PENILAIAN JAWABAN ESAI BERDASARKAN PEDOMAN PENSKORAN MENGGUNAKAN *LONGEST COMMON SUBSEQUENCE* DAN *COSINE SIMILARITY*

SEMINAR HASIL

PROGRAM MAGISTER TEKNIK ELEKTRO
MINAT SISTEM KOMUNIKASI DAN INFORMATIKA

Diajukan untuk memenuhi persyaratan
Memperoleh gelar magister teknik

MOHAMMMAD NUR CHOLIS
NIM. 146060300111001

UNIVERSITAS BRAWIJAYA
FAKULTAS TEKNIK
M A L A N G
2018

LEMBAR PENGESAHAN

PENILAIAN JAWABAN ESAI BERDASARKAN PEDOMAN PENSKORAN MENGGUNAKAN *LONGEST COMMON SUBSEQUENCE* DAN *COSINE SIMILARITY*

SEMINAR HASIL

PROGRAM MAGISTER TEKNIK ELEKTRO
MINAT SISTEM KOMUNIKASI DAN INFORMATIKA

Diajukan untuk memenuhi persyaratan
Memperoleh gelar magister teknik

MOHAMMMAD NUR CHOLIS
NIM. 146060300111001

Menyetujui

Komisi Pembimbing

Ketua,	Anggota,
Dr. Ir. Erni Yudaningtyas, M.T.	Dr. Ir. Muhammad Aswin, M.T.
NIP. 19650913 199002 2 001	NIP. 19640626 199002 1 001

Mengetahui,

Ketua Program Studi Magister Teknik Elektro

Dr. Eng. Panca Mudjirahardjo. S.T., M.T.
NIP. 19700329 200012 1 001

RINGKASAN

Mohammad Nur Cholis, Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya, Juli 2019, *Penilaian Jawaban Esai Berdasarkan Pedoman Penskoran menggunakan Longest Common Subsequence dan Cosine Similarity*, Dosen Pembimbing: Erni Yudaningtyas dan Muhammad Aswin.

Evaluasi merupakan salah satu hal penting yang harus dilakukan pada proses pembelajaran. Evaluasi dalam proses pembelajaran salah satunya dapat dilakukan dengan mengadakan ujian. Salah satu ujian yang sering digunakan untuk melakukan evaluasi pembelajaran adalah ujian esai. Alasan digunakannya jenis ujian esai adalah kemudahan dalam penyusunan soal dan memiliki kelebihan dalam melatih pelajar menjawab soal dengan pendapat dan pengetahuannya sendiri.

Jawaban untuk satu soal ujian esai bisa bervariasi yang menyebabkan sulitnya melakukan penilaian. Sifat subjektifitas dalam melakukan penilaian juga mengakibatkan ketidakadilan dan mempengaruhi hasil penilaian jawaban ujian esai. Membuat pedoman penskoran yang akan menjadi acuan dalam melakukan penilaian jawaban dapat menjadi solusi untuk mengatasi hal tersebut Pedoman penskoran dapat dikatakan sebagai strategi pemberian skor terhadap jawaban yang akan muncul. Selain penggunaan pedoman penskoran, penggunaan teknologi berupa sistem yang dapat melakukan penilaian secara otomatis juga akan membantu untuk menilai jawaban ujian esai.

Dalam membuat sistem penilaian esai secara otomatis dibutuhkan proses pengekstrakan khususnya pada jawaban pelajar supaya menjadi lebih terstruktur. Proses pengekstrakan jawaban dilakukan dengan *text mining* yang dalam prosesnya dibagi dalam dua tahap yaitu *text preprocessing* dan *feature selection.* Pada tahap *text preprocessing* dilakukan proses untuk mempersiapkan data. Tahap *text preprocessing* terdiri atas proses mengubah semua huruf menjadi huruf kecil dan proses mengurai (memenggal) kalimat-kalimat menjadi daftar kata (*tokenizing).* Tahap *feature selection* adalah tahap setelah tahap *text preprocessing.* Tahap *feature selection* bertujuan mengurangi dimensi dari daftar kata dengan menghapus kata-kata yang tidak penting. Pada penelitian ini, ditambahkan proses *spelling correction* dan *synonym recognition* pada tahap *feature selection*. Proses *spelling correction* digunakan untuk memperbaiki kesalahan pengetikan. Proses *synonym recognition* digunakan untuk pendeteksian sinonim.

Hasil proses pengekstrakan kemudian dihitung kemiripan jawaban dengan setiap kunci jawaban pada pedoman penskoran. Pada proses menghitung kemiripan ini, langkah pertama yang dilakukan adalah mencari kombinasi terbaik antara penggalan jawaban dengan kunci jawaban yang ada pada pedoman penskoran menggunakan algoritma *Longest Common Subsequence* (LCS). Kombinasi terbaik antara penggalan jawaban dengan kunci jawaban kemudian dihitung untuk mencari nilai kemiripannya menggunakan *cosine similarity*. Nilai kemiripan kemudian dikalikan dengan skor pada kunci jawaban. Penjumlahan dari semua hasil kali nilai kemiripan dan skor yang didapatkan tersebut yang menjadi rekomendasi nilai dari sistem.

Pengujian untuk mengetahui tingkat akurasi sistem dilakukan dengan menggunakan data ujian mata pelajaran bahasa indonesia, seni budaya dan IPA dengan jumlah soal masing-masing ujian adalah 5 soal dan diikuti oleh 24 pelajar. Hasil rekomendasi nilai sistem penilaian esai berdasarkan pedoman penskoran menggunakan *synonym recognition* serta *spelling correction* menunjukkan tingkat akurasi sebesar 93,61 % dengan *Root Mean Square Error* (RMSE) sebesar 0,85, sedangkan tanpa pedoman penskoran memiliki akurasi 51,39 % dengan RMSE sebesar 4,47. Penggunaan *synonym recognition* serta *spelling correction* dapat meningkatkan akurasi rata-rata sebesar 35,28 % dan memperkecil RMSE rata-rata sebesar 3,80.

Kata kunci : Penilaian Esai, Pedoman Penskoran, *Longest Common Subsequence* (LCS), *Cosine Similarity.*

SUMMARY

Mohammad Nur Cholis, Electrical Engineering Department, Faculty of Engineering, Brawijaya University, Juli 2019, *Essay Answer Assessment Based on scoring guidelines using Longest Common Subsequence and Cosine Similarity*, Academic Supervisor : Erni Yudaningtyas and Muhammad Aswin.

Evaluation is one of the important things that must be done in the learning process. One of the evaluations in the learning process can be done by holding an exam. One test that is often used to conduct learning evaluations is essay examinations. The reason for using this type of essay exam is the ease with which questions are prepared but has advantages in training students to answer questions with their own opinions and knowledge.

Answers to one essay exam question can vary which makes it difficult to make an assessment. The nature of subjectivity in making judgments also results in injustice and affects the results of the assessment of essay exam answers. Making scoring guidelines that will be a reference in evaluating answers can be a solution to overcome this problem. Scoring guidelines can be said as a strategy for scoring the answers that will appear. In addition to the use of scoring guidelines, the use of technology in the form of a system that can automatically assess will also help to assess the answers to essay exams.

In making an essay scoring system automatically an extraction process is needed especially in the student's answer so that it becomes more structured. The process of extracting answers is done with text mining which in the process is divided into two stages, namely text preprocessing and feature selection. In the text preprocessing stage, a process is prepared to prepare the data. The text preprocessing stage consists of the process of converting all letters into lowercase letters and the process of breaking down the sentences into tokenizing lists. The feature selection stage is the stage after the text preprocessing stage. The feature selection stage aims to reduce the dimensions of the word list by removing non-essential words. In this research added the process of spelling correction and synonym recognition at the feature selection stage. The spelling correction process is used to correct typing errors. The synonym recognition process is used to detect synonyms.

The results of the extraction process are then calculated for the similarity of answers with each answer key in the scoring guidelines. In the process of calculating this similarity, the first step taken is to find the best combination between the answers with the answer key in the scoring guidelines using the Longest Common Subsequence (LCS) algorithm. The best combination of fragmenting the answer with the answer key is then calculated to look for similarities using cosine similarity. The similarity value is then multiplied by the score on the answer key. The sum of all the results of the similarity value and the score obtained are recommendations for the value of the system.

Testing to determine the level of accuracy of the system is done by using test data on Indonesian subjects, arts and culture and science with the number of questions for each exam is 5 questions and followed by 24 students. The results of the essay rating system recommendations based on scoring guidelines using synonym recognition and spelling correction showed an accuracy of 93.61% and Root Mean Square Error (RMSE) of 0.85, while without scoring guidelines it had an accuracy of 51,39% with RMSE of 4,47 . The use of synonym recognition and spelling correction can increase the average accuracy by 35.28% and reduce the RMSE by an average of 3.80.

Keywords: Essay Assessment, Scoring Guidelines, Longest Common Subsequence (LCS), Cosine Similarity.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR ISTILAH

Case folding	:	Proses untuk merubah semua huruf besar pada teks dalam dokumen menjadi huruf kecil atau sebaliknya. Proses ini disebut juga *toLowerCase*
Confix stripping stemmer	:	Metode *stemming* pada bahasa indonesia yang diperkenalkan oleh Jelita Asian yang merupakan pengembangan dari metode s*temming* yang dibuat oleh Nazief dan Adriani (1996).
Cosine similarity	:	Metode yang digunakan untuk menghitung tingkat kesamaan (*similarity*) antara dua buah objek.
Spelling correction	:	Disebut juga pengoreksian ejaan adalah proses yang digunakan untuk mengoreksi kesalahan pengejaan
Derivation prefixes	:	Awalan-awalan yang dapat langsung diberikan pada kata dasar murni, atau pada kata dasar yang sudah mendapatkan penambahan sampai dengan 2 awalan. Termasuk di dalamnya adalah awalan yang dapat bermorfologi (“me-”, “be-”, “pe-”, dan “te-”) dan awalan yang tidak bermorfologi (“di-”, “ke-” dan “se-”).
Derivation suffixes	:	Kumpulan akhiran yang secara langsung dapat ditambahkan pada kata dasar. Termasuk di dalam tipe ini adalah akhiran “-i”, “-kan”, dan “-an”.
Document frequency	:	Banyaknya kalimat dimana suatu kata muncul.
Enhanced confix stripping stemmer	:	Algoritma *confix stripping stemmer* yang telah mendapatkan revisi dan tambahan
Infiks	:	Imbuhan sisipan kata
Inflection suffixes	:	Kelompok-kelompok akhiran yang tidak mengubah bentuk kata dasar. . Kelompok ini dapat dibagi menjadi dua : *Particle* dan *Possessive Pronoun*
Konfiks	:	Imbuhan di awalan dan akhir kata

Levenshtein distance	:	Sering disebut juga sebagai *edit distance*, adalah suatu pengukuran (*metrik*) yang dihasilkan melalui perhitungan jumlah perbedaan yang terdapat pada dua *string*.
Longest common subsequences	:	Metode untuk mencari *sequence* terpanjang yang sama
Particle	:	Imbuhan yang berupa "-lah", "-kah", "-tah", dan "-pun".
Pedoman penskoran	:	Pedoman yang digunakan untuk menentukan skor hasil pekerjaan pelajar
Possessive pronoun	:	Kata ganti kepunyaan, termasuk di dalamnya adalah "-ku", "-mu", dan "-nya".
Prefiks	:	Imbuhan di awal kata
Recoding	:	Proses yang dilakukan dengan menambahkan karakter di awal kata proses *stemming* yang dilakukan berdasarkan aturan Tabel 2.2 (dengan revisi Tabel 2.3)
Root mean square error	:	Perhitungan yang bertujuan untuk menggambarkan besarnya selisih suatu nilai dari sistem terhadap nilai sebenarnya
Feature selection	:	Tahap mengurangi dimensi dari kumpulan teks dengan menghapus kata-kata yang tidak penting dalam dokumen
Stemming	:	Proses pemetaan dan penguraian berbagai bentuk varian dari suatu kata menjadi bentuk kata dasar (umum)
Stopword	:	Daftar kata yang sering muncul atau kata-kata yang tidak memiliki makna penting dalam dokumen (informasi) dan hanya berfungsi sebagai penanda gramatikal
Stopword removal	:	Proses untuk membuang *token* yang termasuk dalam daftar *stopword*
Subsequence	:	Sebuah *sequence* yang diturunkan dari *sequence* lain dengan menghapus beberapa elemen tanpa mengganti urutan dari sisa *element* tersebut
Sufiks	:	Imbuhan di akhir kata

Synonym recognition	:	Teknik yang digunakan untuk mengenali kata dengan penulisan berbeda namun memiliki makna yang sama
Term frequency	:	Frekuensi kemunculan kata pada kalimat.
Term frequency-inverse document frequency	:	Cara pemberian bobot hubungan suatu kata (*term*) terhadap dokumen.
Text mining	:	Seperangkat teknik dan metode yang digunakan untuk pemrosesan otomatis pada data teks bahasa alami yang tersedia dalam jumlah yang cukup besar dalam bentuk file komputer dengan tujuan mengekstraksi dan menyusun konten untuk tujuan analisis cepat, penemuan data tersembunyi, atau pengambilan keputusan otomatis
Text Preprocessing	:	Tahap yang mencangkup semua proses untuk mempersiapkan data
Tokenizing	:	Proses penguraian kalimat-kalimat menjadi kata-kata dan menghilangkan *delimiter* seperti tanda titik (.), koma (,), spasi dan karakter angka yang ada
Ujian esai	:	Ujian dimana pertanyaan yang diajukan menuntut pelajar untuk menjawab dengan cara menguraikan, menjelaskan, mendiskusikan, membandingkan, memberikan alasan, dan bentuk sejenis menggunakan kata-kata serta bahasa sendiri

DAFTAR SINGKATAN

A	:	Vokal atau konsonan
C	:	Konsonan
CS	:	*Confix stripping stemmer*
DF	:	*Document frequency*
DP	:	*Derivation Prefixes*
DS	:	*Derivation Suffixes*
ECS	:	*Enhanced confix stripping stemmer*
IDF	:	*Inverse document frequency*
LCS	:	*Longest Common Subsequences*
P	:	*Particle*
PP	:	*Possessive Pronoun*
RMSE	:	*Root mean square error*
TF	:	*Term frequency*
TF-IDF	:	*Term frequency-inverse document frequency*
V	:	Vokal

BAB I
PENDAHULUAN

1.1 Latar Belakang

Evaluasi merupakan salah satu hal penting yang harus dilakukan pada proses pembelajaran. Dengan mengadakan evaluasi, seorang pengajar dapat mengetahui keberhasilan proses pembelajaran serta dapat menjadi tolok ukur untuk memperbaiki dan menyempurnakan proses pembelajaran (Arifin, 2017). Oleh karena itu seorang pengajar harus memiliki kompenensi dalam melakukan evaluasi pembelajaran. Evaluasi pembelajaran salah satunya dapat dilakukan dengan mengadakan ujian atau tes.

Ujian pilihan ganda (objektif) dan ujian esai dapat dijadikan pilihan dalam melakukan evaluasi proses pembelajaran. Pada ujian pilihan ganda, pelajar diharuskan memilih jawaban yang paling benar dari pilihan yang telah disediakan. Berspekulasi dalam memilih jawaban yang benar dimungkinkan ketika ujian dilakukan dalam bentuk pilihan ganda. Sedangkan pada ujian esai pilihan jawaban tidak ada. Pelajar diharuskan menjawab dalam bentuk tulisan dengan bahasanya sendiri.

Ujian esai dapat melatih pelajar dalam mengorganisasikan pikirannya, melatih pelajar dalam menyampaikan ide atau pendapat menggunakan kata-katanya sendiri dalam bentuk tulisan dan melatih penalaran (Sary, 2015). Ujian esai dapat mengatasi kelemahan daya ukur soal objektif dan cocok digunakan untuk mengukur hasil belajar yang memiliki level kognisi yang lebih tinggi dan bersifat kompleks, serta dapat mengetahui sejauh mana tingkat penguasaan pelajar terhadap materi (Astiti, 2017). Menyusun soal ujian esai lebih mudah dilakukan. Hal-hal itulah yang membuat ujian esai memiliki kelebihan dari ujian pilihan ganda.

Akan tetapi menilai jawaban ujian esai lebih sulit dari pada ujian pilihan ganda. Penilaian jawaban esai tidak cukup dengan melihat panjang pendeknya jawaban. Menurut Astiti (2017) penggunaan ujian dalam bentuk esai akan menimbulkan bervariasinya jawaban yang diberikan, memberikan peluang untuk melakukan penilaian yang bersifat subjektif baik karena penilai yang berbeda atau situasi yang

berbeda dan membutuhkan waktu koreksi yang lama. Bervariasinya jawaban, sifat subjektif dalam menilai dan lamanya pengoreksian akan mempengaruhi keakurasian dalam melakukan penilaian akan menimbulkan ketidakadilan dalam menilai. Padahal berdasarkan Permendikbud No. 66 Tahun 2013 tentang Standar Penilaian, penilaian harus dilakukan dengan objektif dan akuntabel, yaitu berbasis pada standar dan tidak dipengaruhi oleh faktor subjektif penilai serta harus dapat dipertanggungjawabkan.

Untuk menimalisir ketidakadilan dan sifat subjektif dalam menilai jawaban esai tersebut harus dibuat standar penilaian dalam pemberian skor pada jawaban. Pada saat soal dibuat pengajar juga harus memikirkan strategi pemberian skor terhadap jawaban yang akan muncul, hal itu dapat dilakukan dengan menyusun pedoman penskoran (Sary, 2015). Pedoman penskoran ini sangat penting disiapkan khususnya untuk ujian dalam bentuk esai. Dengan adanya pedoman penskoran, akan memudahkan penilai dalam membandingkan jawaban dengan kunci jawaban yang ideal sesuai kemiripan dan tingkat kesempurnaan dan kelengkapan jawaban pelajar (Sary, 2015).

Disisi lain saat ini untuk melaksanakan ujian baik ujian pilihan ganda atau ujian esai dengan memanfaatkan teknologi informasi jauh lebih mudah. Telah banyak sistem manajemen pembelajaran yang dibuat salahsatunya untuk tujuan tersebut. Sistem manajemen pembelajaran tersebut sudah memiliki fasilitas untuk menyimpan kumpulan soal beserta kunci jawabannya, mengacaknya, mengatur waktu pelaksanaan ujian secara otomatis, dan mengatur lamanya pelaksanaan ujian. Sistem tersebut juga telah dapat menilai jawaban dalam bentuk ujian pilihan ganda akan tetapi belum dapat menilai jawaban ujian esai. Untuk itu perlu dikembangkan sistem yang dapat melakukan penilaian ujian esai, terutama yang menggunakan pedoman penskoran dalam melakukan penilaian ujian esainya.

Dalam mengembangkan sistem penilaian ujian esai tersebut perlu adanya proses pengekstrakan. Proses pengekstrakan diperlukan karena jawaban esai dari pelajar bisa bervariasi dan tidak terstuktur. Proses pengekstrakan dapat menggunakan *text mining*. *Text mining* adalah proses mengekstrak informasi yang berguna dari sumber data yang berupa dokumen bahasa alami dimana data dalam dokumen tersebut tidak terstuktur (Fledman, 2007).

Setelah proses pengekstrakan, proses lain yang perlu dilakukan adalah proses menghitungkan kemiripan jawaban pelajar dengan kunci jawaban. Dimana hasil dari perhitungan kemiripan ini yang akan dikonversi menjadi nilai ujian. Metode yang dapat digunakan untuk menghitung kemiripan dua data tersebut adalah *cosine similarity* dan *longest common subsequence* (LCS). *Cosine similarity* menghitung tingkat kemiripan antara dua data berdasarkan nilai *cosinus* sudut dari perkalian dua data yang dibandingkan (Saputra, 2013). Sedangkan *longest common subsequence* (LCS) adalah metode untuk menghitung relasi berurutan yang paling panjang (Saadah, 2013).

Penggunaan *text mining* dan *cosine similarity* untuk menyelesaikan masalah penilaian jawaban esai berdasarkan penelitian yang dilakukan oleh Fitri, R. dan Asyikin, A. N. pada tahun 2015 menghasilkan akurasi yang baik yaitu 89,48%. Hasil tersebut berdasarkan perbandingan antara nilai manual dengan dari nilai sistem. Penambahan proses memperbaiki kesalahan pengetikan dan pendeteksian sinonim mungkin dapat meningkatkan akurasi. Sebagaimana yang disebutkan oleh Fitri, R. & Asyikin, A. N. bahwa kesalahan penilaian terjadi karena adanya kesalahan ketikan dan Darsono, A. D. B. dan Wijono, S. H. (2014) yang juga menggunakan *cosine similarity* pada kasus yang sama menyimpulkan bahwa perbedaan nilai yang sangat mencolok antara nilai sistem dan koreksi manual disebabkan karena permasalahan makna kata atau sinonim.

Cosine similarity dalam menghitung kemiripan didasarkan pada keberadaan dan jumlah kata yang terdapat pada dua data dengan memperhatikan nilai *cosinus. Cosine similarity* tidak mempertimbangkan penempatan kata dalam teks yang dianalisis (Crocetti, 2015). *Cosine similarity* akan menghasilkan nilai kemiripan 1 atau sama persis pada dua kalimat yang memiliki variasi dan jumlah kata yang sama tetapi memiliki susunan penempatan kata yang berbeda. Untuk itu perlu adanya perhitungan kemiripan yang memperhatikan urutan pembentuk kalimat. Metode yang menghitung kemiripan berdasarkan relasi berurutan yang paling panjang adalah *longest common subsequence* (LCS). Oleh karena itu, dengan mengkombinasikan *longest common subsequence* (LCS) dan *cosine similarity* mungkin akan menghasilkan hitungan kemiripan yang lebih baik.

