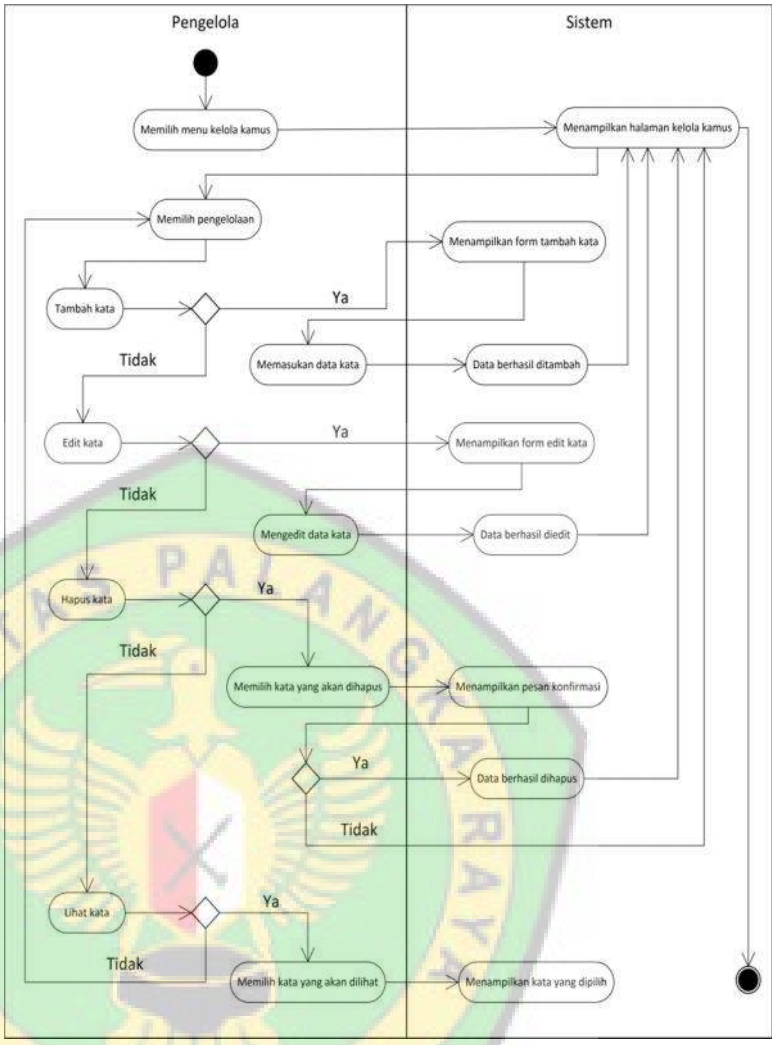
3) Kelola Artikel



Gambar 3.9 *Activity Diagram* Kelola Artikel

Pengelola memilih menu kelola artikel lalu kemudian system akan menampilkan halaman kelola artikel. Di halaman ini pengelola dapat memilih operasi tambah artikel baru, mengedit, menghapus, dan melihat artikel yang sudah ada.

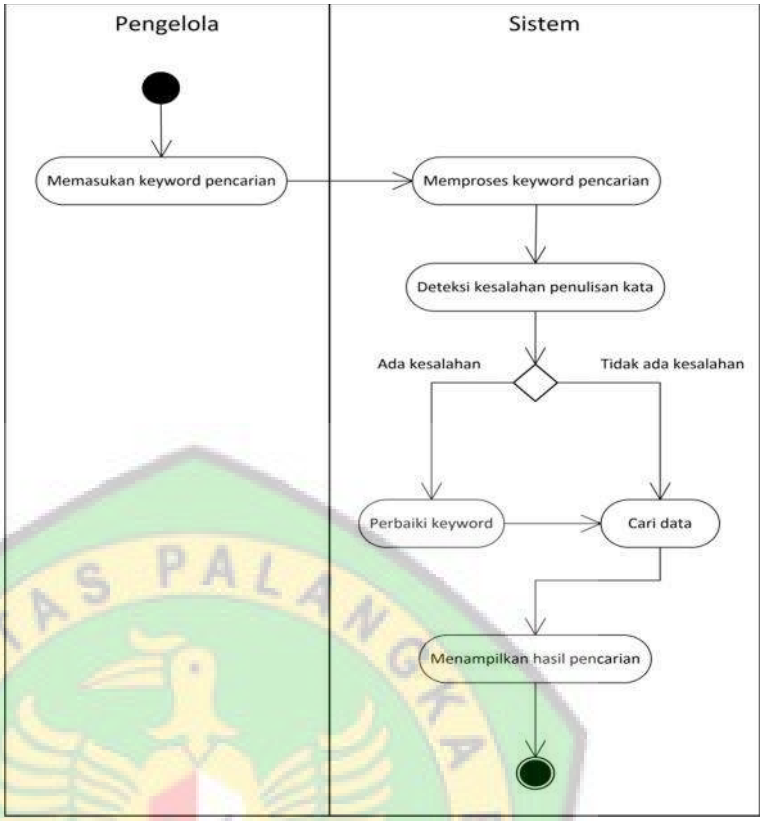
4) Kelola Kamus



Gambar 3.10 *Activity Diagram* Kelola Kamus

Pengelola memilih menu kelola kamus lalu kemudian system akan menampilkan halaman kelola kamus. Di halaman ini pengelola dapat memilih operasi tambah kata baru, mengedit, menghapus, dan melihat kata yang sudah ada.

5) Cari Artikel

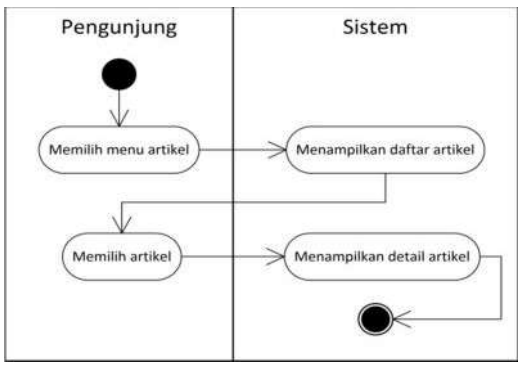


Gambar 3.11 *Activity Diagram* Cari Artikel

Pengelola memasukan *keyword* pencarian yang diinginkan kemudian system akan menampilkan hasil pencarian yang relevan terhadap *keyword*.