Berdasarkan beberapa hal diatas, penilaian jawaban esai dalam penelitian ini akan menggunakan pedoman penskoran yang berisi kunci-kunci jawaban sebagai pembanding dengan jawaban pelajar. Proses mengekstrak jawaban pelajar menggunakan *text mining*. Sebelum proses perhitungan kemiripan akan dilakukan proses pengenalan sinonim dan pengoreksi kesalahan pengetikan. Proses penentuan hasil nilai ujian diperoleh dari pengecek kemiripan antara jawaban pelajar dan kunci jawaban yang dilakukan dengan *cosine similarity* dan *longest common subsequences* (LSC). Dengan proses-proses tersebut diharapkan akan memperoleh hasil rekomendasi nilai yang lebih akurat.

1.2 Rumusan Masalah

Berdasarkan latar belakang diatas, maka rumusan masalah dalam penelitian ini adalah

1. Bagaimana sistem melakukan penilaian jawaban esai yang mengacu pada pedoman penskoran dengan menggunakan kombinasi metode *longest common subsequence* (LCS) dan *cosine similarity*,
2. Bagaimana tingkat akurasi dan nilai error berupa *root mean square error* (RMSE) yang didapatkan sistem jika dibandingkan dengan penilaian jawaban secara manual,
3. Bagaimana pengaruh penambahan pengenalan sinonim (*synonym recognition*) dan pengoreksi ejaan (*spelling correction*) pada tingkat akurasi hasil penilaian.

1.3 Batasan Masalah

Untuk menghindari meluasnya pembahasan, maka batasan masalah dalam penelitian ini adalah :

1. Dirancang hanya untuk menilai jawaban esai ber-Bahasa Indonesia.
2. Penilaian hanya dilakukan pada jawaban berupa teks narasi.
3. *Input* jawaban yang akan dinilai berupa hasil ketikan pada aplikasi ujian online berbasis web yang dibuat dengan bahasa pemrograman PHP.
4. Kunci jawaban esai berupa pedoman penskoran dimana setiap bagian pada jawaban telah memiliki acuan menilaian.
5. Data penelitian berupa soal ujian, jawaban pelajar, dan kunci jawaban disimpan dalam database.

6. Database yang digunakan adalah MySQL.
7. Sistem penilaian ujian esai dibuat terpisah dengan sistem ujian. Sistem penilaian ujian esai dibuat berbasis desktop dengan bahasa pemrograman JAVA.
8. Data pendukung proses penilaian berupa daftar stopword, daftar kata dasar, dan daftar kata sinonim disimpan dalam file dengan tipe data *.txt.
9. Daftar kata *stopword* didapatkan dari jurnal Tala (2003) yang berjudul *A Study of Stemming Effects on Information Retrieval in Bahasa Indonesia*.
10. Daftar kata dasar dan sinonim didapatkan dari KBBI online dan kamusbesar.com.
11. Pertanyaan esai dibatasi hanya untuk pertanyaan "sebutkan".

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah sebagai berikut :

1. Untuk membuat sistem yang dapat melakukan penilaian jawaban esai yang mengacu pada pedoman penskoran dengan menggunakan kombinasi metode *longest common subsequence* (LCS) dan *cosine similarity*,
2. Untuk mengetahui tingkat akurasi dan RMSE yang didapatkan sistem jika dibandingkan dengan penilaian jawaban secara manual,
3. Untuk mengetahui pengaruh penambahan pengenalan sinonim dan pengoreksi ejaan terhadap tingkat akurasi hasil penilaian.

1.5 Manfaat Penelitian

Dengan penelitian ini diharapkan dapat memberikan manfaat sebagai berikut :

1. Secara akademik

Dengan penelitian ini diharapkan memberikan kontribusi dalam bentuk pengembangan ilmu di bidang *text mining* dengan cara mengkombinasikannya dengan *cosine similarity* dan *longest common subsequence* (LCS) serta menambahkan proses pengenalan sinonim (*synonym recognition*) dan pengoreksi ejaan (*spelling correction*), serta dapat memberikan kontribusi keilmuan bagi penelitian selanjutnya.

2. Secara aplikatif

Dengan adanya sistem penilaian jawaban esai diharapkan dapat memberikan kontribusi sehingga membantu pengajar dalam menilai jawaban ujian esai menjadi lebih mudah dan cepat, serta dapat mengurangi subjektifitas pada proses penilaian

BAB II
TINJAUAN PUSTAKA

2.1 Penelitian Relevan

Berikut ini adalah penelitian-penelitian yang pernah dilakukan dan berkaitan dengan penelitian ini :

Lahitani, A.R., Permanasari, A.E. & Setiawan, N.A. (2016) melakukan penelitian dengan menerapkan pembobotan metode *Term Frequency – Inverse Document Frequency* (tf-idf) dan *Cosine Similarity* untuk mengukur derajat kemiripan dokumen dalam studi kasus penilaian esai secara online. Pengujian dilakukan pada sejumlah dokumen esai ber-Bahasa Indonesia yang telah melalui tahap pra-pemrosesan. Pra-pemrosesan yang dilakukan adalah *case folding*, *tokenizing* yang termasuk didalamnya adalah melakukan *stemming* dengan Algoritma Nazief-Andriani, *filtering*, dan yang terakhir *indexing*. Hasil Pra-pemrosesan kemudian dibandingan dengan jawaban ahli dengan melakukan perhitungan bobot tf-idf dan menghitung derajat kemiripan dengan *cosine similarity*. Hasil dari penelitian ini berupa perangkingan bobot dokumen berdasarkan kemiripan dengan jawaban ahli. Pada penelitian ini diusulkan adanya konversi bobot kemiripan menjadi nilai skor.

Fitri R. dan Asyikin A. N. (2015) melakukan penelitian tentang penilaian esai otomatis dengan menggunakan metode *cosine similarity* pada ujian esai ber-Bahasa Inggris. Data jawaban siswa dan kunci jawaban yang akan dibandingkan kemiripannya sama-sama dilakukan pemrosesan teks. Dimana pemrosesan teks tersebut terdiri dari proses *tokenisasi*, membuang kata yang termasuk daftar *stopword*, dan terakhir *stemming*. Hasil dari pemrosesan teks kemudian dilakukan pembobotan menggunakan tf-idf. Kemudian dilakukan perhitungan derajat kemiripan dengan menggunakan *cosine similarity*. Berbeda dengan penelitian yang dilakukan oleh Lahitani, dkk, pada penelitian ini dilakukan konversi nilai kemiripan menjadi nilai ujian. Hasil penelitian ini menunjukkan kesesuaian nilai sistem dengan nilai yang diberikan oleh pengajar rata-rata 89,48 %. Kesalahan penilaian disebabkan

karena kesalahan pengetikan yang menyebabkan sistem akan menganggap salah karena secara susunan huruf berbeda.

Darsono, A. D. B., & Wijono, S. H. (2014) menggabungkan *k-nearest neighbo*r (K-NN) dan *latent semantic analysis* (LSA) untuk menambah akurasi penilaian jawaban esai. Pada penelitian ini, proses pertama dilakukan *preprocessing* yang meliputi proses *tokenisasi*, menghilangkan *stopword* dan melakukan *stemming*. Dokumen yang telah melewati proses *preprocessing* akan dilakukan klasifikasi menggunakan K-NN. Jika proses K-NN mengklasifikasikan benar, akan dilanjutkan dengan proses LSA. Terakhir dihitung kemiripan dengan *cosine similarity*. Hasil penelitian menunjukkan rata - rata akurasi untuk K-NN k=1 sebesar 85,13% dan k=3 sebesar 86.67%, rata - rata tingkat kesalahan nilai LSA untuk k=1 sebesar 14.02 % dan k = 3 sebesar 13.11% dan perbedaan nilai yang sangat mencolok antara nilai sistem dan koreksi manual disebabkan karena permasalahan makna kata atau sinomim. Saran dari peneliti untuk perbaikan adalah menambahkan proses untuk mendeteksi sinonim kata atau menambahkan kata kunci.

Mustapha, S. S.M.F.D., Idris, N. & Abdullah, R. (2005) melakukan pengelompokan pada kata hasil pra-pemrosesan kedalam kelas-kelas sesuai jenisnya. Kelas-kelas yang digunakan Antara lain #YEAR, #PERSON, #NAME, #ACTION, #EVENT, #FACTOR, #PLACE, #COUNTRY dan #STATE. Hasil pengelompokan pada kelas-kelas tersebut yang kemudian dilakukan perhitungan kesamaan. Peneliti menyimpulkan bahwa sistem penilaian ini dapat diimplementasikan pada mata pelajaran sejarah dan mungkin perlu penyesuaian pada mata pelajaran yang lain

Braddley, M.O., Fachrurrozi, M & Yusliani, N. (2017) meneliti tentang pengoreksian ejaan pada dokumen bahasa Indonesia. Sebelum dilakukan proses pengecekan kata pada dokumen, terlebih dahulu dilakukan proses pra-pengolahan yang terdiri dari *case folding* dan *tokenizing*. Hasil pra-pengolahan berupa daftar kata yang terdapat pada dokumen. Masing-masing kata tersebut kemudian dibandingkan dengan kata yang berada pada kamus, Jika kata tersebut tidak terdapat pada kamus akan dilakukan proses pengoreksian. Algoritma yang digunakan untuk pengoreksian adalah *levenshein distance*. Hasil dari penelitian ini berupa tabel saran perbaikan kata. Pada penelitian ini, pengujian dilakukan dengan 90 data yang terdiri dari 3

skenario yaitu penghapusan, penambahan dan pengubahan karakter. Hasil pengujian akurasi rata-rata sebesar 100% dan waktu 23 mili detik pada operasi penghapusan karakter, hasil 96% dan waktu 5 mili detik pada operasi pengubahan karakter dan hasil 93% dan waktu 88 mili detik pada operasi penambahan.

Ratna A. A. P., Sanjaya R., Wirianata T. & Purnamasari P. D. (2017) menambahkan fitur koreksi kesalahan pengetikan dan pustaka kata yang berisi daftar kata bersinonim dalam penelitian tentang sistem penilaian esai dengan *latent semantic analysis*. Dalam mengoreksi kesalahan pengetikan digunakan algoritma *jaro-winkler*. Dalam penelitian tersebut disimpulkan bahwa menggunakan algoritma koreksi otomatis terbukti bermanfaat untuk mengurangi kesalahan ketik dan menghasilkan akurasi sebesar 85.246%.

Mahalakshmi & Kavitha S. (2016) melakukan penelitian tentang pendeteksian plagiasi pada perangkat lunak. Pada penelitian tersebut metode yang digunakan untuk mendeteksi kesamaan string pada *code program* adalah *longest common subsequence* (LCS). Hasilnya menunjukkan bahwa metode LCS efektif dan efisien dalam mendeteksi plagiarisme perangkat lunak.

2.2 Ujian Esai

Ujian esai merupakan ujian dimana pertanyaan yang diajukan menuntut pelajar untuk menjawab dengan cara menguraikan, menjelaskan, mendiskusikan, membandingkan, memberikan alasan, dan bentuk sejenis menggunakan kata-kata serta bahasa sendiri (Astiti, 2017). Ujian esai digunakan untuk mengatasi kelemahan daya ukur soal objektif dan cocok untuk mengukur hasil belajar yang level kognisinya lebih tinggi dan bersifat kompleks (Astiti, 2017). Ujian bentuk esai ini sangat baik digunakan untuk melatih pelajar mengorganiasikan pikiran-pikirannya, mengeluarkan ide atau pendapatnya dengan menggunakan kata-kata sendiri dan melatih penalaran (Sary, 2015).

Ciri utama ujian esai menurut Yusuf (2015) adalah pelajar menyusun jawabannya sendiri, pelajar menggunakan bahasa dan kata-katanya sendiri dalam menjawab pertanyaan, pertanyaan yang diajukan lebih bersifat umum dan jumlahnya

sedikit serta pelajar mengemukakan jawabannya dengan bermacam-macam kelengkapan dan ketelitian sesuai dengan kemampuan masing-masing.

Kelebihan mengadakan ujian dalam bentuk esai adalah menuntut pelajar untuk menjawab pertanyaan dengan seluruh pengetahuannya, menuntut kreativitas dalam menyusun jawaban sendiri, memotivasi untuk mengemukakan pendapat, dan pengajar mengetahui tingkat penguasaan pelajar terhadap materi, mengetahui jalan pikiran pelajar dalam menjawab pertanyaan serta pengajar lebih mudah dalam menyusun ujian dalam bentuk ini (Astiti, 2017).

Hal-hal yang menyulitkan dan menjadi kelemahan ujian esai menurut Astiti (2017) yaitu jawaban-jawaban yang diberikan bervariasi sehingga tidak ada rumusan jawaban benar yang pasti, dalam memberikan penilaian lebih memberikan peluang untuk bersifat subjektif baik karena dinilai oleh orang yang berbeda ataupun orang yang sama dalam situasi yang berbeda, pengoreksian membutukan waktu yang lama, dan skor yang dicapai pelajar yang mengikuti ujian tidak konsisten jika ujian yang sama dilakukan kembali atau ujian yang parallel diuji ulang beberapa kali.

2.3 Pedoman Penskoran

Pedoman penskoran adalah pedoman yang digunakan untuk menentukan skor hasil pekerjaan pelajar (Sumaryanta, 2015). Dengan menyusun pedoman penskoran artinya pengajar telah memikirkan strategi pemberian skor terhadap jawaban yang akan muncul (Sary, 2015). Pedoman penskoran ini sangat penting disiapkan khususnya untuk ujian dalam bentuk esai. Dengan adanya pedoman penskoran akan memudahkan pengajar atau penilai dalam membandingkan jawaban dengan kunci jawaban sesuai kemiripan dan tingkat kesempurnaan serta kelengkapan jawaban pelajar dengan ideal (Sary, 2015).

Dalam Permendikbud No. 66 Tahun 2013 tentang Standar Penilaian dijelaskan bahwa penilaian harus dilakukan dengan objektif dan akuntabel serta berbasis pada standar, tidak dipengaruhi oleh faktor subjektivitas penilai serta harus dapat dipertanggungjawabkan kepada pihak internal maupun eksternal sekolah. Sehingga pedoman penskoran yang baik akan membantu pengajar dalam memenuhi kedua prinsip penilaian tersebut. Dengan pedoman penskoran pengajar dapat memberikan

penghargaan yang lebih akurat dan adil untuk seluruh pelajar dengan masing-masing cara penyelesaiannya yang mungkin satu dengan yang lain berbeda.

Contoh pedoman penskoran untuk soal “Sebutkan lima menu dalam excel” (Arifin, 2017) :

Tabel 2.1 Pedoman Penskoran

Kunci Jawaban	**Skor**
File	1
Edit	1
View	1
Insert	1
Format	1
Skor Maksimum	**5**

2.4 *Text Mining*

Text mining adalah seperangkat teknik dan metode yang digunakan untuk pemrosesan otomatis pada data teks bahasa alami yang tersedia dalam jumlah yang cukup besar dalam bentuk file komputer dengan tujuan mengekstraksi dan menyusun konten untuk tujuan analisis cepat, penemuan data tersembunyi, atau pengambilan keputusan otomatis (Tuffery, 2011). Sedangkan menurut Fledman (2007), *Text mining* adalah proses mengekstrak informasi yang berguna dari sumber data yang berupa dokumen bahasa alami melalui identifikasi dan mengeksplorasi pola yang penting. Proses pada *text mining* banyak terinspirasi dan mengadopsi proses pada *data mining*. Pada *text mining* proses identifikasi dan pengekstrakan dilakukan pada dokumen bahasa alami, dimana data dokumen tersebut tidak terstuktur dan operasi *preprocessing* perlu dilakukan untuk mengubah dokumen menjadi lebih terstruktur (Fledman, 2007).

Text preprocessing dan *feature selection* adalah tahap-tahap umum pada *text mining* (Berry, 2010). *Text preprocessing* adalah tahap yang mencangkup semua proses untuk mempersiapkan data (Fledman, 2007). Tahap *text preprocessing* dibutuhkan karena dokumen-dokumen teks yang akan diproses merupakan data yang tidak terstruktur. Proses yang dilakukan pada pada tahap tersebut adalah *toLowerCase* (*Case Folding*) dan *tokenizing*. Sedangkan tahap *feature selection*

bertujuan mengurangi dimensi dari kumpulan teks dengan menghapus kata-kata yang tidak penting dalam dokumen, proses yang dilakukan adalah *stopword* dan *stemming* (Berry, 2010).

2.5 *Case Folding*

Case folding (*toLowerCase*) merupakan proses untuk merubah semua huruf besar pada teks dalam dokumen menjadi huruf kecil atau sebaliknya. Proses ini dibutuhkan karena tidak semua dokumen menggunakan huruf besar secara konsisten. Sebagai contoh pada kata RMIT University, Rmit University, atau rmit university yang seharusnya merupakan informasi yang sama. Proses ini dilakukan supaya semua teks dalam dokumen memiliki standar yang sama (Asian, 2007).

2.6 *Tokenizing*

Tokenizing adalah proses penguraian kalimat-kalimat menjadi kata-kata dan menghilangkan *delimiter* seperti tanda titik (.), koma (,), spasi dan karakter angka yang ada (Weiss, 2005). Sedangkan menurut Asian (2007), *Tokenizing* adalah proses untuk memecah *string* menjadi *token* yang merupakan unit dasar sebelum diproses lebih lanjut, *token* dapat didefinisikan dalam format yang berbeda-beda biasa berupa kata, *idioms*, *morphemes*, dan *n-grams*. Dokumen juga dapat dipecah berdasarkan bab, bagian, paragraf, kalimat, kata, dan bahkan suku kata atau *fonem*, akan tetapi pendekatan yang paling sering dilakukan pada *text mining* adalah memecah atau mengurai teks menjadi kalimat dan kata-kata (Fledman, 2007).

2.7 *Stopword*

Proses untuk membuang *token* yang termasuk dalam daftar *stopword* disebut dengan *stopword removal*. *Stopword* adalah daftar kata yang sering muncul atau kata-kata yang tidak memiliki makna penting dalam dokumen (informasi) dan hanya berfungsi sebagai penanda gramatikal (Asian, 2007). *Stopword removal* berfungsi untuk mengurangi ukuran indeks, mengurangi waktu pemprosesan dan juga mengurangi tingkat *noise* (Asian, 2007). *Stopword* untuk setiap bahasa berbeda. Salah satu *stopword* untuk bahasa indonesia adalah *stopword* Tala yang terdapat pada artikel berjudul *A Study of Stemming Effects on Information Retrieval in Bahasa Indonesia* (2003).

2.8 *Stemming*

Stemming adalah proses pemetaan dan penguraian berbagai bentuk varian dari suatu kata menjadi bentuk kata dasar (umum) (Tala, 2003). Proses *stemming* untuk setiap bahasa berbeda, teknik yang bekerja dengan baik pada satu bahasa mungkin tidak bekerja pada bahasa lain (Asian, 2007). *Stemming* untuk Bahasa Indonesia salahsatunya adalah *stemming* Nazief dan Adriani (1996), *stemming* ini dilakukan berdasarkan struktur *morfologi* kata Bahasa Indonesia, yang terdiri dari *prefiks* (awalan), *sufiks* (akhiran), *infiks* (sisipan), dan *konfiks* (awalan+akhiran). *Stemming* tersebut kemudian dikembangkan oleh Asian pada tahun 2007 dengan menambah beberapa aturan dan diberi nama *confix stripping (sc) stemmer*. *Confix stripping (sc) stemmer* ini kemudian dikembangkan kembali dengan perubahan dan penambahan beberapa aturan serta penambahan langkah loopPengembalianAkhiran untuk penyusunan kembali kata-kata yang mengalami proses *stemming* berlebih. Mahendra menamai stemming tersebut dengan *enhanced confix stripping stemmer*.

2.9 *Enhanced Confix Stripping Stemmer*

Enhanced confix stripping stemmer adalah *stemming* untuk Bahasa Indonesia yang dikembangkan oleh I Putu Adhi Kerta Mahendra pada tahun 2008. Pengembangan tersebut dilakukan atas dasar kegagalan *stemming* pada *confix stripping* (CS) *stemmer*. Kegagalan tersebut terjadi karena (Mahendra, 2008) **:**

1. Kurangnya aturan penguraian awalan pada format kata “mem+p...”, misalnya untuk kata “mempromosikan” dan “memprediksi”.
2. Kurangnya aturan penguraian awalan pada format kata “men+s”, misalnya untuk kata “mensyaratkan”.
3. Tidak relevannya aturan penguraian awalan pada format kata “menge+kata dasar”, contohnya pada kata “mengerem”.
4. Tidak relevannya aturan penguraian awalan pada format kata “penge+kata dasar”, contohnya kata “pengeboman”.
5. Tidak relevannya aturan penguraian awalan pada format kata “peng+k...”, seperti pada kata “pengkajian”.
6. Kegagalan *stemming* pada beberapa kata dasar berakhiran menyerupai imbuhan yang seharusnya tidak dihilangkan, contohnya akhiran “-an” dan “-ku” pada kata “pelanggan” dan “pelaku”.

Berdasarkan hal-hal diatas kemudian dilakukan perbaikan untuk memperoleh hasil *stemming* yang lebih baik dengan cara sebagai berikut :

1. Merevisi aturan penguraian awalan untuk format kata “mem+p...”, “men+s...”, “menge+...”, “penge+...”, dan “peng+k...” pada Tabel 2.2. Hasil revisi dapat dilihat pada Tabel 2.3.
2. Menambahkan langkah untuk mengatasi kegagalan *stemming* karena kesalahan penguraian akhiran yang seharusnya tidak dilakukan. Langkah ini disebut loopPengembalianAkhiran.

Tabel 2.2 Aturan Penguraian Awalan

Aturan	**Format Kata**	**Pemenggalan**
1	berV	ber-V… | be-rV…
2	berCAP	ber-CAP… dimana C!=’r’ & P!=’er’
3	berCAerV…	Ber-CaerV… dimana C!=’r’
4	belajar	bel-ajar
5	beC_1erC_2…	be-C_1erC_2… dimana C1!={‘r’|’l’}
6	terV…	ter-V… | te-rV…
7	terCerV…	ter-CerV… dimana C!=’r’
8	terCP…	ter-CP… dimana C!=’r’ dan P!=’er’
9	teC_1erC_2…	te-C_1erC_2…dimana C1!=’r’
10	me{l|r|w|y}V…	me-{l|r|w|y}V…
11	mem{b|f|v}…	mem-{b|f|v}…
12	mempe…	mem-pe…
13	mem{rV|V}…	me-m{rV|V}… | me-p{rV|V}…
14	men{c|d|j|z}…	men-{c|d|j|z}…
15	menV…	me-nV… | me-tV
16	meng{g|h|q|k}…	meng-{g|h|q|k}…
17	mengV…	meng-V… | meng-kV…
18	menyV…	meny-sV…
19	mempV…	mem-pV… dimana V!=’e’
20	pe{w|y}V…	pe-{w|y}V…
21	perV…	Per-V… | pe-rV…
23	perCAP	per-CAP… dimana C!=’r’ dan P!=’er’
24	perCAerV…	per-CaerV… dimana C!=’r’
25	pem{b|f|V}…	pem-{b|f|V}…
26	pem{rV|V}…	pe-m{rV|V}… | pe-p{rV|V}…
27	pen{c|d|j|z}…	Pen-{c|d|j|z}…
28	penV…	pe-nV… | pe-tV…
29	peng{g|h|q}…	peng-{g|h|q}…
30	pengV…	Peng-V… | peng-kV…
31	penyV…	Peny-sV…
32	pelV…	pelV… kecuali pelajar yang menghasilkan “ajar”

Aturan	Format Kata	Pemenggalan
33	peCerV…	per-erV… dimana C!={r|w|y|l|m|n}
34	peCP	pe-CP… dimana C!= {r|w|y|l|m|n} dan P!='er'
35	terC_1erC_2	ter-C_1erC_2… dimana P!='er'
36	peC_1erC_2	pe-C_1erC_2… diaman C1!={r|w|y|l|m|n}

Tabel 2.3 Revisi untuk Tabel 2.2

Aturan	Format Kata	Pemenggalan
14	men{c|d|j|s|z}…	men-{c|d|j|s|z}
17	mengV…	Meng-V… | meng-kV… | (mengV-… jika V='e')
19	mempA…	mem-pA… dimana A!='e'
29	pengC…	peng-C…
30	pengV…	peng-V… | peng-kV… | (pengV-… jika V='e')

Keterangan :

- C merupakan kosonan,
- V merupakan vokal,
- A merupakan vokal atau kosonan dan
- P merupakan partikel

Aturan *morfologi* kata imbuhan pada bahasa indonesia dikelompokkan dalam beberapa kategori yaitu (Mahendra, 2008) :

1. *Inflection Suffixes* adalah kelompok-kelompok akhiran yang tidak mengubah bentuk kata dasar, terdiri dari :
   - *Particle* (P) yaitu "-lah", "-kah", "-tah", dan "-pun".
   - *Possessive Pronoun* (PP) atau kata ganti kepunyaanya yaitu "-ku", "-mu", dan "-nya".
2. *Derivation Suffixes* (DS) adalah akhiran-akhiran yang secara langsung dapat ditambahkan pada kata dasar yaitu akhiran "-i", "-kan", dan "-an".
3. *Derivation Prefixes* (DP) adalah awalan-awalan yang dapat langsung diberikan pada kata dasar murni, atau pada kata dasar yang sudah mendapatkan penambahan sampai dengan 2 awalan yaitu awalan bermorfologi ("me-", "be-", "pe-", dan "te-") dan awalan yang tidak bermorfologi ("di-", "ke-" dan "se-").

Berdasarkan kategori-kategori diatas maka bentuk kata dalam bahasa indonesia dapat dimodelkan sebagai berikut :

[DP+[DP + [DP+]]] Kata Dasar [[+DS][+PP][+P]] (2.1)

Dengan batasan-batasan sebagai berikut (Mahendra, 2008) :

- Terdapat kombinasi imbuhan yang dilarang seperti yang dapat dilihat pada Tabel 2.4.
- Penggunaan imbuhan yang sama secara berulang tidak diperbolehkan.
- Proses *stemming* tidak dilakukan jika suatu kata hanya terdiri dari satu atau dua huruf.
- Penambahan suatu awalan tertentu dapat mengubah bentuk asli kata dasar dan awalan yang telah diberikan sebelumnya pada kata dasar bersangkutan (bermorfologi).

Tabel 2.4 Kombinasi Imbuhan yang tidak Diperbolehkan

Awalan (prefix)	**Akhiran (suffix) yang tidak perbolehkan**
be-	-i
di-	-an
ke-	-i, -kan
me-	-an
se-	-i,-kan
te-	-an

Berikut adalah langkah-langkah yang dilakukan pada *enhanced confix stripping stemmer* dalam mencari kata dasar (Mahendra, 2008) :

1. Kata yang akan dicari kata dasarnya terlebih dahulu dicek dalam kamus kata dasar. Kata tersebut merupakan kata dasar jika ditemukan di kamus kata dasar. Jika tidak ditemukan lanjut langkah ke 2.
2. Cek *rule precedence* dengan cara memperhatikan pasangan awalan-akhiran pada kata tersebut. Pada kata yang memiliki pasangan awalan-akhiran “be-lah”, “be-an”, “me-i”, “di-i”, “pe-i”, atau “te-i”, *stemming* yang dilakukan dengan urutan langkah 5, 6, 3, 4, 7. Sedangkan kata yang tidak memiliki awalan-akhiran tersebut langkah *stemming* dilakukan secara berurutan yaitu 3, 4, 5, 6, 7.
3. Menghapus *particle* (P) dan *possessive pronoun* (PP) sehingga menjadi

[DP+[DP+[DP+]]] Kata Dasar [+DS] (2.2)

4. Menghapus *derivation suffixes* (DS)

[DP+[DP+[DP+]]] Kata Dasar .(2.3)

5. Menghapus *derivational Prefixes* (DP) dengan iterasi maksimum adalah 3 kali:
   a) Langkah 5 berhenti jika:
      - Terjadi kombinasi imbuhan terlarang seperti pada Tabel 2.4.
      - Awalan yang dideteksi saat ini sama dengan awalan yang dihapus sebelumnya.
      - Tiga awalan telah dihapus
   b) Identifikasikan tipe awalan dan hapus. Awalan ada dua tipe:
      - Standar : "di-", "ke-", "se-" yang dapat langsung dihilangkan dari kata.
      - Kompleks : "me-", "be-", "pe", "te-" adalah tipe-tipe awalan yang dapat bermorfologi sesuai kata dasar yang mengikutinya. Oleh karena itu, gunakan aturan pada Tabel 2.2 (dengan revisi Tabel 2.3) untuk mendapatkan penuraian yang tepat.
   c) Cari kata yang telah dihilangkan awalannya ini di dalam kamus kata dasar. Apabila tidak ditemukan, ulangi kembali langkah 5. Apabila ditemukan, maka keseluruhan proses dihentikan
6. Lakukan proses *recoding* apabila setelah langkah 5 kata dasar masih belum ditemukan. *Recoding* dilakukan dengan menambahkan karakter *recoding* di awal kata yang diurai. Pada Tabel 2.2 (dengan revisi Tabel 2.3) karakter *recoding* adalah karakter setelah tanda hubung ('-') dan terkadang berada sebelum tanda kurung. Sebagai contoh, pada kata "menangkap" (aturan 15), setelah dipenggal menjadi "nangkap". Karena tidak valid, maka *recoding* dilakukan dan menghasilkan kata "tangkap".
7. Jika semua langkah gagal dan kat belum ditemukan pada kamus kata dasar, maka lakukan proses Langkah loopPengembalianAkhiran dengan cara sebagai berikut :
   1) Kembalikan semua awalan yang telah dihapus, sehingga menghasilkan model kata seperti berikut:

      [DP+[DP+[DP]]] + Kata Dasar (2.4)

   2) Kembalikan akhiran sesuai dengan urutan model kata dimulai dari DS, lalu PP, dan terakhir adalah P. Pada setiap pengembalian lakukan langkah 3) hingga 5). Khusus untuk akhiran "-kan", pengembalian pertama dimulai dengan "k", baru kemudian dilanjutkan dengan "an".

3) Lakukan pencarian di kamus kata dasar. Apabila ditemukan, proses dihentikan. Apabila gagal, maka lakukan proses penguraian awalan berdasarkan aturan pada Tabel 2.2 (dengan revisi Tabel 2.3).
4) Lakukan *recoding* apabila diperlukan.
5) Apabila pencarian di kamus kata dasar tetap gagal setelah *recoding*, maka awalan-awalan yang telah dihilangkan dikembalikan seperti model kata pada langkah 1) dan kata tersebut dianggap sebagai kata dasar.

Apabila pada kata yang akan di-*stemming* memiliki tanda hubung (’-’), maka kemungkinan kata tersebut adalah kata ulang. Untuk kata ulang, *stemming* dilakukan dengan memecah kata menjadi dua bagian yaitu bagian kiri dan kanan (berdasarkan posisi tanda hubung (’-’) dan lakukan *stemming* pada dua kata tersebut. Apabila hasil *stemming* keduanya sama, maka kata dasar berhasil didapatkan.

2.10 Pengoreksian Ejaan *(Spelling Correction)*

Spelling Correction adalah proses yang digunakan untuk mengoreksi kesalahan pengejaan. *Spelling Correction* ini bertujuan memberi saran berupa kata atau kumpulan kata untuk memperbaiki kata yang dianggap salah (Braddley, 2017)**.** Disebut terjadi kesalahan pengejaan jika kata tersebut tidak ditemukan dalam kamus. Kesalahan ejaan sering kali terjadi karena kesalahan huruf, penyisipan huruf, hilangnya huruf, dan tertukarnya letak huruf (Dwitiyastuti, 2013)**.**

Pada penelitian yang dilakukan oleh Braddley (2017) dan Dwitiyastuti (2013), proses mendapatkan saran kata untuk memperbaiki kesalahan ejaan dilakukan dengan bantuan algoritma *levenshtein distance*. Pada penelitian-penelitian tersebut mula-mula dilakukan memeriksa keberadaan kata dalam kamus, lalu memberikan pilihan kata yang benar untuk kata yang tidak terdapat pada kamus. Algoritma *levenshtein distance* digunakan untuk mencari kata rekomendasi perbaikan berdasarkan jumlah perbedaan terdekat dengan kata yang salah ejaannya.

2.11 *Levenshtein Distance*

Levenshtein distance atau sering disebut juga sebagai *edit distance* adalah suatu pengukuran (metrik) yang dihasilkan melalui perhitungan jumlah perbedaan yang terdapat pada dua *string* atau kata (Dwitiyastuti, 2013). Ada tiga operasi yang dapat

dilakukan algoritma ini yaitu operasi penambahan (*insert*) dilakukan dengan menambahkan sebuah karakter pada kata contohnya "sya" menjadi "saya", operasi penghapusan (*delete*) dilakukan dengan menghilangkan sebuah karakter pada kata contohnya "lagio" menjadi "lagi" dengan menghilangkan karakter 'o', dan operasi penggantian karakter (*substitute*) dilakukan dengan menganti sebuah karakter pada kata misalnya mengganti karakter 'm' menjadi 'n' pada kata yang semula "yamg" menjadi "yang" (Braddley, 2017). Perhitungan jumlah perbedaan (*distance*) ditentukan dengan banyaknya operasi untuk mengubah suatu bentuk *string* menjadi bentuk *string* yang lain contohnya *string* "hasal" membutuhkan 1 operasi pengantian untuk menjadi "hasil" artinya *distance* "hasal" menjadi "hasil" adalah 1 (Dwitiyastuti, 2013).

Berikut ini adalah cara kerja algoritma *levenshtein distance*, misalkan S = *String* Awal, dan T = *String* Target (Haldar, 2011) :

Langkah 1 : Inisialisasi

a) Hitung banyaknya karakter pada string 'S' dan simpan hasilnya di 'n'. Hitung banyaknya karakter pada string 'T' dan simpan hasilnya di m.
b) Buat matriks berukuran 0...m baris dan 0...n kolom
c) Inisialisasi baris pertama dengan 0...n
d) Inisialisasi kolom pertama dengan 0...m

Langkah 2 : Proses

a) Ambil karakter pada S[i], dimana i adalah indeks dari matriks string 'S'.
b) Ambil karakter pada T[j], dimana j adalah indeks dari matriks string 'S'.
c) Jika S[i] = T[j], *cost* = 0
d) Jika S[i] ≠ T[j], *cost* = 1
e) Isi d[i,j] nilai minimum dari:
    - Nilai yang terletak tepat diatasnya, ditambah satu, yaitu d[i,j-1]+1
    - Nilai yang terletak tepat dikirinya, ditambah satu, yaitu d[i-1,j]+1
    - terletak pada tepat didiagonal atas sebelah kirinya, ditambah *cost*, yaitu d[i-1,j-1]+*cost*

Langkah 3 : Hasilnya adalah yang diisiakan pada matriks baris ke-i dan kolom ke-j, yaitu d[i,j], kemudian ulang langkah 2 hingga entri d[m,n] ditemukan.

2.12 *Pengenalan Sinonim (Synonym Recognition)*

Synonym recognition adalah teknik yang digunakan untuk mengenali kata dengan penulisan berbeda namun memiliki makna yang sama. *Synonym recognition* diperlukan karena hampir setiap kata khususnya dalam bahasa indonesia memiliki sinonim. Banyaknya kata sinonim yang berasal dari kata dasar, oleh karena itu apabila proses *stemming* tidak berjalan dengan baik, maka pengenalan kata bersinonim juga menjadi tidak sesuai, dan berdampak pada berkurangnya keakuratan pendeteksian (Djafar, 2014). Untuk melakukan *synonym recognition* dibutuhkan kamus sinonim. Proses *synonym recognition* dilakukan dengan cara mengecek kata dalam kamus sinonim, jika kata tersebut ada pada kamus sinonim maka kata tersebut diubah menjadi kata utamanya (Jody, 2015).

2.13 *Longest Common Subsequences (LCS)*

Longest common subsequences (LSC) merupakan metode untuk mencari kesamaan terpanjang pada *sequence* (Nicholas, 2015). Atau dapat diartikan juga bahwa LCS adalah metode untuk menghitung relasi berurutan yang paling panjang antara *query* dengan dokumen (Saadah, 2013). Dimana dalam menghitung kesamaan urutan terpanjang tersebut dapat dilakukan dengan melewati beberapa elemen yang tidak sama urutannya (Mahalakshmi, 2016). Hal itu dapat dilakukan karena *subsequence* adalah sebuah *sequence* yang diturunkan dari *sequence* lain dengan menghapus beberapa elemen tanpa mengganti urutan dari sisa *element* tersebut (Nicholas, 2015)**.** Sebagai contoh *string* “ACCTGGTTTTGTTC” merupakan hasil pencarian LCS pada *string* “AAACCGTGAGTTATTCGTTCTAGAA” dengan *string* “CACCCCTAAGGTACCTTTGGTTC” (Nicholas, 2015).

Berikut ini adalah contoh langkah-langkah untuk pencarian *subsequence* dari *string* X yang memiliki sequence BDCB dan Y yang memiliki sequence BACDB menggunakan LCS (Nicholas, 2015) :

1. Peletakan sequence X dan Y tidak berpengaruh terhadap perhitungan.
2. Input kolom paling kiri dan paling atas dengan angka 0.
3. Kemudian pertemukan setiap *sequence* satu per satu. Bila X sama dengan Y, maka nilai pada kolom tersebut adalah (n,n) = (n-1,n-1) + 1. Sedangkan bila X tidak sama dengan Y, maka nilai yang menjadi input adalah nilai terbesar

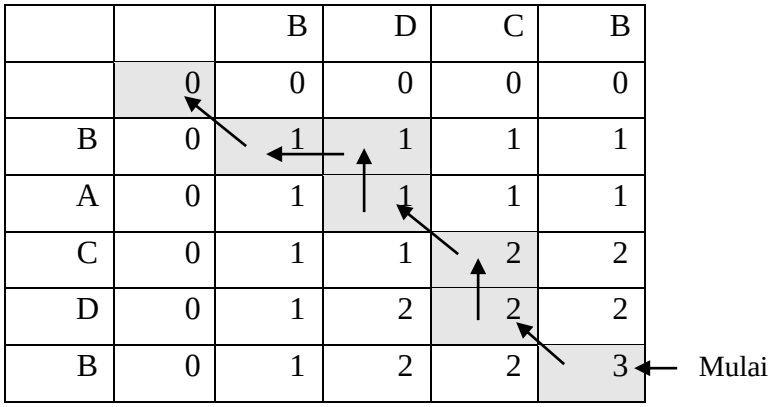
yang terdapat pada kolom (n-1,n) atau pada kolom (n,n-1). Misalkan nilai X = B dan nilai Y = B, bertemu pada kolom (2,2), maka input kolom tersebut dengan nilai (1,1) + 1, dalam hal ini nilai (1,1) = 0, sehingga 0+1 = 1. Bila, nilai X = D dan nilai Y = B yang bertemu pada kolom (3,2), maka bandingkan nilai dari kolom (2,2) yang bernilai 1 dengan nilai dari kolom (3,1) yang bernilai 0. Nilai 1 > 0, maka input pada kolom (3,2) adalah 1. Apabila nilai pada kolom (2,2) sama dengan nilai pada kolom (3,1) maka input kolom (3,2) sama dengan kedua kolom tersebut

4. Lakukan langkah 3 pada seluruh kolom yang tersisa.

		B	D	C	B
	0	0	0	0	0
B	0	1	1	1	1
A	0	1	1	1	1
C	0	1	1	2	2
D	0	1	2	2	2
B	0	1	2	2	3

Gambar 2.1 Tabel LSC String X dan Y

5. Setelah semua kolom terisi, selanjutnya dilakukan pencarian nilai LCS dengan langkah - langkah sebagai berikut :
   1) Meletakkan *pointer* pada kolom sudut kanan bawah, yakni kolom (5,6).
   2) Pointer akan bergerak dengan salah satu pola, yakni ke kiri lalu ke atas atau ke kanan lalu ke atas. Tetapi *pointer* tidak bisa bergerak menggunakan pola ini bila nilai pada kolom yang dituju tidak sama dengan nilai pada kolom semula. Bila hal ini terjadi maka *pointer* akan bergerak menyerong, misalnya dari (5,6) ke (4,5).
   3) *Sequence* tersebut akan diletakkan dengan urutan dari kanan ke kiri.
   4) *Pointer* akan terus bergerak sampai pada kolom dengan nilai 0. *Pointer* akan berhenti, dan nilai dari LCS telah didapat.



Gambar 2.2 Gambar Ilustrasi Perhitungan LCS

5) Pada gambar 2.2 dapat diambil kesimpulan bahwa nilai LCS antara X dan Y adalah BCB

2.14 *Term Frequency Inverse Document Frequency (tf-idf)*

Term frequency-inverse document frequency (tf-idf) merupakan metode pemberian bobot hubungan suatu kata (*term*) terhadap dokumen. Metode ini menggunakan dua konsep untuk pembobotan yaitu *term frequency* (tf) dan *document frequency* (df). Untuk dokumen tunggal tiap kalimat dianggap sebagai dokumen. *Term frequency* (tf) adalah pembobotan berdasarkan frekuensi kemunculan kata (t) pada kalimat (d) yang menunjukkan seberapa penting kata (t) di dalam kalimat. Sedangkan *document frequency* (df) adalah pembobotan berdasarkan banyaknya kalimat yang terdapat kata (t) yang menunjukkan seberapa umum kata tersebut. Bobot kata semakin besar jika sering muncul dalam suatu dokumen dan semakin kecil jika muncul dalam banyak dokumen yang ditunjukkan dengan bobot *inverse document frequency* (idf). Hasil akhir pembobotan metode ini didapatkan dari pengalian nilai tf dan idf (Robertson, 2004).

Inverse document frequency (idf) didapatkan dengan menggunakan rumus (Robertson, 2004):

$$idf_j = log\left({}^{D}/_{df_j}\right) \qquad (2.5).$$

Dengan :
D = jumlah semua dokumen dalam koleksi.
df_j = jumlah dokumen yang mengandung term t

Term frequency-inverse document frequency (tf-idf) adalah diperoleh dengan cara mengalikan nilai *term frequency* (tf) dengan nilai *inverse document frequency* (idf).

$$w_{ij} = tf_{ij} \times idf_j \tag{2.8}$$

$$W_{ij} = tf_{ij} \times log\left({}^{D}/_{df_j}\right) \tag{2.9}$$

Dimana :

W_{ij}	=	bobot term tj terhadap dokumen di
tf_{ij}	=	jumlah kemunculan term tj dalam dokumen di
D	=	jumlah semua dokumen yang ada dalam *database*
d_{fj}	=	jumlah dokumen yang mengandung term tj (minimal ada satu kata yaitu term tj)

Berdasarkan rumus 2.9, Maka dapat ditambahkan nilai 1 pada sisi idf, hal ini untuk menghindari nilai 0 jika D = df_j sehingga perhitungan bobot menjadi :

$$w_{ij} = tf_{ij} \times \left(log\left({}^{D}/_{df_j}\right) + 1\right) \tag{2.10}$$

2.15 *Cosine Similarity*

C*osine similarity* merupakan metode untuk menghitung tingkat kesamaan (*similarity*) antara dua buah objek (Sugiyamto, 2014). *Cosine similarity* menghitung tingkat kesamaan antara dua buah objek berdasarkan nilai cosinus sudut dari perkalian dua buah objek yang dibandingkan, karena cosinus dari 0^0 adalah 1 dan kurang dari 1 untuk nilai sudut yang lain, maka dua objek dikatakan sama ketika tingkat kesamaan sama dengan 1 (Saputra, 2013). Untuk notasi himpunan pada c*osine similarity* dapat dirumuskan (Sugiyamto, 2014) :

$$Similarity(X,Y) = \frac{|x \cap y|}{|x|^{\frac{1}{2}}.|y|^{\frac{1}{2}}} \tag{2.11}$$

Dimana :

$\|x \cap y\|$	=	jumlah term (kata) pada dokumen x dan pada dokumen y
$\|x\|$	=	jumlah term (kata) pada dokumen x
$\|y\|$	=	jumlah term (kata) pada dokumen y

Dari notasi himpunan diatas dapat dibuat persamaan matematika yaitu

$$Similarity\ (X,Y)\ =\ \frac{\sum_{i=1}^{i} x_i y_i}{\sqrt{\sum_{i=1}^{i} x_i^2 . \sum_{i=1}^{i} y_i^2}} \qquad (2.12)$$

Dimana :

x dan *y*	= dokumen yang berbeda
x_i	= term (kata) *i* yang ada pada dokumen *x*
y_i	= term (kata) *i* yang ada pada dokumen *y*

BAB III
KERANGKA KONSEP PENELITIAN

Bab ini akan membahas analisis masalah dan pemaparan konsep solusi yang akan dilakukan serta berisi hipotesis yang akan dibuktikan. Pemarapan konsep solusi dilakukan dengan mencari variabel, menentukan metode yang akan digunakan dan merancang desain sistem penilaian jawaban esai. Dengan konsep tersebut kemudian dibuat hipotesis penelitian yang akan dibuktikan dalam pengujian.