2. Pengguna/pengunjung Sistem

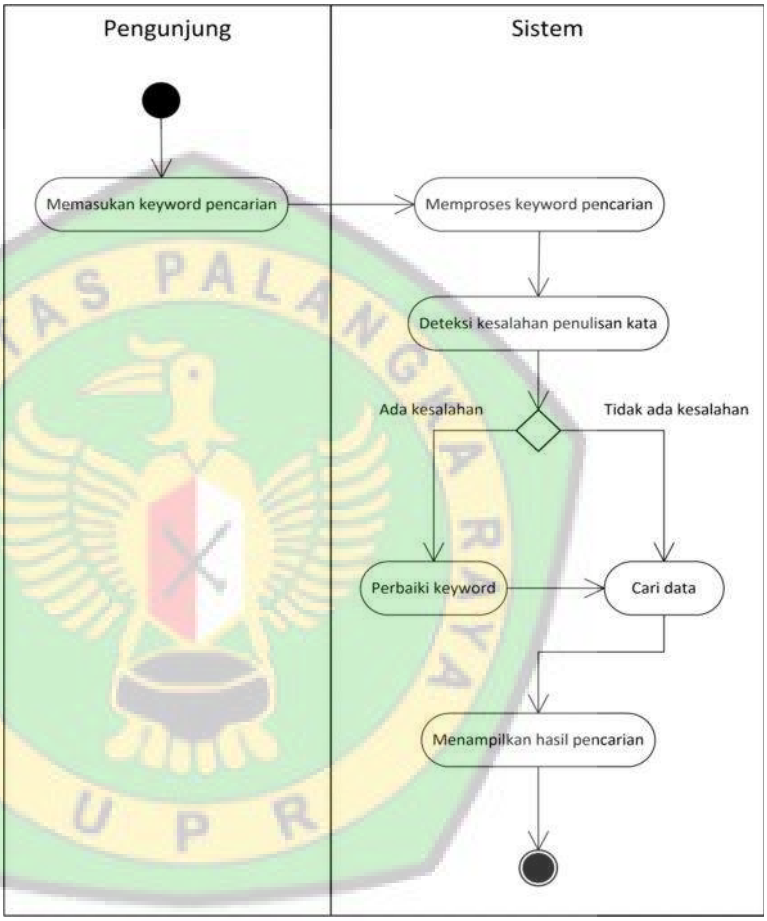
1) Lihat Artikel



Gambar 3.12 *Activity Diagram* Lihat Artikel

Pengguna/pengunjung memilih menu artikel, system kemudian menampilkan halaman berisi daftar artikel yang ada di system. Pengguna/pengunjung kemudian memilih artikel yang ingin dilihat.

2) Cari Artikel



Gambar 3.13 *Activity Diagram* Cari Artikel (Pengguna/pengunjung)

Pengguna/pengunjung memasukan *keyword* pencarian yang diinginkan kemudian system akan menampilkan hasil pencarian yang relevan terhadap *keyword*.

C. *Class Diagram*

1. Daftar Proses

Berikut ini adalah daftar proses – proses yang terdapat di dalam system berikut dengan keterangannya.