3.1 Analisis Masalah

Ujian adalah salah satu cara untuk mengetahui pemahaman pelajar terhadap pelajaran dan sumber evaluasi untuk mengetahui kesuksesan hasil belajar. Jenis ujian yang dinilai paling dapat mengasah kemampuan pelajar dalam menyampaikan apa yang telah dipahami dengan bahasanya sendiri salah satunya adalah ujian esai. Dengan ujian esai, pelajar dituntut untuk menyampaikan sesuatu informasi secara verbal dalam bentuk tulisan dengan bahasanya sendiri sehingga lebih dapat mengukur tingkat pemahaman pelajar terhadap materi yang telah dipelajari. Oleh karena hal tersebut ujian esai masih menjadi pilihan pengajar.

Akan tetapi menilai jawaban esai tidak semudah menilai ujian pilihan ganda. Hal tersebut karena jawaban dari pelajar akan beragam yang mengakibatkan membutuhkan waktu yang lama. Jika jumlah jawaban yang harus dikoreksi banyak maka waktu yang dibutuhkan untuk mengoreksi jawaban juga semakin banyak, ini tentunya juga akan berpengaruh pada keakurasian penilaian. Selain hal yang telah disebutkan, hal yang juga sangat mempengaruhi keakuratan penilaian jawaban ujian esai adalah sifat subjektif pada saat melakukan penilaian. Sifat subjektif dari penilaian esai ini akan menyebabkan bervariasi penilaian yang diberikan oleh penilai padahal jawaban memiliki makna yang sama. Penilaian jawaban esai untuk satu ujian juga sebaiknya dilakukan oleh satu orang hal ini untuk mengurangi sifat subjektif tadi.

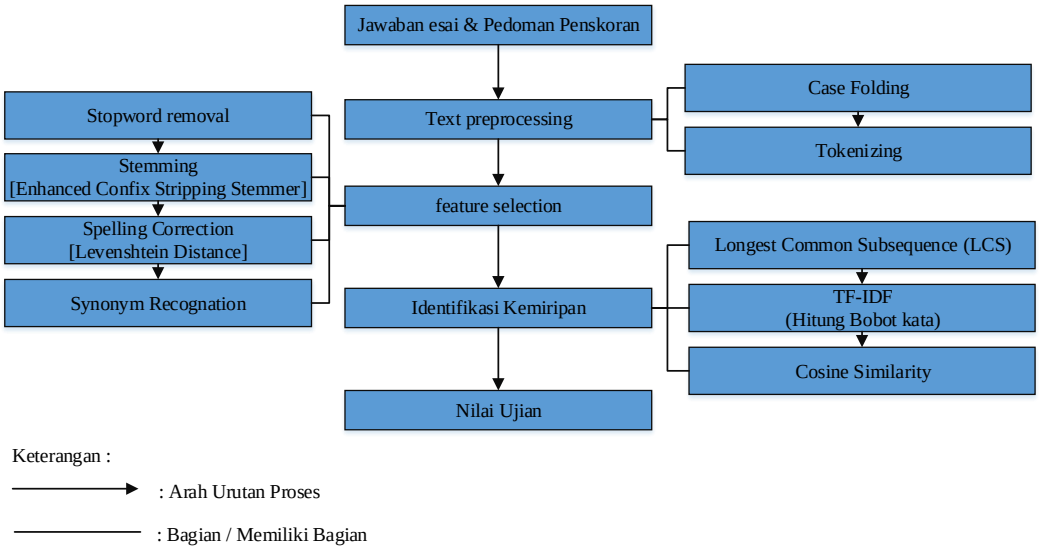
Dalam melakukan ujian disaat ini mulai banyak memanfaatkan kemajuan dibidang teknologi informasi yaitu dengan ujian secara online menggunakan

komputer. Akan tetapi dalam melakukan penilaian jawaban untuk ujian esai masih banyak yang menggunakan cara manual. Untuk itu dibutukan sistem yang dapat melakukan koreksi jawaban esai secara otomatis. Ada hal-hal yang harus diperhatikan untuk membuat sistem koreksi jawaban esai ini yaitu penggunaan pedoman penskoran dapat mengurangi subjektifitas penilaian, dimungkinkan adanya kesalahan pengetikan dan penggunaan kata bersinonim. Oleh karena itu sistem juga harus bisa menangani masalah tersebut.

3.2 Konsep Solusi

Untuk melakukan penilaian jawaban esai dalam sistem, langkah awal yang perlu dilakukan adalah membuat data terutama jawaban esai dari pelajar menjadi lebih terstuktur. Sebagaimana pengertian dari Astiti (2017) bahwa ujian esai merupakan ujian yang pertanyaannya menuntut pelajar untuk menjawab soal dengan cara menguraikan, menjelaskan, mendiskusikan, membandingkan, memberikan alasan menggunakan kata-kata sendiri, dengan kata lain ujian esai mengharapkan peserta didik menyusun sendiri jawabannya. Dimana hal itu menyebabkan jawaban ujian esai dari pelajar merupakan sekumpulan kata-kata yang tidak terstuktur tergantung pemahaman pelajar terhadap pelajaran yang diujikan. Mekanisme yang dapat dilakukan oleh sistem untuk mengubah jawaban esai menjadi lebih terstuktur adalah dengan menggunakan *text mining.*

Setelah data menjadi lebih terstuktur, langkah selanjutnya adalah menetukan rekomendasi nilai ujian. Sumaryanta (2015) dan Sary (2015) menyarankan untuk mempermudah dalam melakukan penilaian perlukan adanya pedoman penskoran. Pedoman penskoran tersebut berisi kunci-kunci kata atau kalimat yang harus ada pada jawaban, dimana kunci-kunci kata atau kalimat tersebut masing-masing memiliki skor. Maka untuk menerapkan hal ini, sistem harus mencari kombinasi yang paling tepat antara kunci-kunci kata atau kalimat dalam pedoman penskoran dengan bagian-bagian dari jawaban untuk kemudian dihitung kemiripannya. Hasil perhitungan kemiripan kemudian dikalikan dengan skor pada masing-masing kunci jawaban dan hasil akhirnya merupakan nilai rekomendasi dari sistem yaitu penjumlahan dari semua skor yang didapatkan.



Gambar 3.1 Kerangka Konsep Solusi

Kerangka konsep solusi yang akan digunakan untuk menyelesai masalah penilaian jawaban esai seperti yang terdapat pada Gambar 3.1. Data berupa jawaban esai dan kunci dalam pendoman penskoran pertama kali akan dilakukan proses pengekstrakan dengan menggunakan *text mining*. Tahap-tahap yang umum ada pada *text mining* adalah *text preprocessing* dan *feature selection. Text preprocessing* adalah tahap yang mencangkup semua proses untuk mempersiapkan data (Fledman, 2007), sedangkan *feature selection* bertujuan mengurangi dimensi dari kumpulan teks dengan menghapus kata-kata yang tidak penting dalam dokumen (Berry, 2010). Hasil dari pengekstrakan dengan *text mining* kemudian dilakukan identifikasi kemiripan untuk kemudian dikonversi menjadi rekomendasi nilai ujian.

Pada tahap *text preprocessing* akan melalui dua proses yaitu *case folding* dan *tokenizing*. Pada tahap *text preprocessing* ini, proses pertama yang akan dilakukan adalah proses *case folding.* Proses tersebut digunakan untuk proses untuk merubah semua huruf besar pada teks dalam dokumen menjadi huruf kecil. Hasil proses *case folding* akan menjadi *input* untuk proses *tokenizing.* Hasil proses *tokenizing* berupa daftar kata yang merupakan penguraian teks yang terdapat pada jawaban esai dan kunci dalam pendoman penskoran.

Tahap *feature selection* dilakukan setelah tahap *text preprocessing.* Pada tahap *feature selection* dilakukan secara berurutan proses *stopword removal*, *stemming*,

pengoreksian ejaan (*spelling correction*), dan terakhir mengubah kata-kata yang bersinonim menjadi kata utama (*synonym recognition*). Penelitian ini metode *stemming* yang digunakan adalah *enhanced confix stripping stemmer* karena metode ini merupakan pengembangan dari nazief dan adriani dan *confix stripping stemmer*. Pengoreksian ejaan dilakukan dengan algortima *levenshtein distance*. Proses *spelling correction* dan *synonym Recognition* merupakan proses tambahan yang dilakukan berdasarkan kesimpulan dan saran dari penelitian-penelitian penilaian esai sebelumnya.

Tahap identifikasi kemiripan merupakan tahap terakhir. Hasil dari tahap ini yang akan digunakan untuk dikonversi menjadi rekomendasi nilai ujian. Pada tahap ini pertama-tama dilakukan proses pencarian kombinasi. proses pencarian kombinasi dilakukan dengan mencari penggalan jawaban yang memiliki kemiripan urutan kata paling banyak (panjang) saat dibandingkan dengan kunci jawaban. Untuk mencari penggalan jawaban yang memiliki kemiripan urutan kata yang paling banyak tersebut digunakan *longest common subsequence*. Setelah ditentukan kombinasinya kemudian dilakukan pembobotan dengan metode *term frequency inverse document frequency* (tf-idf) dan terakhir dihitung kemiripan antara kedua data tersebut. Penghitungan kemiripan antara kedua data tersebut menggunakan *cosine similarity*. Hasil kemiripan *cosine similarity* kemudian dikalikan dengan skor pada masing-masing kunci jawaban dan hasil akhirnya merupakan nilai rekomendasi dari sistem yaitu penjumlahan dari semua skor yang didapatkan.

3.3 Hipotesis

Berdasarkan konsep solusi yang telah dijelaskan maka hipotesis dari penelitian ini adalah

1. Sistem mampu melakukan penilaian jawaban esai dengan cara mencari penggalan jawaban pada kunci-kunci jawaban yang terdapat pada pedoman penskoran yang memiliki kesamaan urutan yang paling banyak dengan menggunakan *longest common subsequence* (LSC) dan kemudian hasilnya dihitung tingkat kemiripannya menggunakan *cosine similarity*, untuk kemudian dikonversi menjadi nilai dari jawaban esai,

2. Sistem mendapatkan tingkat akurasi diatas 90% dan root mean square error (RMSE) mendekati 0 jika dibandingkan dengan penilaian jawaban secara manual,
3. Penambahan pengenalan sinonim (*synonym recognition*) dan pengoreksi ejaan (*spelling correction*) berperangaruh untuk meningkatkan tingkat akurasi hasil penilaian.

BAB IV
METODE PENELITIAN

4.1 Alat dan Bahan

Alat dan bahan merupakan komponen penting dalam penelitian. Alat dan bahan tersebut akan digunakan untuk mendukung pembuatan dan pengujian dari penelitian yang dilakukan. Alat yang digunakan dalam penelitian ini terdiri dari perangkat keras dan perangkat lunak. Berikut ini adalah rincian alat yang digunakan :

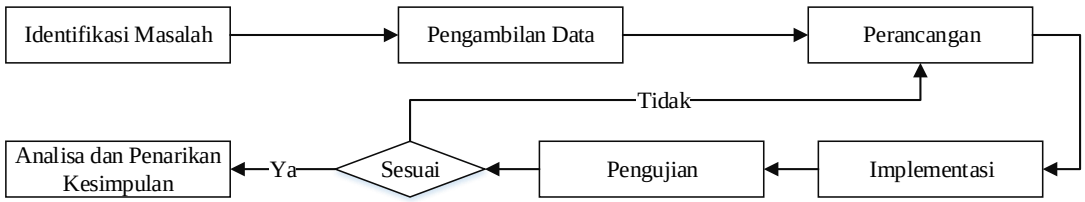
- Perangkat keras (laptop)
  - Processor : Intel(R) Core(TM) i5-4200U
  - Hardisk : 500 Gb
  - RAM : DDR3 4GB
- Perangkat lunak
  - Sistem Operasi Windows 10
  - XAMPP
  - JDK 1.8.0
  - Netbeans IDE 8.2

Sedangkan bahan penelitian adalah data yang digunakan untuk melakukan pengujian yang berupa *input* untuk sistem. Data tersebut adalah soal ujian esai, jawaban ujian esai pelajar dan kunci jawaban yang berupa pedoman penskoran serta sebagai pembanding dengan penilaian dari sistem adalah data nilai ujian pelajar yang dinilai manual oleh pengajar. Data tersebut didapatkan dari ujian di SMP Negeri 1 Paiton pada mata pelajaran bahasa indonesia, seni budaya dan IPA dengan jumlah soal masing-masing ujian adalah 5 soal. Ujian tersebut diikuti oleh 24 pelajar. Selain data yang menjadi objek penelitian diatas, untuk mendukung penelitian ini dibutuhkan juga daftar kata *stopword*, daftar kata dasar dan daftar kata sinonim.

4.2 Metode Penelitian

Metode penelitian yang digunakan dalam sistem penilaian jawaban ujian esai ini terdiri dari beberapa tahap yaitu identifikasi masalah, perancangan sistem,

implementasi dan yang terakhir adalah analisa dan penarikan kesimpulan. Proses tersebut dijelaskan pada Gambar 4.1.



Gambar 4.1 Metode Penelitian

4.2.1 Identifikasi Masalah

Pada tahap ini dilakukan pengkajian untuk mengidentifikasi masalah yang akan diangkat. Identifikasi masalah dilakukan dengan menentukan hal–hal penting dalam penyelesaian permasalahan dengan cara menganalisa kebutuhan perancangan sistem penilaian jawaban ujian esai. Tahap ini dilakukan dengan mengkaji jurnal-jurnal dan buku-buku terkait.

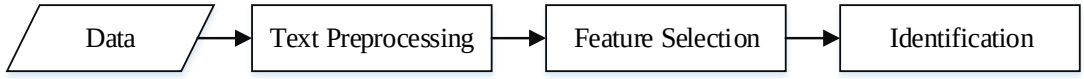
4.2.2 Pengambilan Data

Data yang digunakan untuk penelitian berupa soal ujian esai, jawaban ujian esai pelajar ber-Bahasa Indonesia berupa teks narasi dan kunci jawaban dari soal beserta pedoman penilaian jawaban soal serta sebagai pembanding dengan penilaian dari sistem adalah data nilai ujian pelajar yang dinilai manual oleh pengajar. Data tersebut didapatkan dari ujian di SMP Negeri 1 Paiton pada mata pelajaran bahasa indonesia, seni budaya dan IPA dengan jumlah soal masing-masing ujian adalah 5 soal. Ujian tersebut diikuti oleh 24 pelajar. Selain data yang menjadi objek penelitian diatas, untuk mendukung penelitian ini dibutuhkan juga daftar kata *stopword*, daftar kata dasar dan daftar kata sinonim. Daftar kata *stopword* didapatkan dari jurnal Tala (2003) yang berjudul *A Study of Stemming Effects on Information Retrieval in Bahasa Indonesia*. Daftar kata dasar dan sinonim didapatkan dari KBBI online dan kamusbesar.com.

4.2.3 Perancangan

Tahap perancangan digunakan untuk menggambarkan kebutuhan berdasarkan permasalahan dan perkiraan solusi yang akan diterapkan. Perancangan diperlukan

supaya proses implementasi sesuai dengan yang diharapkan dan dapat berjalan terarah. *Flowchart* Gambar 4.2 menggambarkan tahapan yang akan dilakukan.



Gambar 4.2. *Flowchart* Sistem Penilaian Jawaban Esai

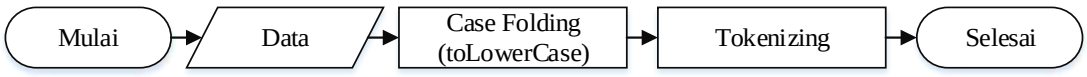
Berikut ini adalah penjelasan dari setiap tahapan pada sistem penilaian jawaban esai berdasarkan *flowchart* Gambar 4.2 :

a. Data

Tahap awal adalah memasukkan data yang akan diproses. Data yang digunakan pada proses ini adalah jawaban ujian pelajar berbahasa indonesia berupa teks narasi dan kunci jawaban dalam bentuk pendoman penskoran. Kedua data inilah yang akan dibandingkan.

b. *Text Preprocessing*

Tahap *text preprocesing* digunakan untuk mempersiapkan data sebelum proses *feature selection*. Proses pertama yang dilakukan pada tahap ini adalah *case folding* (*toLowerCase*) yang bertujuan untuk mengubah huruf pada data (baik jawaban pelajar maupun kunci jawaban) menjadi huruf kecil. Setelah itu dilakukan *tokenizing* untuk mengurai (memenggal) data yang berupa kalimat-kalimat menjadi daftar kata penyusun. Hasil dari tahap *text preprocessing* berupa dua daftar kata yang merupakan penguraian kalimat-kalimat yang terdapat pada kedua data *input*. Satu daftar kata merupakan hasil penguraian kalimat-kalimat yang terdapat pada jawaban pelajar dan satu daftar kata merupakan hasil penguraian kalimat-kalimat pada kunci jawaban. *Flowchart text preprocessing* dapat dilihat pada Gambar 4.3.

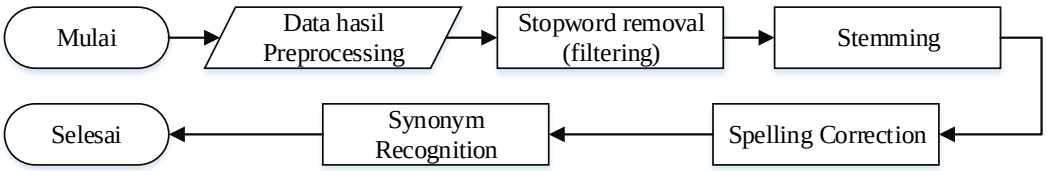


Gambar 4.3 *Flowchart Text Preprocessing*

c. *Feature Selection*

Tahap *feature selection* merupakan tahap untuk mendapatkan ciri atau parameter informasi yang merupakan kata kunci dari beberapa kata yang terdapat pada data. Pada tahap ini data yang telah berupa daftar kata yang merupakan hasil dari tahap *text preprocessing* dilakukan proses *filtering* (*stopword removal*), *stemming*, koreksi

ejaan dan *synonym recognition*. Proses *filtering* (*stopword removal*) dilakukan untuk membuang kata-kata yang tidak memiliki makna penting dan tidak mempengaruhi makna dari sebuah data sehingga akan mengurangi ukuran daftar kata dan hanya kata yang merupakan *keyword* dari data saja yang dipertahankan. Kata-kata hasil dari proses *filtering* (*stopword removal*) kemudian dicari bentuk kata dasarnya pada proses *stemming. Stemming* dilakukan dengan metode *enchanced confix stripping stemmer.* Dengan mengubah kata yang terdapat pada daftar kata tersebut menjadi kata dasar akan mempermudah pengecekan arti dari sebuah kata dimana kata yang serupa namun memiliki bentuk yang berbeda dapat diketahui dan dikelompokkan sehingga akan mengurangi jumlah pengindekan. Kedua proses diatas dilakukan pada daftar kata kunci jawaban dan jawaban pelajar. Proses selanjutnya pada tahap *feature selection* adalah melakukan pengecekan kesalahan pengetikan (*spelling correction*). Hasil dari *spelling correction* ini adalah rekomendasi kata hasil perbaikan ejaan dari setiap kata yang terdapat pada daftar kata jawaban pelajar. Koreksi ejaan dilakukan dengan metode *levenshtein distance.* Proses terakhir adalah proses *synonym recognition* yang bertujuan untuk mengecek dan mengubah kata pada daftar kata jawaban pelajar yang memiliki sinonim kekata utamanya. Alur proses *feature selection* seperti yang ditunjukkan pada Gambar 4.4.

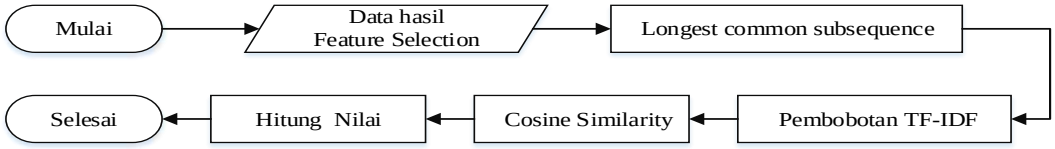


Gambar 4.4 *Flowchart Feature Selection*

d. *Identification*

Tahap *identification* adalah tahap terakhir dalam sistem penilaian jawaban esai. Seperti yang terdapat pada Gambar 4.5, proses pertama yang dilakukan adalah mencari kombinasi terbaik antara penggalan jawaban dengan kunci-kunci jawaban sesuai dengan pedoman penskoran masing-masing soal. Perhitungan kombinasi dilakukan dengan mencari kesamaan urutan kata terpanjang dengan menggunakan algoritma *longest common subsequence.* Hasilnya kemudian dilakukan perhitungan kemiripan jawaban dan kunci jawaban sesuai kombinasi dengan cara melakukan pembobotan kata menggunakan *term frequency-inverse document frequency* (tf-idf) dan perhitungan kemiripan dilakukan dengan *cosine similarity*. Hasil kemiripan

cosine similarity kemudian dikalikan dengan skor pada kunci jawaban dan hasil akhirnya merupakan nilai rekomendasi dari sistem yaitu penjumlahan dari semua hasil kali skor dan kemiripan yang didapatkan.