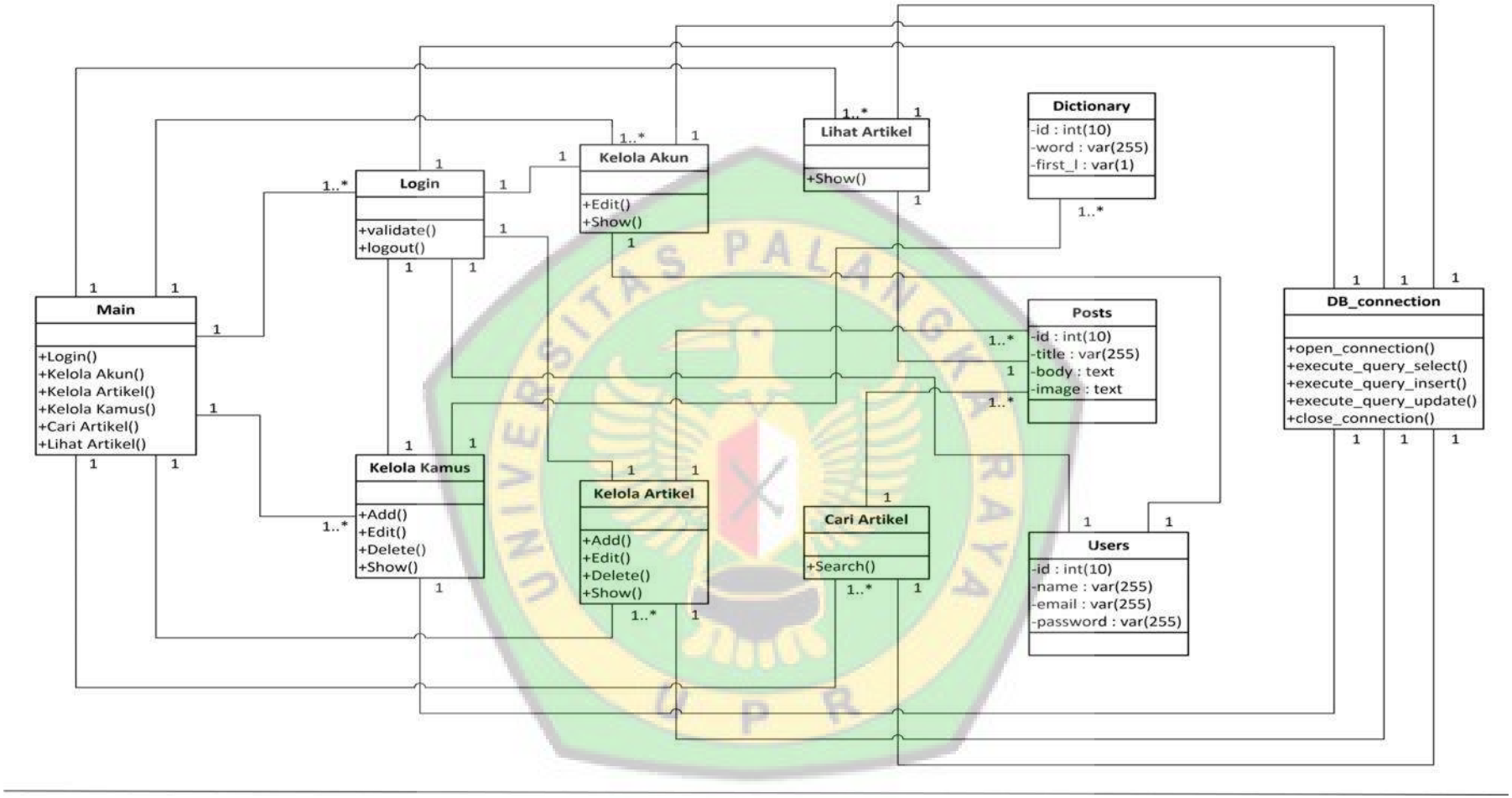
Tabel 3.11 Daftar Proses

No.	Nama Proses	Keterangan
1.	Login	Proses *login* ke system dengan hak akses penuh oleh pengelola
2.	Kelola Akun	Proses mengelola data akun milik pengelola yang bersangkutan
3.	Kelola Artikel	Proses mengelola data artikel yang ada di system
4.	Kelola Kamus	Proses mengelola kata – kata yang disimpan sebagai kamus di system
5.	Cari Artikel	Proses pencarian data artikel
6.	Lihat Artikel	Proses melihat data artikel yang dipilih

Berikut ini merupakan keterangan tentang table *database* yang digunakan pada system untuk menyimpan data yang berhubungan dengan proses – proses yang ada.

Tabel 3.12 Keterangan Tabel *Database*

No.	Nama Tabel	Keterangan
1.	*Users*	Menyimpan data akun pengelola
2.	*Posts*	Menyimpan data artikel
3.	*Dictionary*	Menyimpan data kata – kata pembanding



Gambar 3.14 *Class Diagram* Sistem

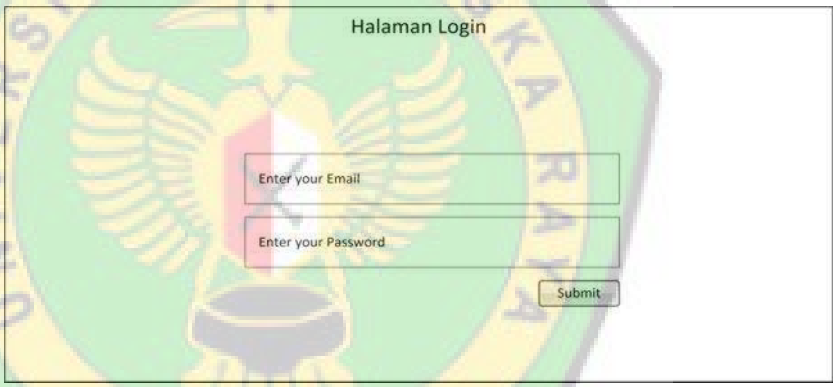
3.3. Desain User Interface

Setelah tahap analisis dan desain *UML*, maka tahap berikutnya adalah mendesain *user interface* system yang akan dibangun. Desain ini akan memberikan gambaran bagaimana system ini akan terlihat nantinya setelah diselesaikan.

A. Bagian Pengelola

1. Halaman Login

Halaman ini menyediakan *form login* untuk pengelola. Di *form* pada halaman ini pengelola memasukan kredensial *login* berupa *email* dan *password*.



Gambar 3.15 Desain Halaman Login

2. Halaman Beranda

Gambar 3.16 Desain Halaman Beranda Pengelola

Halaman ini hanya akan menampilkan deskripsi tentang penelitian yang dikerjakan dan sebuah *form* pencarian data artikel.

3. Halaman Kelola Artikel

Gambar 3.17 Desain Halaman Kelola Artikel

Halaman ini berisi daftar artikel yang ada di dalam system dan juga pilihan menu untuk menambah artikel baru, melihat, mengedit dan menghapus artikel yang sudah ada.