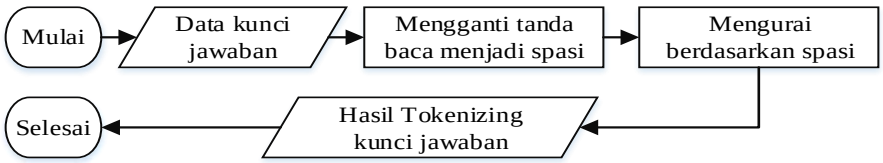
Gambar 4.5 *Flowchart Identification*

4.2.4 Implementasi

Pada penelitian ini sistem akan dibangun dengan bahasa pemrograman JAVA. Cara kerja dari sistem adalah seperti yang telah dipaparkan pada subbab perancangan. Pada subbab ini akan dijelaskan lebih detail tentang implementasi sistem berdasarkan metode yang digunakan dimasing-masing proses.

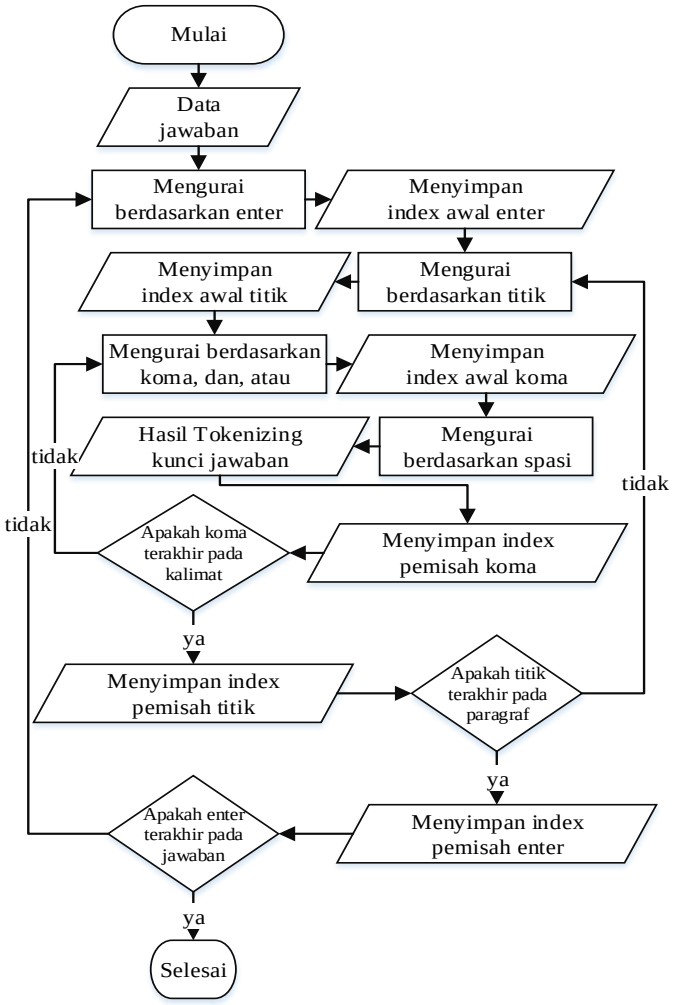
Proses pertama yang dilakukan setelah *input* data yang berupa kunci jawaban dan jawaban pelajar adalah *case folding* (*toLowerCase*). Proses ini bertujuan untuk mengubah huruf pada data (baik jawaban pelajar maupun kunci jawaban) menjadi huruf kecil. Proses ini dilakukan untuk mencegah masalah *case sensitive* dimana sistem membaca berbeda dua atau lebih kata hanya karena perbedaan antara huruf kapital dengan huruf kecil. Dengan *case folding* akan mengubah keseluruhan teks dalam data menjadi suatu bentuk standar yaitu berhuruf kecil.

Proses selanjutnya adalah *tokenizing* yaitu mengurai setiap kata dalam kalimat. Terdapat perbedaan proses *tokenizing* pada jawaban dari pelajar dan kunci jawaban. Perbedaan proses ini karena *tokenizing* pada jawaban dari pelajar membutuhkan proses penyimpanan index untuk setiap pemisah baris baru (*enter*), titik, koma, spasi, dan kata penghubung atau simbol yang menganti kata penghubung tersebut seperti dan, &, atau, /.



Gambar 4.6 *Flowchart Tokenizing* Kunci Jawaban

Proses *tokenizing* pada kunci jawaban dilakukan melalui 2 proses seperti pada Gambar 4.6. Proses pertama adalah mengganti semua tanda baca pada kunci jawaban menjadi spasi. Tanda baca ini seperti !"#$%&'()*+,-./:;<=>?@[\]^_`{|}~. Proses kedua adalah mengurai kalimat menjadi daftar kata berdasarkan pemisah spasi. 2 proses tersebut dilakukan pada setiap kunci jawaban yang terdapat pada pedoman penskoran pada setiap soal yang diujikan.

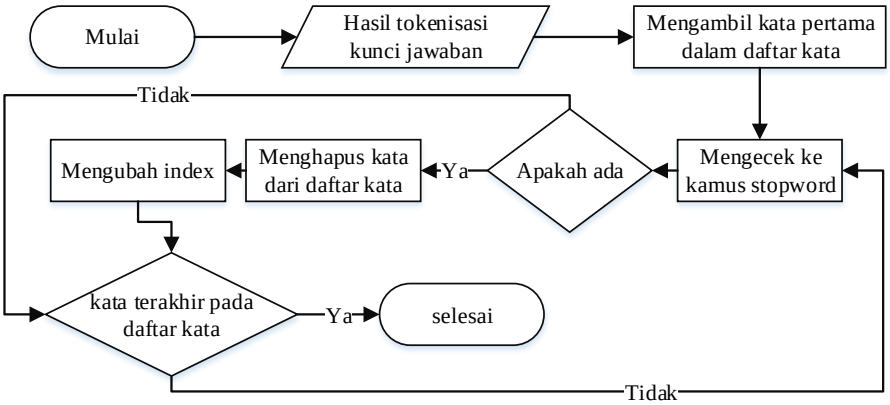


Gambar 4.7 *Flowchart Tokenizing* Jawaban

Proses *tokenizing* pada jawaban pelajar ditunjukkan pada Gambar 4.7. Terdapat proses penyimpanan index pada setiap pemisahan baris baru (*enter*), titik, koma, spasi, dan kata penghubung atau simbol yang menganti kata penghubung tersebut. Hal tersebut dilakukan karena pada proses *tokenizing* selain memisahkan kalimat-kalimat pada data menjadi daftar kata, proses ini juga akan menghilangkan baris baru

(*enter*), titik, koma, dan spasi. Sementara pada saat akan dilakukan perbandingan kemiripan di tahap *identification*, daftar kata tersebut akan dikelompokkan dan disusun kembali berdasarkan baris baru (*enter*), titik, koma, dan spasi dan susunan daftar kata tersebut harus tidak berubah meskipun telah melalui tahap *feature selection.* Penyusunan kembali tersebut bertujuan untuk mencari kombinasi jawaban terbaik dan paling mirip dengan kunci jawaban dalam pedoman penskoran.

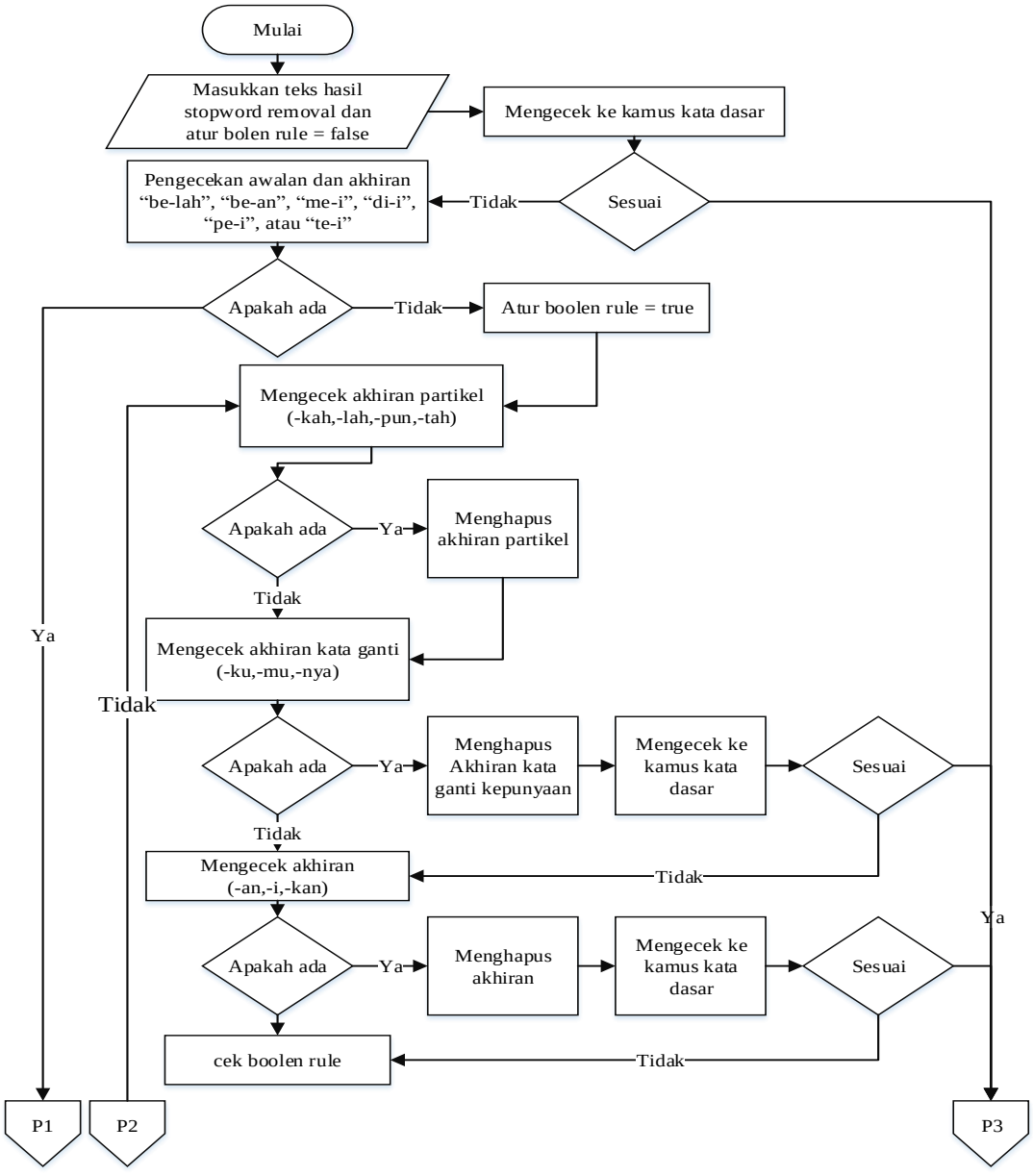
Pada proses *filtering* (*stopword removal*), data yang telah menjadi daftar kata kemudian difilter untuk membuang kata-kata yang termasuk dalam daftar kata *stopword*. Pada penelitian ini daftar *stopword* diambil dari jurnal Tala (2003) yang berjudul *A Study of Stemming Effects on Information Retrieval in Bahasa Indonesia*. Proses ini dilakukan pada daftar kata jawaban pelajar dan daftar kata kunci jawaban. Pada proses *filtering* daftar kata jawaban pelajar, setiap kata yang dibuang diikuti juga dengan perubahan index pemisahan baris baru (*enter*), titik, koma, spasi, dan kata penghubung. Gambar 4.8 menunjungkan proses *filterin*g pada jawaban pelajar.



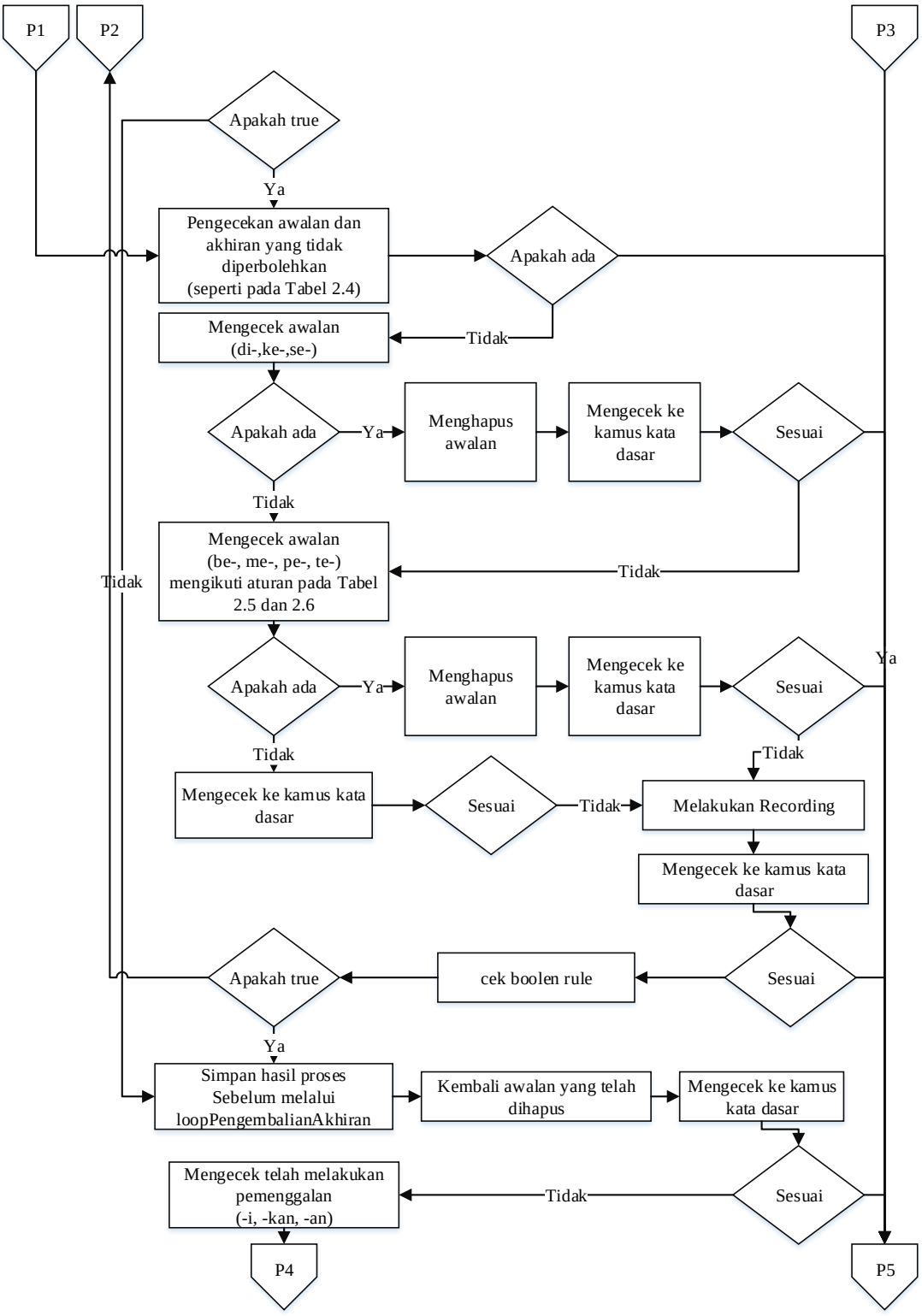
Gambar 4.8 *Flowchart Filtering* Jawaban

Daftar kata kunci jawaban dan daftar kata jawaban hasil proses *filtering* kemudian di*stemming* untuk mengembalikan setiap kata yang ada pada daftar-daftar kata tersebut kebentuk kata dasarnya. Dengan mengubah kata yang terdapat pada daftar kata tersebut menjadi kata dasar akan mempermudah pengecekan arti dari sebuah kata dimana kata yang serupa namun memiliki bentuk yang berbeda dapat diketahui dan dikelompokkan sehingga akan mengurangi jumlah pengindekkan. Pada penelitian ini *stemming* dilakukan dengan metode *enchanced confix stripping stemmer*. *Enchanced confix stripping stemmer* dipilih karena kata yang akan dicari

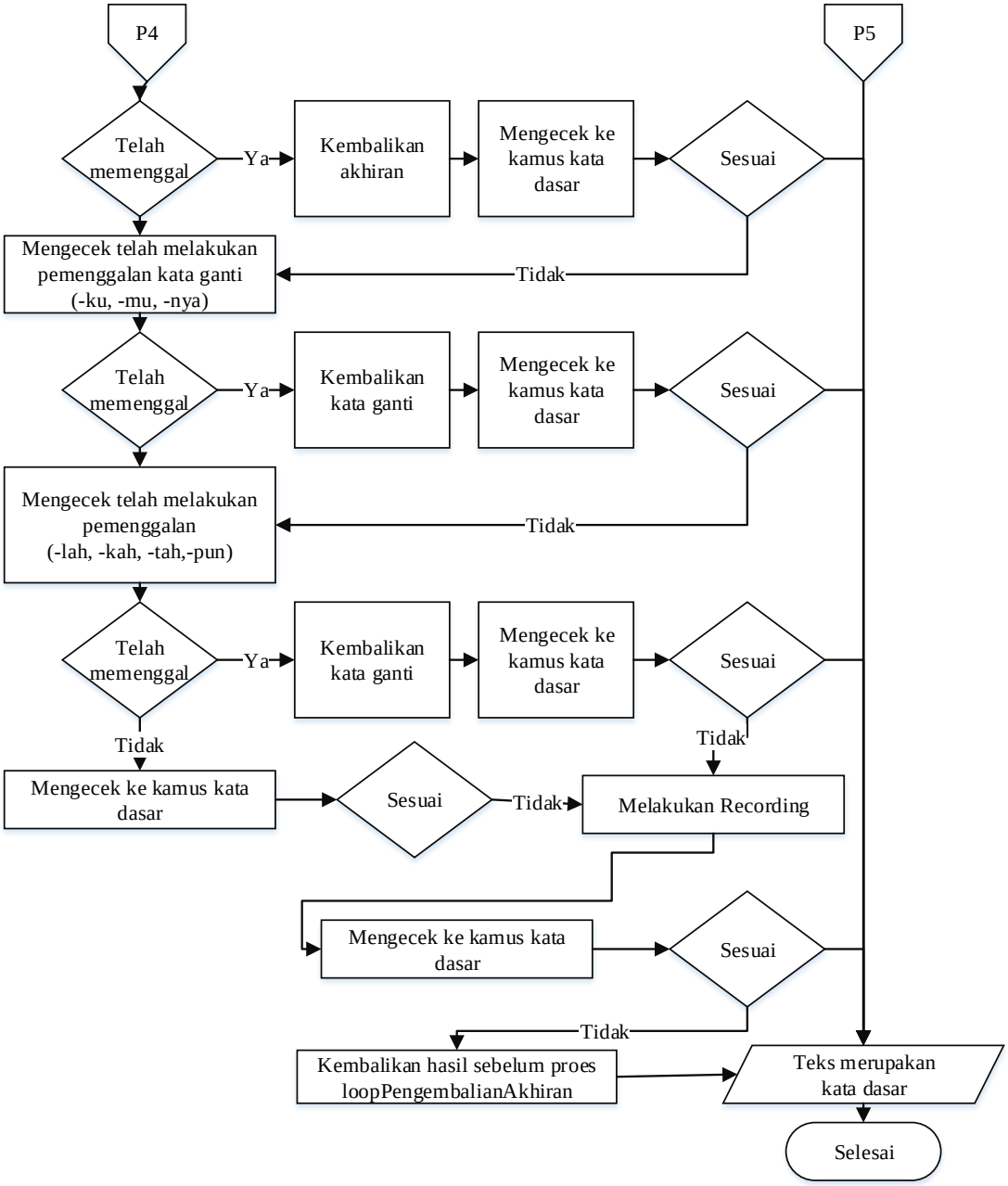
bentuk kata dasarnya adalah kata berbahasa indonesia dan metode *stemming* ini merupakan metode perbaikan dari metode *confix stripping stemmer* seperti yang telah disebutkan pada bab sebelumnya sehingga diharapkan akan meningkatkan keberhasilan dalam mengembalikan kata kebentuk dasarnya. *Enchanced confix stripping stemmer* adalah metode *stemming* yang membutuhkan daftar kata dasar dalam mencari bentuk kata dasar dari sebuah kata. Daftar kata dasar dalam penelitian ini diambil dari KBBI online dan kamusbesar.com. Untuk *flowchart* Stemming dengan metode *enchanced confix stripping stemmer* dapat dilihat pada Gambar 4.9.