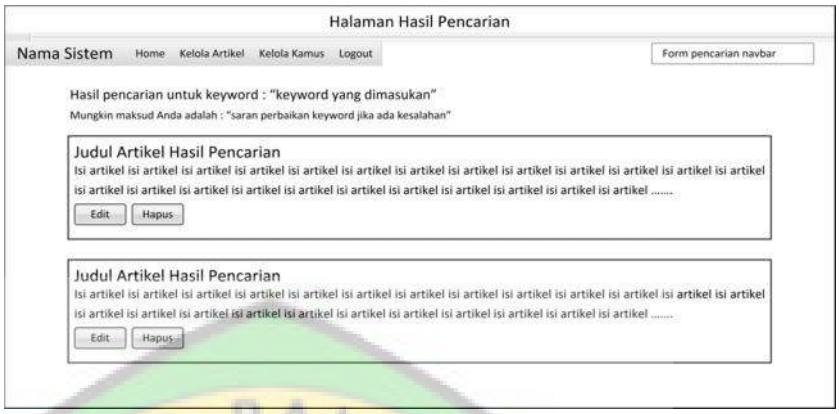
4. Halaman Kelola Kamus

Gambar 3.18 Desain Halaman Kelola Kata

Halaman ini berisi daftar kata yang ada di dalam system dikelompokan berdasarkan huruf awalan kata. Halaman ini juga

menyediakan pilihan menu untuk menambah kata baru, melihat, mengedit dan menghapus kata yang sudah ada.

5. Halaman Hasil Pencarian



Gambar 3.19 Desain Halaman Hasil Pencarian

Halaman ini akan menampilkan hasil pencarian data artikel yang dilakukan oleh pengelola.

6. Halaman Lihat Artikel

Gambar 3.20 Desain Halaman Lihat Artikel

Halaman ini akan menampilkan *detail* artikel yang ingin dilihat.

B. Bagian Pengguna/Pengunjung

1. Halaman Beranda

Gambar 3.21 Desain Halaman Beranda Pengunjung

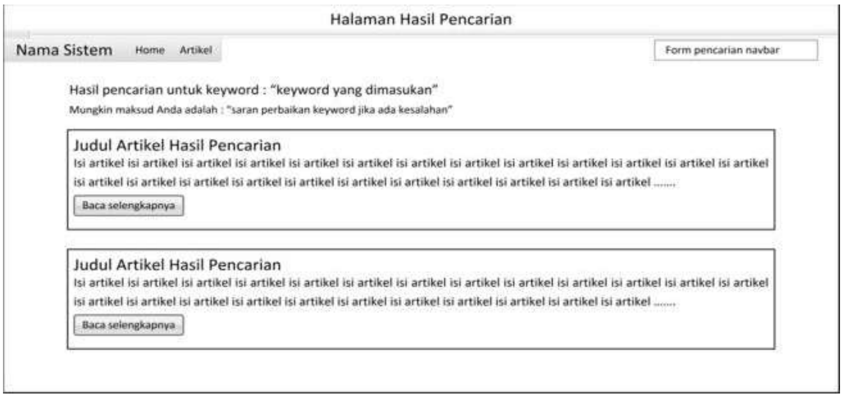
Halaman ini hanya akan menampilkan deskripsi tentang penelitian yang dikerjakan dan sebuah *form* pencarian data artikel.

2. Halaman Artikel

Gambar 3.22 Desain Halaman Artikel

Halaman ini akan menyediakan daftar artikel yang ada di dalam system.

3. Halaman Hasil Pencarian



Gambar 3.23 Desain Halaman Hasil Pencarian

Halaman ini akan menampilkan hasil pencarian data artikel yang dilakukan oleh pengelola.

4. Halaman Lihat Artikel

Gambar 3.24 Desain Halaman Lihat Artikel

Halaman ini akan menampilkan *detail* artikel yang ingin dilihat.