Gambar 4.9 *Flowchart Stemming*



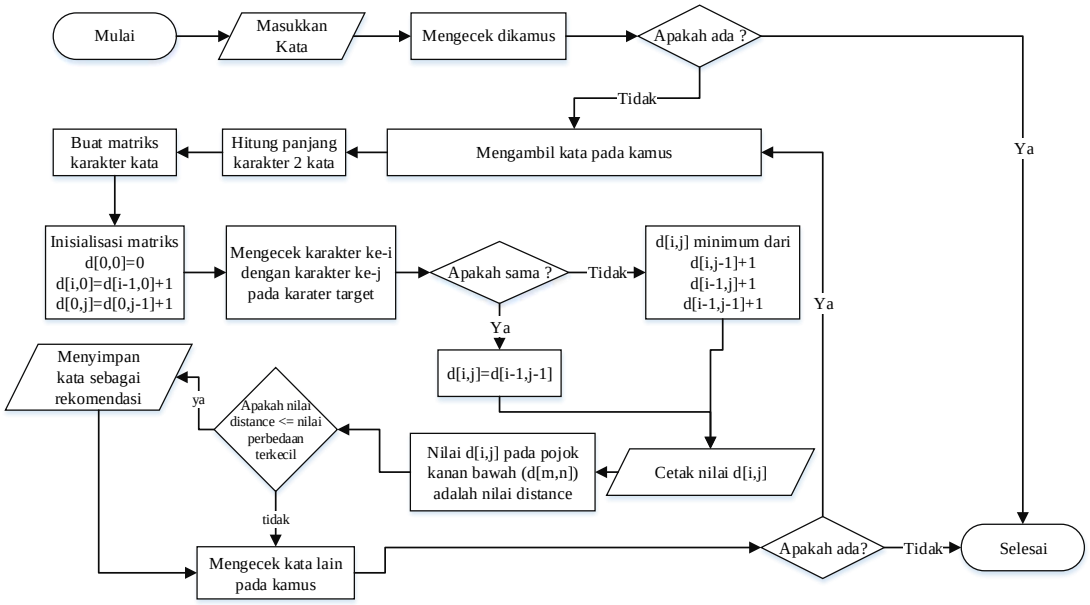
Gambar 4.9 *Flowchart Stemming* (Lanjutan)



Gambar 4.9 *Flowchart Stemming* (Lanjutan)

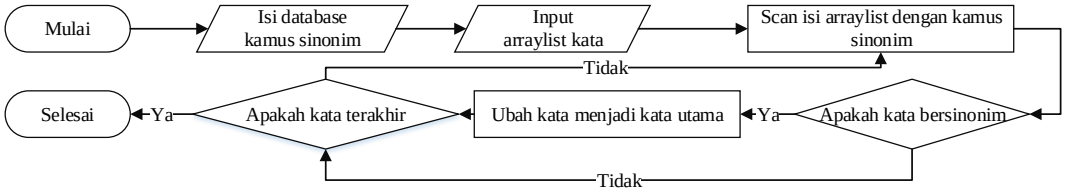
Daftar kata jawaban pelajar hasil proses *stemming* kemudian dilakukan pengorkesian ejaan (*spelling correction*). *Spelling correction* dilakukan untuk memberikan daftar kata rekomendasi perbaikan ejaan (penulisan) kata yang dianggap salah oleh sistem. Kata yang dianggap salah ejaannya adalah kata yang tidak ada pada kamus kata dasar, kamus sinonim, dan daftar kunci jawaban untuk soal tersebut. Proses pengoreksian ejaan hanya dilakukan pada daftar kata jawaban pelajar. Proses *spelling correction* dilakukan dengan metode *levenshtein distance*. Metode tersebut

melakukan pengoreksian ejaan dengan cara menghitung perbedaan dua kata (*string*) dengan mempertimbangkan adanya penambahan penambahan (*insert*), penghapusan (*delete*), atau penggantian (*substitute*) karakter. Kata yang akan direkomendasikan pada proses ini adalah kata yang terdapat pada kamus kata dasar, kamus sinonim, dan kata pada kunci jawaban yang memiliki perbedaan paling kecil. Satu kata daftar kata dapat memiliki lebih dari satu rekomendasi kata yang benar, penentuan kata yang akan dipilih sebagai perbaikan dari kata dalam daftar kata jawaban pelajar dilakukan pada proses *identification*. Gambar 4.10 adalah gambaran proses pengoreksian ejaan.



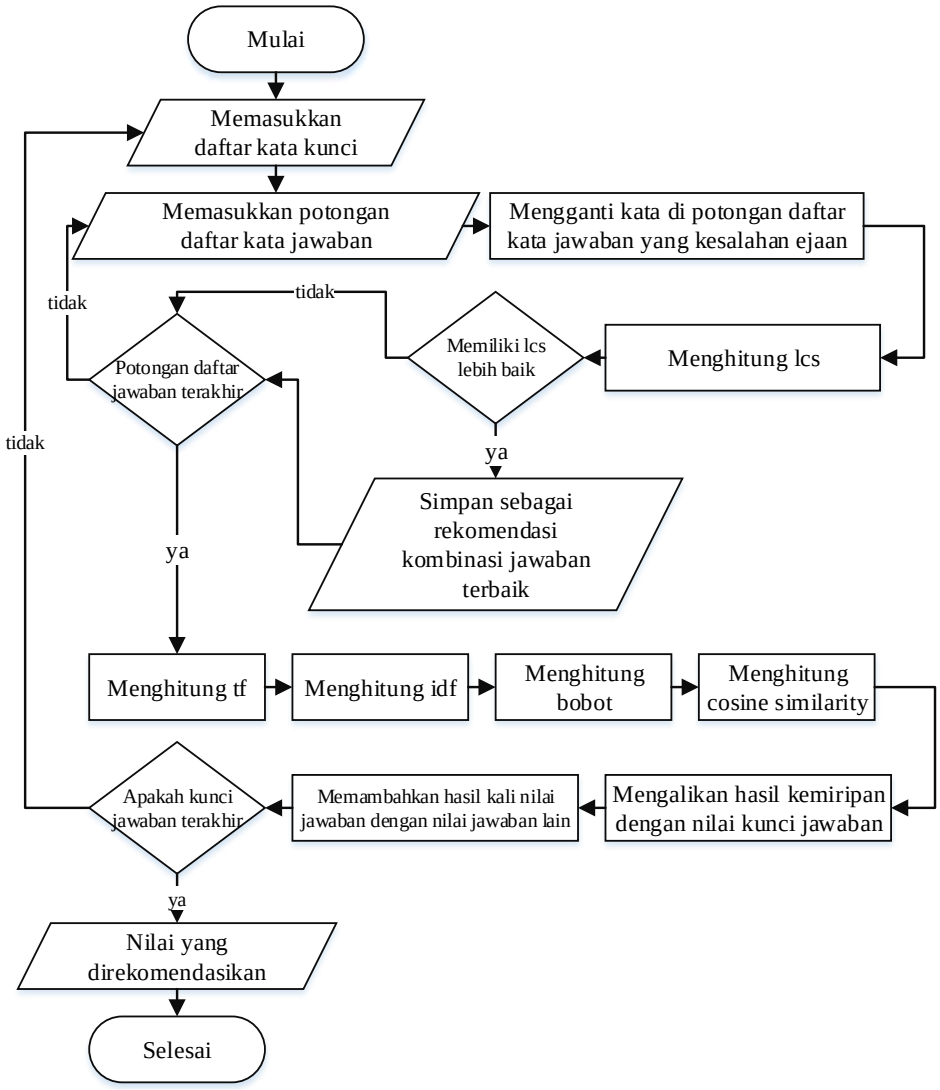
Gambar 4.10 *Flowchart Spelling Correction*

Proses terakhir dari tahap *feature selection* adalah pencarian sinonim (*synonym recognition*). Masing-masing kata yang ada dalam daftar kata (baik daftar kata kunci jawaban maupun daftar kata jawaban) dicari dalam kamus sinonim, jika kata tersebut ada dalam kamus sinonim maka kata tersebut akan diganti dengan kata utamanya, gambaran proses *synonym recognition* ini ditunjukkan pada Gambar 4.11.



Gambar 4.11 *Flowchart Synonym Recognition*

Tujuan *synonym recognition* adalah untuk mengurangi keberagaman kata pada daftar kata dengan cara mengubah kata yang bersinonim dengan kata utama, proses ini diharapkan akan memperkecil keberagaman kata sehingga dapat meningkatkan tingkat akurasi di tahap *identification*.



Gambar 4.12 *Flowchar*t Proses *Identification*

Hasil dari tahap *feature selection* akan digunakan sebagai masukan dalam tahap *identification*. Pada tahap ini dilakukan perhitungan kesamaan urutan karakter dengan *longest common subsequense* untuk menentukan kombinasi terbaik antara penggalan jawaban dengan setiap kunci jawaban yang terdapat pada pedoman penskoran. Penggalan jawaban tersebut adalah daftar kata jawaban yang dikelompokkan dan disusun kembali berdasarkan baris baru (*enter*), titik, koma, dan

spasi. Kombinasi terbaik adalah kombinasi yang memiliki kesamaan urutan karakter paling panjang (banyak). Hasil kombinasi terbaik kemudian dihitung bobot setiap kata dengan *term frequency-inverse document frequency* (tf-idf). Hasil tf-idf digunakan sebagai masukan untuk menghitung kemiripan pada *cosine similarity*. Besar kemiripan tersebut kemudian dikalikan dengan skor pada masing-masing kunci jawaban dan hasil akhirnya merupakan nilai rekomendasi dari sistem yaitu penjumlahan dari semua skor yang didapatkan. Untuk alur pada tahap *identification* secara lebih jelas dapat dilihat pada Gambar 4.12.

4.2.5 Pengujian

Pengujian dilakukan untuk mengetahui performa sistem. Pengujain dilakukan dengan menghitung akurasi dan *root mean square error* (RMSE). Akurasi digunakan untuk menghitung keakuratan nilai yang direkomendasikan oleh sistem. Semakin banyak nilai rekomendasi sistem yang sama dengan nilai sebenarnya (penilaian oleh pengajar) menunjukkan semakin akuratnya hasil dari sistem. Akurasi sistem didapatkan dengan menggunakan persamaan (4.1) dibawah ini (Hamzah, 2014) :

$$\text{Akurasi} = \frac{\text{Jumlah Data Benar}}{\text{Jumlah Data}} \text{ x } 100\% \tag{4.1}$$

RMSE digunakan untuk mengetahui nilai error dari nilai yang direkomendasikan oleh sistem terhadap nilai sebenarnya. Semakin kecil nilai RMSE menunjukkan semakin baik nilai yang direkomendasikan oleh sistem. RMSE dihitung dengan persamaan (Murray, 2018) :

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(A-F)^2}{n}} \tag{4.2}$$

Dengan :
A = Nilai seharusnya (penilaian oleh pengajar)
F = Nilai rekomendasi sistem
n = Jumlah data

4.2.6 Analisa dan Penarikan Kesimpulan

Tahap ini adalah tahap terakhir dari pelaksanaan penelitian. Pada tahap ini, hasil pengujian kemudian dianalisa tingkat akurasinya. Dari hasil analisa tersebut kemudian diambil kesimpulan sebagai hasil akhir dari penelitian dan sebagai jawaban dari rumusan masalah dan tujuan yang terdapat pada bab pendahuluan.

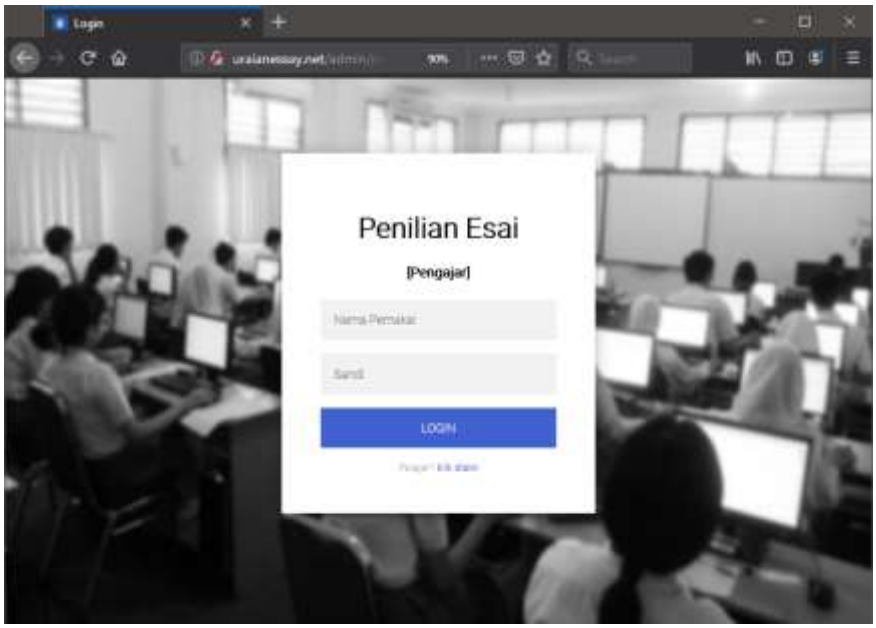
BAB V
HASIL DAN PEMBAHASAN

Bab sebelumnya telah membahas tentang metode penelitian dan gambaran perancangan sistem. Pada bab ini akan berisi pembahasan hasil proses setiap tahapan yaitu *input* data, *text preprocessing*, *feature selection* dan *identification*. Serta pada bagian akhir bab akan membahas pengujian tingkat akurasi dari sistem.

5.1 Sistem Ujian Online

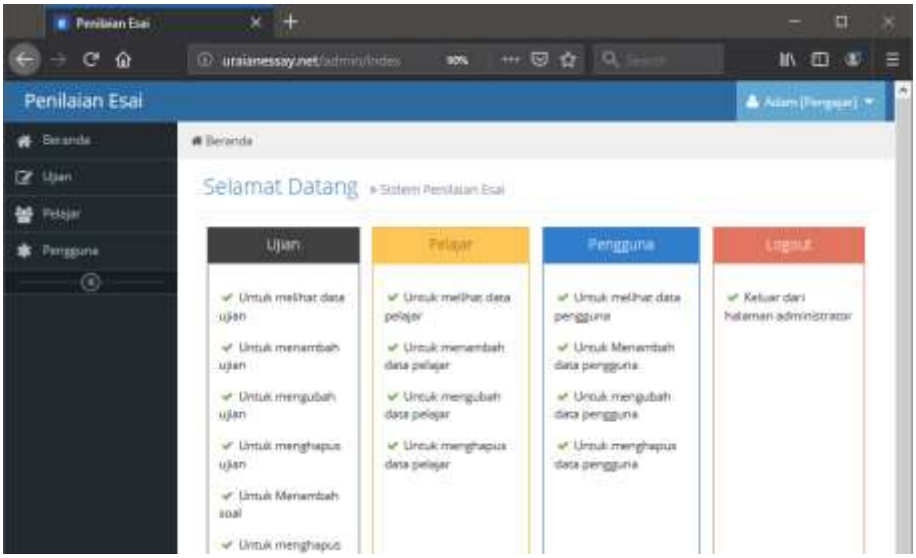
Sistem ujian *online* ini dibuat berbasis web dengan bahasa pemrograman PHP. Sistem ini dibuat khusus untuk melakukan ujian *online* untuk soal-soal jenis esai. Sistem ujian *online* ini dapat diakses pada alamat "http://uraianessay.net/". Dalam sistem terdapat dua hak akses yaitu pengajar (sebagai admin) dan pelajar (sebagai peserta ujian).

Untuk menggunakan sistem ujian *online* dengan hak akses pengajar dilakukan melalui alamat "http://uraianessay.net/admin" dan pengajar akan ditampilkan halaman *login* seperti Gambar 5.1, dimana pada halaman tersebut pengajar yang harus mengisi dengan nama pemakai dan sandi terlebih dahulu.



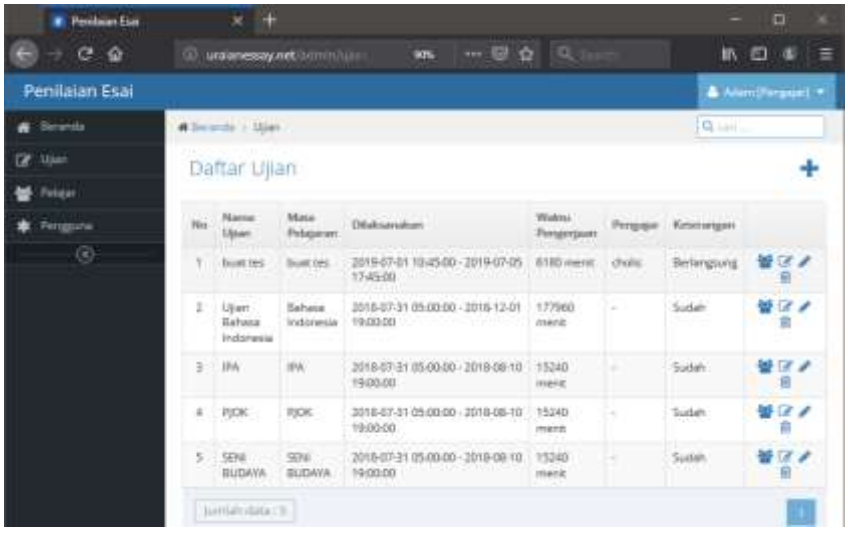
Gambar 5.1 Halaman *Login* Pengajar

Hak akses pengajar memiliki hak untuk mengelola data ujian, mengelola data soal tiap ujian, mengelola data kunci jawaban tiap soal, selain itu hak akses pengajar atau admin juga dapat mengelola pengguna sistem ujian online yang termasuk didalamnnya data pengajar atau admin dan data peserta didik. Halaman Beranda hak akses pengajar dapat dilihat seperti Gambar 5.2.



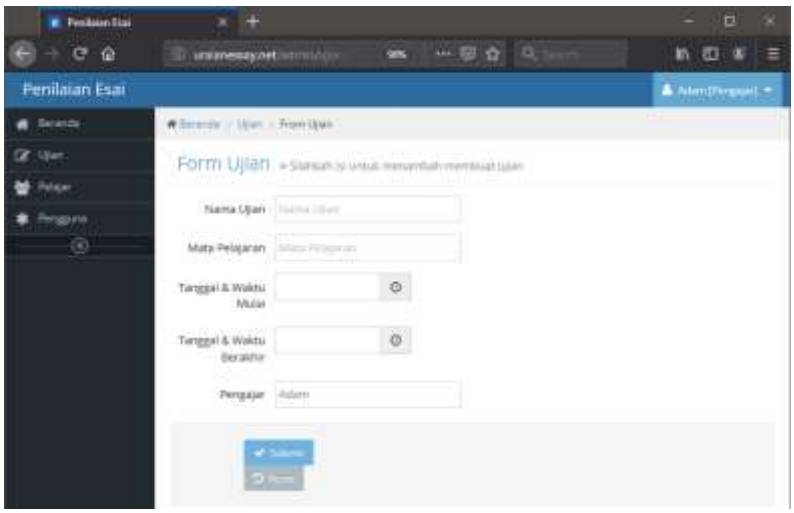
Gambar 5.2 Halaman Beranda Pengajar

Untuk mengelola data ujian, pengajar harus memilih menu ujian dan sistem akan menampilkan halaman data ujian seperti pada Gambar 5.3. Pada halaman tersebut menampilkan semua daftar data ujian baik yang sedang berlangsung, sudah dan belum dilaksanakan



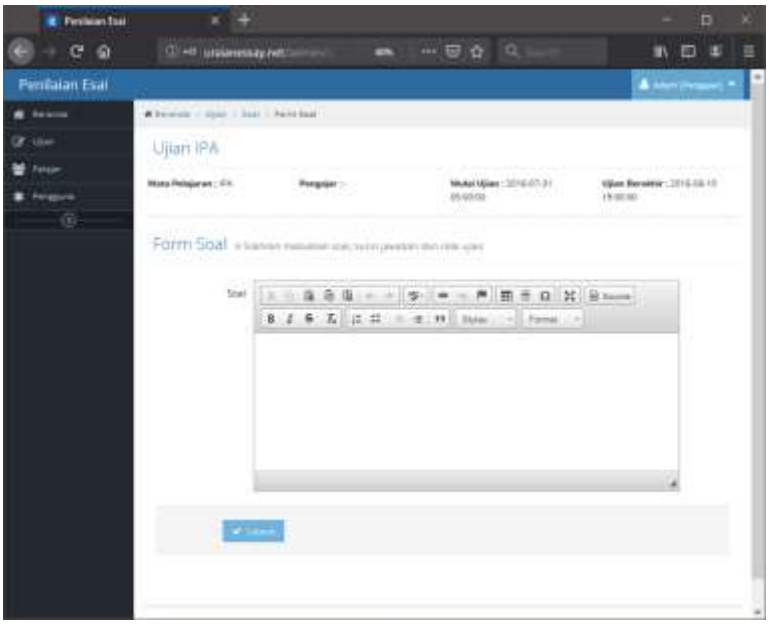
Gambar 5.3 Halaman Data Daftar Ujian

Data ujian dimasukkan melalui menu tambah ujian pada halaman data ujian. Halaman tambah ujian seperti pada Gambar 5.4. Untuk menambahkan data ujian pengajar harus memasukkan nama ujian, mata pelajaran, tanggal dan waktu ujian dimulai dan tanggal dan waktu ujian berkahir.



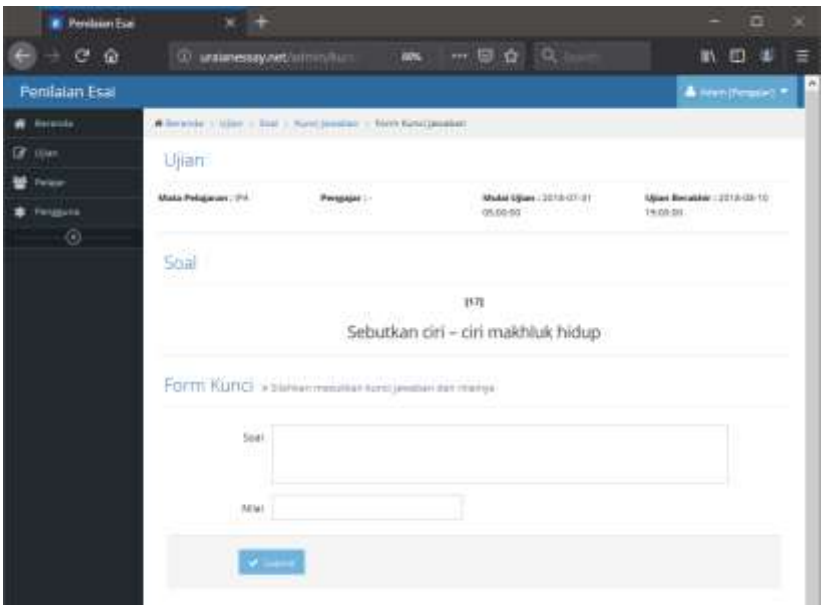
Gambar 5.4 Halaman Tambah Ujian

Data soal dimasukkan setelah pengajar menambahkan data ujian. Untuk menambahkan data soal dilakukan dengan memilih menu soal pada ujian yang telah dibuat. Gambar 5.5 memunjukkan halaman tambah soal.