BAB V

PENUTUP

5.1. Kesimpulan

Berdasarkan hasil serangkaian uji coba yang telah dilakukan pada bab sebelumnya, dapat disimpulkan bahwa algoritma *Damerau-levenshtein distance* untuk memperbaiki kesalahan penulisan kata dalam *keyword* pencarian mampu memberikan nilai rata – rata 91.24% untuk *precision* dan 89.58% untuk *recall* berdasarkan pengujian yang telah dilakukan. Meski begitu algoritma ini dirasa tidak cocok jika digunakan untuk mengoreksi kata dalam sebuah frasa yang memiliki makna atau konteks seperti misalnya *keyword* pencarian, karena kemungkinan keadaan *false positive* yang tinggi sehingga kadang kata saran perbaikan yang muncul tidak sesuai dengan makna atau konteks dari *keyword* yang diperbaiki.

Dari hasil pengujian juga terlihat bahwa *keyword* yang mengandung kata *typo* sangat berpengaruh pada hasil pencarian yang didapatkan, seperti pada hasil pengujian di mana rata – rata hasil pencarian artikel yang didapatkan sebelum *typo* pada *keyword* diperbaiki selalu tidak ditemukan karena semua kata *typo* pada *keyword* tidak memiliki kecocokan dengan data *searchable* yang ada. Setelah diperbaiki oleh sistem baru didapatkan hasil pencarian yang relevan terhadap *keyword* dengan rata – rata nilai *precision* sebanyak 0.80 dan nilai *recall* sebanyak 0.98.

5.2. Saran

Adapun saran yang dapat diberikan untuk pengembangan implementasi algoritma *Damerau-levenshtein distance* untuk memperbaiki kesalahan penulisan *keyword* pencarian yaitu menambahkan konteks atau makna dari *keyword* yang sedang diperbaiki sebagai bahan pertimbangan untuk memilih kata koreksi yang tepat sehingga mengurangi kemungkinan keadaan *false positive*.

DAFTAR PUSTAKA

Ema Utami, Sukrisno. 2005. *10 Langkah Mudah Memahami Logika Algoritma Menggunakan Bahasa C/C++ di GNU/Linux, ISSN 979-763-020-X*. Yogyakarta. Penerbit: Andi Offset.

Expertsystem, 2016. *Natural Language Processing and Text Mining.* [online] Tersedia di: <http://www.expertsystem.com/natural-language-processing-and-text-mining> [Diakses 20 September 2017]

Feldman, Ronen dan Sanger, James. 2006. *The Text Mining Handbook : Advanced Approaches in Analyzing Unstructured Data*. New York. Penerbit : Cambridge University Press

Jugiyanto. 2008. *Metodologi Penelitian Sistem Informasi.* Yogyakarta. Penerbit : ANDI

Manning, C.D., Raghavan, P. & Schütze, H. 2009. *An Introduction to Information Retrieval* Online Edi., Cambridge University Press. Available at: http://www.informationretrieval.org/.

Mishra, R. & Kaur, N. 2013. A Survey of Spelling Error Detection and Correction Techniques. , 4, pp.372–374.

Mutammimah, Henry Sujaini, Rudy Dwi Nyoto. 2016. Analisis Perbandingan Metode Spelling Corrector Peter Norvig dan Spelling Corrector BK-Trees Pada Kata Berbahasa Indonesia. 3(4), pp.30-35.

Nugroho, Adi.. 2009. *Rekayasa Perangkat Lunak Menggunakan UML dan Java*. Yogyakarta. Penerbit : ANDI

Rossa A.S dan M. Shalahuddin. 2013. *Rekayasa perangkat lunak : terstruktur dan Berorientasi Objek*. Bandung. Penerbit : Informatika

Setiadi, Iskandar. 2013. Damerau-Levenshtein Algorithm and Bayes Theorem for Spell Checker Optimization

Sutisna, U. & Adisantoso, J. 2010. Koreksi Ejaan Query Bahasa Indonesia Menggunakan Algoritme Damerau Levenshtein. , 15(2), pp.25–29.

Tusty Nadia Maghfira, Imam Cholissodin, Agus Wahyu Widodo. 2017. Deteksi Kesalahan Ejaan dan Penentuan Rekomendasi Koreksi Kata yang Tepat Pada Dokumen Jurnal JTIIK Menggunakan Dictionary Lookup dan Damerau-Levenshtein Distance. 1(6), pp.498-506.

Wibowo, Ari. 2012. Pengujian Kerelevanan Sistem Temu Kembali Informasi. , 6(2), pp.10–14.