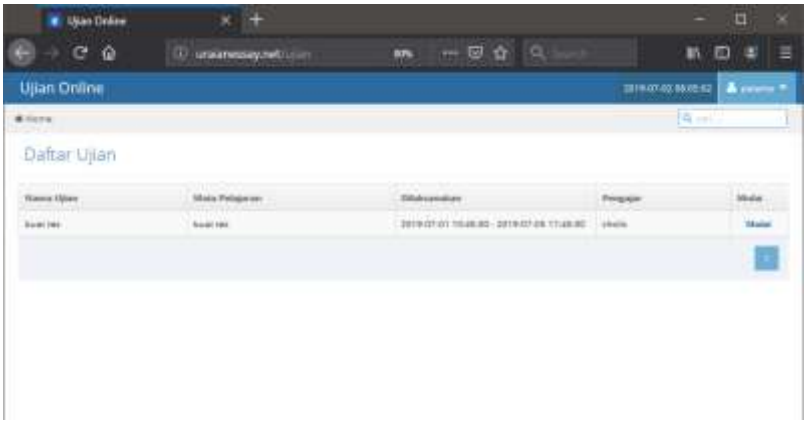
Gambar 5.5 Halaman Tambah Soal

Setelah menambahkan data soal, pengajar juga harus memasukkan kunci jawaban untuk tiap soal yang ditambahkan. Kunci jawaban yang dimasukkan adalah dalam bentuk pendoman penskoran dimana setiap kunci jawaban memiliki nilai. Halaman untuk menambahkan kunci jawaban seperti yang ditunjukkan gambar 5.6.



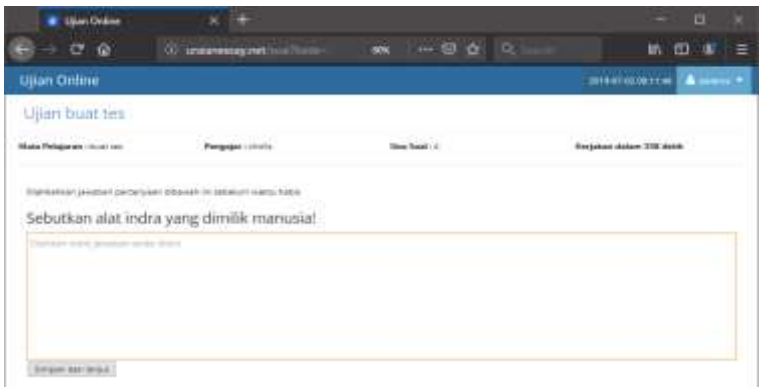
Gambar 5.6 Halaman Tambah Kunci Jawaban

Data jawaban dimasukkan oleh pelajar (peserta) dalam sistem ujian online pada alamat “http://uraianessay.net” disaat mengikuti ujian. Untuk mengikuti ujian pelajar harus login terlebih dahulu kemudian memilih ujian yang akan diikuti. Data ujian di halaman beranda pelajar akan tampil sesuai jadwal ujian akan dilaksanakan seperti yang ditunjukkan pada Gambar 5.7.



Gambar 5.7 Halaman Beranda Pelajar

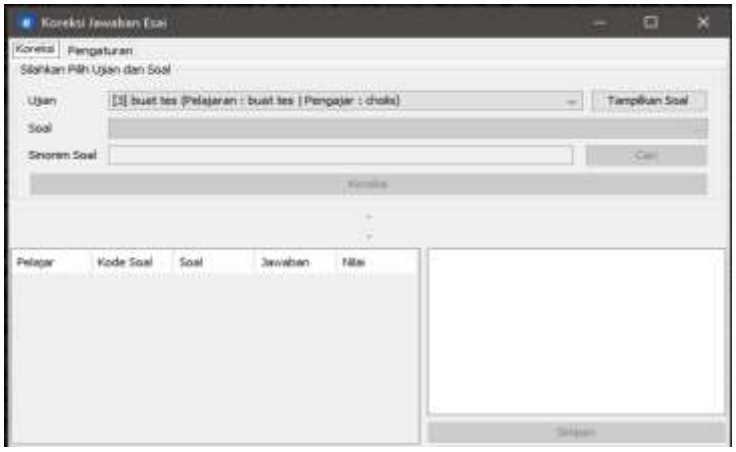
Pada halaman berada pelajar, pelajar harus memilih ujian yang akan diikuti dengan memilih menu mulai. Setelah itu sistem akan memunculkan halaman soal yang berisi soal dan form untuk menjawab soal. Tampilan halaman soal ujian dapat dilihat pada Gambar 5.8.



Gambar 5.8 Halaman Soal Ujian

5.2 Sistem Penilaian Jawaban Esai

Sistem penilaian jawaban esai ini dibuat terpisah dari sistem ujian *online*. Sistem penilaian jawaban esai dibuat berbasis desktop dengan bahasa pemrograman Java. Data yang dibutuhkan untuk melakukan penilaian berupa jawaban pelajar dan data jawaban yang berupa pedoman penskoran diambil dari *database* yang sama dengan sistem ujian *online*. Selain dua data tersebut dibutuhkan juga daftar sinomin kata yang juga harus masukkan sebelum melakukan penilaian jawaban. Gambar 5.9 menunjukkan tampilan sistem penilaian jawaban esai.



Gambar 5.9 Tampilan Sistem Penilaian Jawaban Esai

5.3 Hasil Penilaian Jawaban Esai

5.3.1 Hasil *Text Preprocessing*

5.3.1.1 Hasil *Case Folding* (*toLowerCase*)

Seperti yang telah dibahas pada bab sebelumnya, *case folding* adalah proses pertama pada tahap *text preprocessing*. *Case folding* digunakan untuk mengubah keseluruhan karakter pada teks dalam data menjadi suatu bentuk standar yaitu berhuruf kecil. Gambar 5.10 menunjukan fungsi untuk mengubah kalimat *input* menjadi huruf kecil. Pengubahan *input* menjadi huruf kecil dilakukan dengan fungsi "toLowerCase".

```
public class case_folding {
    public String getKalimatlower(String kalimat) {
        kalimat = kalimat.toLowerCase();
        return kalimat;
    }
}
```

Gambar 5.10 Fungsi *Case Folding*

Tabel 5.1 adalah hasil proses *case folding* pada salah satu jawaban untuk soal ujian IPA yaitu "sebutkan ciri – ciri makhluk hidup ?". Pada Tabel 5.1 tersebut dapat dilihat bahwa semua karakter huruf besar pada jawaban dari pelajar diubah menjadi huruf huruf kecil.

Tabel 5.1 Hasil Proses *Case Folding* Pada Jawaban Pelajar

Sebelum	Sesudah
Bergerak, Iritabilitas, Memerlukan Makanan, Respirasi, Tumbuh, Reproduki, Beradaptasi dan Ekkresi.	bergerak, iritabilitas, memerlukan makanan, respirasi, tumbuh, reproduki, beradaptasi dan ekkresi.

Case folding juga dilakukan pada setiap kunci jawaban yang ada pada pedoman penskoran. Pada soal "sebutkan ciri – ciri makhluk hidup ?" tersebut terdapat 8 kunci jawaban yang menjadi poin penentu penilaian. Proses *case folding* pada setiap kunci jawaban tersebut dilakukan secara terpisah untuk mempermudah dalam membandingkan kesamaan dengan jawaban pelajar nantinya. Tabel 5.2 adalah hasil proses *case folding* pada kunci jawaban.

Tabel 5.2 Hasil Proses *Case Folding* Pada Kunci Jawaban

Kunci Jawaban	Sebelum	Sesudah
1	Bergerak	bergerak
2	Iritabilitas	iritabilitas
3	Memerlukan Nutrisi	memerlukan nutrisi
4	Bernapas	bernapas
5	Tumbuh	tumbuh
6	Berkembang Biak	berkembang biak
7	Beradaptasi	beradaptasi
8	Ekskresi	ekskresi

5.3.1.2 Hasil *Tokenizing*

Pada proses *tokenizing* dilakukan penguraian kalimat-kalimat pada data menjadi daftar kata (kata-kata yang menyusun kalimat-kalimat pada data) dengan memperhatikan baris baru (*enter*), titik, koma, spasi, dan kata penghubung atau simbol yang menganti kata penghubung tersebut seperti dan, &, atau, /. Untuk melakukan *tokenizing* tersebut terlebih dahulu dilakukan perubahan pada semua tanda baca menjadi spasi dengan fungsi "replaceAll("\\p{Punct}|[a-d1-7]\\)|\\s", " ")". Kemudian dilakukan penguraian. Fungsi proses tokenizing dapat dilihat pada Gambar 5.11.

```
public ArrayList<String> getKata(String kalimat) {
    ArrayList<String> kata = new ArrayList();
    String temp = "";
    String hasilKalimat = kalimat.replaceAll("\\p{Punct}", " ");
    String hapusNewline = hasilKalimat.replace(System.getProperty("line.separator"), " ");
    StringTokenizer token = new StringTokenizer(hapusNewline, " ");
    while (token.hasMoreTokens()) {
        temp = token.nextToken();

    }
    return kata;
}
```

Gambar 5.11 Fungsi *Tokenizing*

Tabel 5.3 menunjukkan hasil dari proses *tokenizing* pada jawaban pelajar dan Tabel 5.4 menunjukkan hasil *tokenizing* pada kunci jawaban.

Tabel 5.3 Hasil Proses *Tokenizing* pada Jawaban Pelajar

Sebelum	**Sesudah**		
bergerak, iritabilitas, memerlukan makanan, respirasi, tumbuh, reproduki, beradaptasi dan ekkresi.	bergerak	iritabilitas	memerlukan
	makanan	respirasi	tumbuh
	reproduki	beradaptasi	dan
	ekkresi		

Tabel 5.4 Hasil Proses *Tokenizing* pada Kunci Jawaban

Kunci Jawaban	**Sebelum**	**Sesudah**	
1	bergerak	bergerak	
2	iritabilitas	iritabilitas	
3	memerlukan nutrisi	memerlukan	nutrisi
4	bernapas	bernapas	
5	tumbuh	tumbuh	
6	berkembang biak	berkembang	biak
7	beradaptasi	beradaptasi	
8	ekskresi	ekskresi	

Untuk *tokenizing* pada jawaban pelajar, terdapat proses tambahan yang dilakukan yaitu proses menyimpan susunan kata berdasarkan tanda baca. Proses ini dilakukan karena daftar kata akan dikelompokan dan disusun kembali berdasarkan baris baru (*enter*), titik, koma, dan spasi pada saat dilakukan perbandingan kemiripan dengan kunci-kunci jawaban ditahap *identification.* Gambar 5.12 menunjukkan fungsi tambahan *tokenizing*. Pada fungsi tambahan tersebut data yang disimpan adalah *index* awal dan *index* akhir dari tiap pengelompokan.

```
int awal_lama = 1000, akhir_lama = 1000;
private ArrayList<Integer> pis_awal = new ArrayList();
private ArrayList<Integer> pis_akhir = new ArrayList();

private void tambah(int awal, int akhir) {
    if (awal <= akhir) {
        if (awal_lama == awal && akhir_lama == akhir) {
        } else {
            pis_awal.add(awal);
            pis_akhir.add(akhir);
        }
        awal_lama = awal;
        akhir_lama = akhir;
    }

}
```

Gambar 5.12 Fungsi Tambahan *Tokenizing*

Sebagai contoh setelah untuk index awal dan index akhir dari tiap pengelompokan jawaban pelajar hasil *tokenizing* pada Tabel 5.3 dikonversi menjadi daftar kata didapatkan 12 hasil pengelompokan dan penyusunan kembali, sebagai berikut :

1. bergerak
2. iritabilitas
3. memerlukan
4. makanan
5. memerlukan makanan
6. respirasi
7. tumbuh
8. reproduki
9. beradaptasi
10. dan
11. ekkresi
12. bergerak iritabilitas memerlukan makanan respirasi tumbuh reproduki beradaptasi dan ekkresi

5.3.2 Hasil *Feature Selection*

5.3.2.1 Hasil *Stopword Removal (Filtering)*

Proses ini digunakan untuk membuang kata-kata tidak memiliki makna penting dan tidak akan berpengaruh makna sebuah data serta akan berpengaruh tidak baik pada tahap *identification*. Kata-kata yang akan dibuang adalah kata-kata yang terdapat dalam daftar *stopword*. Gambar 5.13 menunjukkan fungsi pengecekan apakah kata terdapat pada daftar *stopword.* Pada fungsi yang ditunjukkan pada Gambar 5.13, kata yang termasuk dalam daftar *stopword* akan mengembalikan nilai "*true*" sedangkan yang tidak ada dalam daftar stopword akan mengembalikan nilai "*false*".

```
public boolean cekstopword(String kata, ArrayList<String> liststopword) {
    if (liststopword.contains(kata)) {
        return true;
    } else {
        return false;
    }
}
```

Gambar 5.13 Fungsi Cek *Stopword*

Contoh pada jawaban pelajar Tabel 5.5 terdapat dua kata yang termasuk dalam daftar *stopword* yaitu "memerlukan" dan "dan". Kedua kata tersebut akan dibuang.

Tabel 5.5 Hasil Proses *Stopword Removal* pada Jawaban Pelajar

Sebelum			**Sesudah**		
bergerak	iritabilitas	memerlukan	bergerak	iritabilitas	makanan
makanan	respirasi	tumbuh	respirasi	tumbuh	reproduki
reproduki	beradaptasi	dan	beradaptasi		ekkresi
ekkresi					

Proses *stopword removal* tersebut akan mengubah susunan daftar kata. Untuk itu perlu juga dilakukan perubahan pada penyimpanan susunan kata dan hasilnya terdapat 9 pengelompokan dan penyusunan kembali seperti dibawah ini :

1. bergerak
2. iritabilitas
3. makanan
4. respirasi
5. tumbuh

6. reproduki
7. beradaptasi
8. ekkresi
9. bergerak iritabilitas makanan respirasi tumbuh reproduki beradaptasi ekkresi

Untuk proses *stopword removal* pada kunci jawaban dilakukan pada kunci jawaban 3. Pada kunci jawaban 3 terdapat kata yang termasuk *stopword* yaitu "memerlukan". Kata tersebut akan dibuang, sehingga untuk kunci jawaban 3 adalah "nutrisi". Hasilnya sebelum dan sesudah proses *stopword removal* pada kunci jawaban dapat dilihat pada Tabel 5.6.

Tabel 5.6 Hasil Proses *Stopword Removal* pada Kunci Jawaban

Kunci Jawaban	**Sebelum**		**Sesudah**	
1	bergerak		bergerak	
2	iritabilitas		iritabilitas	
3	memerlukan	nutrisi	nutrisi	
4	bernapas		bernapas	
5	tumbuh		tumbuh	
6	berkembang	biak	berkembang	biak
7	beradaptasi		beradaptasi	
8	ekskresi		ekskresi	

5.3.2.2 Hasil *Stemming*

Proses *stemming* dilakukan untuk menghilangkan variasi kata yang sebenarnya berasal dari kata dasar yang sama sehingga nantinya akan memperkecil jumlah indeks pada tahap selanjutnya. Pada penelitian ini *stemming* dilakukan dengan metode *enchanced confix stripping stemmer*. Langkah-langkah dalam *enchanced confix stripping stemmer* dapat dilihat pada subbab 2.9. Potongan fungsi *enchanced confix stripping stemmer* dapat dilihat pada Gambar 5.14.

```
public stemmer_ecs(ArrayList<String> listkatadasar) {
    this.listkatadasar = listkatadasar;
}
private boolean cekKamus(String kata) {
    return listkatadasar.contains(kata);
}
public String KataDasar(String kata) {
    this.akarKata = kata;
    this.asliKata = kata;
    if (cekKamus(kata) && kata.length() > 0) {
    } else {
        boolean rule1 = (akarKata.startsWith("be") && akarKata.endsWith("lah"));
        boolean rule2 = (akarKata.startsWith("be") && akarKata.endsWith("an") && !akarKata.endsWith("kan"));
        boolean rule3 = (akarKata.startsWith("me") && akarKata.endsWith("i"));
        boolean rule4 = (akarKata.startsWith("di") && akarKata.endsWith("i"));
        boolean rule5 = (akarKata.startsWith("pe") && akarKata.endsWith("i"));
        boolean rule6 = (akarKata.startsWith("te") && akarKata.endsWith("i"));
        if (((rule1) || (rule2) || (rule3) || (rule4) || (rule5) || (rule6)) == true) {
            hpsDS = akarKata;
            kondisiDerivationPrefiksA();
            if (cekKamus(akarKata) && akarKata.length() > 1) {
            } else {
                derivationPrefiksB();
                if (cekKamus(akarKata) && akarKata.length() > 1) {
                } else {
                    derivationPrefiksB();
                    if (cekKamus(akarKata) && akarKata.length() > 1) {
                    } else {
                        hapusInfleksionalSuffiks();
                        if (cekKamus(akarKata) && akarKata.length() > 1) {
                        } else {
                            hapusDerivationSuffiks();
                            akarKata = hpsDS;
                            if (cekKamus(akarKata) && akarKata.length() > 1) {
                            } else {
                                loopPengembalianAkhiran();
                            }
                        }
                    }
                }
            }
        } else {
            hapusInfleksionalSuffiks();
            if (cekKamus(akarKata) && akarKata.length() > 1) {
            } else {
                hapusDerivationSuffiks();
                if (cekKamus(hpsDS) && akarKata.length() > 1) {
                    akarKata = hpsDS;
                } else {
                    if (isHapusSuffix == true) {
                        kondisiDerivationPrefiksA();
                    } else {
                        derivationPrefiksB();
                    }
                    if (cekKamus(akarKata) && akarKata.length() > 1) {
                    } else {
                        derivationPrefiksB();
                        if (cekKamus(akarKata) && akarKata.length() > 1) {
                        } else {
                            derivationPrefiksB();
                            if (cekKamus(akarKata) && akarKata.length() > 1) {
                            } else {
                                loopPengembalianAkhiran();
                            }
                        }
                    }
                }
            }
        }
    }
    return akarKata;
}
```

Gambar 5.14 Fungsi *Enchanced Confix Stripping Stemmer*

Contoh Hasil proses *stemming* untuk jawaban pelajar dapat dilihat Tabel 5.7 adalah. Pada jawaban pelajar tersebut terdapat kata “bergerak”, “makanan”, dan “beradptasi” yang merupakan kata berimbuhan. Dari kata tersebut kemudian di*stemming* sehingga “bergerak” menjadi “gerak”, “makanan” menjadi “makan” dan “beradptasi” menjadi “adaptasi”.

Tabel 5.7 Hasil Proses *Stemming* pada Jawaban Pelajar

Sebelum			**Sesudah**		
bergerak	iritabilitas	makanan	gerak	iritabilitas	makan
respirasi	tumbuh	reproduki	respirasi	tumbuh	reproduki
beradaptasi		ekkresi	adaptasi		ekkresi

Sementara untuk proses *stemming* pada kunci jawaban, contohnya dilakukan pada kata “bergerak” menjadi “gerak” pada kunci jawaban 1, “bernapas” menjadi “napas” pada kunci jawaban 4, “berkembang” menjadi “kembang” pada kunci jawaban 6 dan “beradaptasi” menjadi “adaptasi” pada kunci jawaban ke 7. Untuk hasil proses *stemming* pada kunci jawaban dapat dilihat pada Table 5.8.

Tabel 5.8 Hasil Proses *Stemming* pada Kunci Jawaban

Kunci Jawaban	**Sebelum**		**Sesudah**	
1	bergerak		gerak	
2	iritabilitas		iritabilitas	
3	nutrisi		nutrisi	
4	bernapas		napas	
5	tumbuh		tumbuh	
6	berkembang	biak	kembang	biak
7	beradaptasi		adaptasi	
8	ekskresi		ekskresi	

5.3.2.3 Hasil *Spelling Correction*

Spelling correction digunakan untuk mengecek kata dan merekomendasikan perbaikan kata oleh sistem, jika terjadi kesalahan pengetikan. Proses *spelling correction* dilakukan dengan metode *levenshtein.* Pada proses ini setiap kata dalam

jawaban pelajar akan dicek apakah kata tersebut ada dalam daftar kata dasar, sinonim, dan kunci jawaban, jika tidak maka akan dilakukan proses pencarian kata yang memiliki perbedaan paling sedikit dengan kata dalam daftar kata dasar, sinonim, dan kunci jawaban. Dimana perbedaan dihitungan berdasarkan banyaknya operasi penambahan (*insert*), penghapusan (*delete*) dan penggantian (*substitute*) karakter sampai kedua kata tersebut sama. Pengecekan perbedaan ini dilakukan dengan metode *levenshtein* dan langkah-langkah dari metode *levenshtein* dapat dilihat pada subbab 2.11. Kata yang memiliki perbedaan terkecil yang akan direkomendasi sebagai kata pengganti. Gambar 5.15 menunjukkan proses perhitungan banyaknya operasi yang diperlukan untuk membuat 2 kata menjadi sama dengan metode *levenshtein*.

```
public int koreksi(char[] str1, char[] str2) {
    int temp[][] = new int[str1.length + 1][str2.length + 1];

    for (int i = 0; i < temp[0].length; i++) {
        temp[0][i] = i;
    }

    for (int i = 0; i < temp.length; i++) {
        temp[i][0] = i;
    }

    for (int i = 1; i <= str1.length; i++) {
        for (int j = 1; j <= str2.length; j++) {
            if (str1[i - 1] == str2[j - 1]) {
                temp[i][j] = temp[i - 1][j - 1];
            } else {
                temp[i][j] = 1 + min(temp[i - 1][j - 1], temp[i - 1][j], temp[i][j - 1]);
            }
        }
    }
    return temp[str1.length][str2.length];

}

private int min(int a, int b, int c) {
    int l = Math.min(a, b);
    return Math.min(l, c);
}
```

Gambar 5.15 Fungsi *Spelling Correction*

Tabel 5.9 menunjukkan hasil proses *spelling correction* pada jawaban pelajar, dimana pada jawaban pelajar tersebut terdapat dua kata yang salah pengetikannya yaitu “reproduki” dan “ekresi”. Dengan *spelling correction*, sistem akan merekomendasikan perbaikan kata-kata tersebut dengan kata dalam kamus yang memiliki perbedaan karakter paling kecil. Kata yang direkomendasikan untuk

mengganti “reproduki” adalah “reproduksi” dan “ekkresi” adalah “ekskresi”. Hasil secara keseluruhan dapat dilihat pada Tabel 5.9.

Tabel 5.9 Hasil Proses *Spelling Correction* pada Jawaban Pelajar

Sebelum			**Sesudah**		
bergerak	iritabilitas	makanan	gerak	iritabilitas	makan
respirasi	tumbuh	reproduki	respirasi	tumbuh	reproduksi
beradaptasi		ekkresi	adaptasi		ekskresi

5.3.2.4 Hasil *Synonym Recognition*

Synonym recognition dilakukan untuk mengenali kata yang memiliki sinonim dan merubah kata tersebut kekata utama. Kata yang memiliki sinonim adalah kata yang terdapat pada daftar sinonim. Fungsi untuk mengecek sebuah kata terdapat pada daftar sinonim ditunjukkan pada Gambar 5.16.

```
for (int ii = 0; ii < kataal.size(); ii++) {
    String ceksis = sinonim.get(kataal.get(ii));
    if (ceksis != null) {
        kata_sinonim.add(ceksis);
    } else {
        kata_sinonim.add(kataal.get(ii));
    }
}
```

Gambar 5.16 Fungsi *Synonym Recognition*

Proses ini dilakukan pada jawaban pelajar. Sebagai contoh kata “respirasi” memiliki sinonim kata “napas” dan kata “napas” adalah kata utama, maka pada jawaban pelajar kata “repirasi” akan diganti kata “napas”. Hal yang sama juga dilakukan pada kata “reproduksi” digantikan kata “kembang biak” sebagai kata utama. Tabel 5.10 menunjukkan hasil sebelum dan sesudah proses *Synonym Recognition* pada jawaban pelajar**.**

Tabel 5.10 Hasil Proses *Synonym Recognition* pada Jawaban Pelajar

Sebelum			**Sesudah**		
bergerak	iritabilitas	makanan	gerak	iritabilitas	makan
respirasi	tumbuh	reproduksi	napas	tumbuh	kembang
beradaptasi		ekskresi	biak	adaptasi	ekskresi

5.3.3 Hasil *Identification*

5.3.3.1 Hasil *Longest Common Subsequence*

Proses ini digunakan untuk mencari kombinasi terbaik antara pengemlompokkan jawaban dengan kunci-kunci jawaban. Pencarian kombinasi terbaik dilakukan dengan mencari urutan karakter terpanjang (banyak) menggunakan algoritma *longest common subsequence* (LSC). Jawaban yang memiliki urutan karakter terpanjang (banyak) dan kelompok kata tersebut memilki lebih sedikit jumlah karakter yang akan dipilih. Fungsi algoritma *longest common subsequence* (LSC) dapat dilihat pada gambar 5.17

```
public int lcsDynamic(char str1[], char str2[]) {
    int temp[][] = new int[str1.length + 1][str2.length + 1];
    int max = 0;
    for (int i = 1; i < temp.length; i++) {
        for (int j = 1; j < temp[i].length; j++) {
            if (str1[i - 1] == str2[j - 1]) {
                temp[i][j] = temp[i - 1][j - 1] + 1;
            } else {
                temp[i][j] = Math.max(temp[i][j - 1], temp[i - 1][j]);
            }
            if (temp[i][j] > max) {
                max = temp[i][j];
            }
        }
    }
    return max;
}
```

Gambar 5.17 Fungsi *Longest Common Subsequence*

Tabel 5.11 menunjukkan hasil perhitungan LSC untuk kunci jawaban “kembang baik”. Dari hasil pengelompokkan dan penyusunan kembali jawaban pelajar dibandingkan dengan kunci jawaban “kembang biak”, nilai LSC terbesar yang didapatkan adalah 12 dan jumlah karakter paling sedikit yang memiliki nilai LSC = 12 adalah jawaban pelajar “kembang biak”. Dari hasil Tabel 5.11 tersebut, kata yang digunakan untuk menjadi jawaban kunci jawaban “kembang biak” dan kombinasi terbaik adalah “kembang biak”. Proses ini juga dilakukan untuk setiap kata kunci sehingga masing-masing kunci jawaban memiliki kombinasi terbaik. Hasil tersebut kemudian menjadi imputan untuk perhitungan TF-IDF *Cosine Similarity***.**

Tabel 5.11 Hasil Perhitungan LCS untuk kunci jawaban “kembang biak”

Jawaban Pelajar	Jumlah Karakter	Nilai LCS
gerak	5	4
iritabilitas	12	4
makan	5	3
napas	5	2
tumbuh	6	2
kembang	7	7
biak	4	4
kembang biak	12	12
adaptasi	8	2
ekskresi	8	3
gerak iritabilitas makan napas tumbuh kembang biak adaptasi ekskresi	70	12

5.3.3.2 Hasil TF-IDF *Cosine Similarity*

Hasil dari proses LCS kemudian dilakukan perhitungan kemiripan jawaban dan kunci jawaban sesuai kombinasi dengan cara melakukan pembobotan kata menggunakan *term frequency inverse document frequency* (tf-idf) dan perhitungan kemiripan dilakukan dengan *cosine similarity*. Hasil kemiripan *cosine similarity* kemudian dikalikan dengan skor pada masing-masing kunci jawaban dan hasil akhirnya merupakan nilai rekomendasi dari sistem yaitu penjumlahan dari semua skor yang didapatkan.

Pertama-tama dilakukan perhitungan *term frequency* (tf) berdasarkan kemunculan kata pada jawban dan kunci jawaban. Fungsi untuk dapat dilihat pada Gambar 5.18. Dalam fungsi tersebut setiap kata dalam daftar kata jawaban dan daftar kunci akan dipanggil. Jika kata tersebut kata yang belum pernah terpanggil maka selain menambah *frequency* kemunculan juga dilakukan penambahan kata. Jika kata dipanggil sebelumnya maka hanya dilakukan penambahan *frequency* kemunculan.

```
for (int i = 0; i < jaw.size(); i++) {
    String kata = jaw.get(i);
    Integer jumlah = jawaban.get(kata);
    if (jawaban.containsKey(kata)) {
        jawaban.put(kata, jumlah + 1);
    } else {
        jawaban.put(kata, 1);
        term.add(kata);
    }

}
```

(a)

```
for (int i = 0; i < kun.size(); i++) {
    String kata = kun.get(i);
    Integer jumlah = kunci.get(kata);
    if (kunci.containsKey(kata)) {
        kunci.put(kata, jumlah + 1);
    } else {
        kunci.put(kata, 1);
    }
    if (term.contains(kata) == false) {
        term.add(kata);
    }
}
```

(b)

Gambar 5.18 (a) Fungsi *Term Frequency* Jawaban dan *(b) Fungsi Term Frequency* Kunci Jawaban

Setelah diketahui *term frequency* (tf) kata pada jawaban dan kunci jawaban, selanjutnya dihitung *inverse document frequency* (idf). Pada penelitian ini jumlah dokumen yang dibandingkan adalah 2 yaitu jawaban dan kunci jawaban. Perhitungan *inverse document frequency* (idf) ini dilakukan menggunakan persamaan (2.5). Untuk menghindari nilai 0 maka *inverse document frequency* (idf) ditambah 1. Fungsi untuk menghitung *inverse document frequency* (idf) dapat dilihat pada Gambar 5.19.

```
for (int i = 0; i < term.size(); i++) {
    int df = 0;
    if (jawaban.get(term.get(i)) != null) {
        df++;
    }
    if (kunci.get(term.get(i)) != null) {
        df++;
    }
    double idf = Math.log10((2 / df)) + 1;
}
```

Gambar 5.19 Fungsi *inverse document frequency* (idf)

Setelah diketahui *inverse document frequency* (idf) pada masing-masing kata. Kemudian dilanjutkan dengan perhitungan bobot masing-masing kata sesuai persamaan (2.10). Hasil perhitungan bobot ini yang akan menjadi masukan untuk perhitungan *cosine similarity*. Gambar 5.20 menunjukkan fungsi perhitungan bobot dengan *term frequency inverse document frequency* (tf-idf). Fungsi pertama adalah perhitungan bobot untuk kata pada jawaban dan yang kedua perhitungan bobot untuk kata pada kunci jawaban

```
double wq = idf * q;
double wd = idf * d;
```

Gambar 5.20 Fungsi bobot dengan *term frequency inverse document frequency* (tf-idf)

Setelah bobot masing-masing kata didapatkan maka selajutkan dilakukan perhitungan kemiripan dengan menggunakan *cosine similarity*. Perhitungan *cosine similarity* dilakukan menggunakan persamaan (2.11) dan (2.12). Fungsi untuk menghitung *cosine similarity* seperti pada Gambar 5.21.

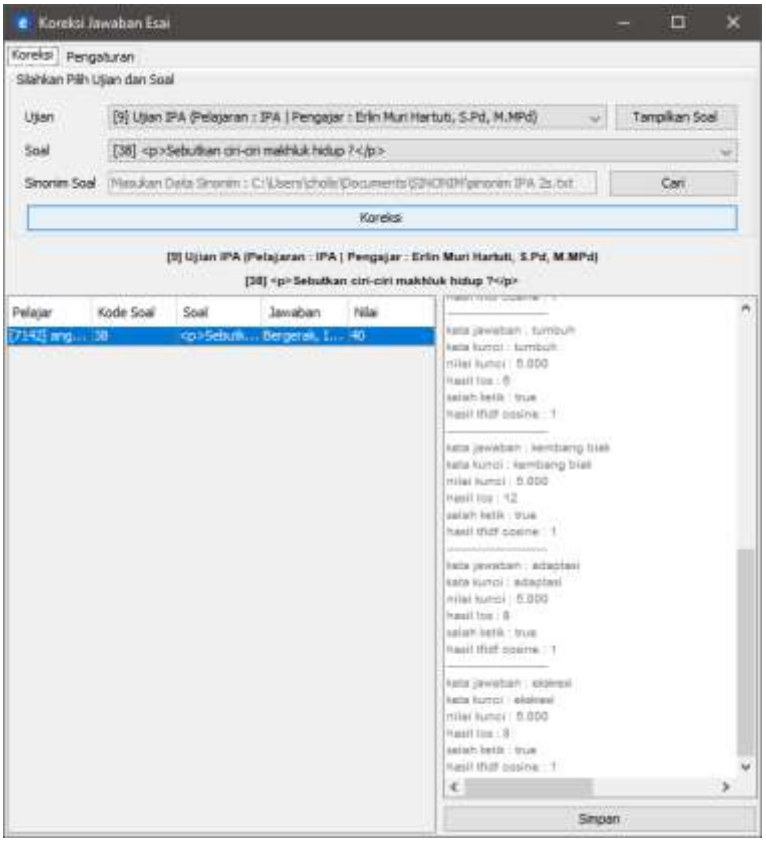
```
double wqwd = wq * wd;

double wq2 = Math.pow(wq, 2);
double wd2 = Math.pow(wd, 2);

cosine = sum_wq_wd / (Math.sqrt(sum_wq2) * Math.sqrt(sum_wd2));
```

Gambar 5.21 Fungsi *Cosine Similarity*

Gambar 5.22 menunjukan hasil penilaian jawaban esai untuk jawaban yang telah diproses pada subbab 5.3. Pada penilaian tersebut semua kata kunci pada pedoman penskoran memiliki nilai 5, sehingga nilai maksimal yang akan diperoleh adalah 40. Untuk contoh jawaban pada subbab 5.3 ini sistem penilian esai merekomendasikan nilai yang didapatkan adalah 8 seperti yang terlihat pada gambar 5.22.



Gambar 5.22 Contoh Hasil Penilaian Esai

5.4. Pengujian

Pengujian dilakukan untuk mengetahui performa dari sistem yang berupa akurasi dan RMSE. Perhitungan akurasi dilakukan dengan menggunakan persamaan (4.1) dan perhitungan RMSE dilakukan dengan menggunakan persamaan (4.2). Data yang digunakan untuk melakukan pengujian adalah data ujian di SMP Negeri 1 Paiton pada mata pelajaran bahasa indonesia, seni budaya dan IPA dengan jumlah soal masing-masing ujian adalah 5 soal yang masing-masing ujian tersebut diikuti oleh 24 pelajar. Sehingga dari setiap ujian akan terdapat sebanyak 120 jawaban. Pertanyaan esai dibatasi hanya untuk pertanyaan "sebutkan".

Pengujian pertama dilakukan dengan membandingan hasil rekomendasi nilai dari sistem yang menggunakan pedoman penskoran dan proses *synonym recognition* serta *spelling correction* dengan penilaian secara manual. Hasilnya menunjukkan bahwa sistem dapat menghasilkan rekomendasi dengan persentase rata-rata akurasi adalah 93,61 % dengan nilai RMSE sebesar 0,85. Nilai akurasi dan RMSE masing-masing ujian dapat dilihat pada Tabel 5.12 dan perhitungan detail perbandingan antar soal untuk setiap pelajar dapat dilihat pada lampiran.

Tabel 5.12 Akurasi dan RMSE Pengujian Sistem Penilaian Esai Berdasarkan Pedoman Penskoran menggunakan *Synonym Recognition* dan *Spelling Correction*

Ujian	**Jumlah nilai yang sama dengan penilaian manual pada setiap soal**					**Akurasi**	**RMSE**
	1	**2**	**3**	**4**	**5**		
IPA	24	19	22	24	22	92,50 %	1,02
Seni Budaya	23	24	24	21	17	90,83 %	0,75
Bahasa Indonesia	24	24	22	24	23	97,50 %	0,79
Rata-rata						93,61 %	0,85

Pengujian kedua dilakukan dengan membandingan penilaian secara manual dengan hasil rekomendasi nilai dari sistem menggunakan *synonym recognition* dan *spelling correction* tanpa pedoman penskoran. Dengan pengujian ini diharapkan dapat diketahui keefektifan dari penggunaan pedoman penskoran dalam sistem. Tabel 5.13 menunjukkan hasil akurasi dan RMSE sistem penilaian esai tanpa pedoman penskoran.

Dari Tabel 5.13 menunjukkan bahwa rata-rata akurasi 51,39 % dengan nilai RMSE sebesar 4,47. Hasil pengujian kedua ini lebih jelek dari hasil pengujian pertama.

Tabel 5.13 Akurasi dan RMSE Pengujian Sistem Penilaian Esai menggunakan *Synonym Recognition* dan *Spelling Correction* tanpa Pedoman Penskoran

Ujian	**Jumlah nilai yang sama dengan penilaian manual pada setiap soal**					**Akurasi**	**RMSE**
	1	**2**	**3**	**4**	**5**		
IPA	10	5	9	13	14	42,50 %	3,99
Seni Budaya	13	17	16	19	14	65,83 %	2,77
Bahasa Indonesia	5	7	16	9	18	45,83 %	6,65
Rata-rata						51,39 %	4,47

Pengujian ketiga dilakukan dengan membandingan hasil rekomendasi nilai dari sistem yang menggunakan pedoman penskoran dan tanpa proses *synonym recognition* serta tanpa proses *spelling correction* dengan penilaian secara manual. Pengujian ini dilakukan untuk mengetahui pengaruh penggunaan proses *synonym recognition* serta tanpa proses *spelling correction.* Hasilnya menunjukkan bahwa tanpa menggunakan *synonym recognition* serta *spelling correction* tingkat akurasi dengan rata-rata sebesar 58,33 % dan *RMSE* dengan rata-rata sebesar 4,65. Hasil Pengujian ketiga ini lebih jelek dari hasil pengujian pertama. Nilai akurasi dan RMSE masing-masing ujian dapat dilihat pada Tabel 5.14.

Tabel 5.14 Akurasi dan RMSE Pengujian Sistem penilaian esai berdasarkan pedoman penskoran tanpa menggunakan *synonym recognition* dan *spelling correction*

Ujian	**Jumlah nilai yang sama dengan penilaian manual pada setiap soal**					**Akurasi**	**RMSE**
	1	**2**	**3**	**4**	**5**		
IPA	22	1	14	18	7	51,67 %	5,97
Seni Budaya	6	14	21	14	5	50.00 %	4,05
Bahasa Indonesia	15	19	20	21	13	73,33 %	3,93
Rata-rata						58,33 %	4.65

5.5. Pembahasan

Pada penelitian ini, sistem telah mampu melakukan penilaian jawaban esai yang mengacu pada pedoman penskoran dengan menggunakan kombinasi metode *longest common subsequence* (LCS) dan *cosine similarity*. Untuk proses dari setiap tahap termasuk didalamnya yaitu mengkombinasikan metode *longest common subsequence* (LCS) dan *cosine similarity* untuk kemudian menghasilkan nilai dapat dilihat pada subbab 5.3.

Sistem penilaian jawaban esai menggunakan pedoman penskoran yang mengacu pada pedoman penskoran dengan menggunakan kombinasi metode *longest common subsequence* (LCS) dan *cosine similarity* dan tambahan proses *synonym recognition* dan *spelling correction* menghasilkan persentase rata-rata tingkat akurasi 93,61 % dengan nilai RMSE sebesar 0,85. Hasil tersebut menunjukkan bahwa sistem dapat menghasilkan rekomendasi nilai yang sangat baik dengan tingkat akurasi di atas 90% dan *root mean square error (RMSE)* mendekati 0 jika dibandingkan dengan penilaian jawaban secara manual. Kesalahan rekomendasi dikarenakan sistem belum dapat mendeteksi adanya kesalahan pengetikan yang berupa adanya spasi pada sebuah kata misalnya “Iri tabilitas” yang seharusnya “Iritabilitas” tanpa spasi dan kata-kata yang tidak dipisahkan oleh spasi misalnya “berkembangbiak” yang seharusnya “berkembang biak”, hal tersebut mengakibatkan kesalahan pada saat perhitungan kemiripan. Akan tetapi hasil penelitian ini menunjukkan adanya perbaikan akurasi dari hasil penelitian Fitri R. dan Asyikin A. N. (2015) yang memiliki kesesuaian nilai sistem rata-rata 89,48 %. Dimana pada penelitian Fitri R. & Asyikin A. N. (2015) tersebut objek yang diteliti adalah ujian esai ber-Bahasa Inggris tanpa pedoman penskoran dan belum ada proses *synonym recognition* dan *spelling correction.*

Hasil akurasi sistem yang menggunakan pedoman penskoran lebih baik dari sistem tanpa menggunakan pedoman penskoran. Hasil tersebut sebagaimana yang ditunjukan pada penggujian pertama dan pengujian kedua pada subbab 5.4. Pada pengujian kedua menunjukkan bahwa rata-rata akurasi 51,39 % dengan nilai RMSE sebesar 4,47. %, hasil tersebut memiliki akurasi lebih rendah 42,22 % dan RMSE lebih besar 3,62 dari hasil rekomendasi nilai dari sistem yang menggunakan pedoman penskoran. Hal tersebut disebabkan karena pada sistem penilaian esai tanpa pedoman

penskoran terdapat kata yang seharusnya tidak ikut dihitung dan menyebabkan semakin besarnya berbedaan dengan kunci jawaban.

Penambahan pengenalan sinonim (*synonym recognition*) dan pengoreksi ejaan (*spelling correction*) terbukti dapat meningkatkan tingkat akurasi hasil penilaian. Hasil penelitian dari Fitri R. & Asyikin A. N. (2015) yang menyebutkan bahwa kesalahan penilaian disebabkan karena kesalahan pengetikan yang menyebabkan sistem akan menganggap salah karena secara susunan huruf berbeda maka pada sistem pengoreksian esai perlu adanya pengoreksi ketikan atau ejaan (*spelling correction*). Algoritma yang dapat digunakan untuk pengoreksian ejaan adalah *levenshein distance* sebagaimana yang telah dilakukan oleh Braddley, M.O., Fachrurrozi, M & Yusliani, N. (2017). Pada penelitian yang dilakukan Darsono, A. D. B., & Wijono, S. H. (2014) menyimpulkan bahwa perbedaan nilai yang sangat mencolok antara nilai sistem dan koreksi manual disebabkan karena permasalahan makna kata atau sinomim. Saran Darsono, A. D. B., & Wijono, S. H. (2014) untuk perbaikan adalah menambahkan proses untuk mendeteksi sinonim kata atau menambahkan kata kunci. Dari saran penelitian tersebut, pada penelitian ini ditambahkan pengenalan sinonim (*synonym recognition*) dan pengoreksi ejaan (*spelling correction*). Hasilnya menunjukan bahwa dengan adanya *synonym recognition* serta *spelling correction* dapat meningkatkan akurasi dengan rata-rata sebesar 35,28 % dan memperkecil *RMSE* dengan rata-rata sebesar 3,80.

BAB VI
KESIMPULAN DAN SARAN

6.1 Kesimpulan

Berdasarkan hasil dan pembahasan pada bab sebelummnya, dapat ditarik kesimpulan sebagai berikut :

1. Sistem mampu melakukan penilaian jawaban esai yang mengacu pada pedoman penskoran dengan menggunakan kombinasi metode *longest common subsequence* (LCS) dan *cosine similarity*,
2. Sistem penilaian esai memiliki tingkat akurasi sebesar 93,61 % dan *root mean square error* (RMSE) sebesar 0,85 jika dibandingkan dengan penilaian jawaban secara manual,
3. Dengan penambahan pengenalan sinonim (*synonym recognition*) dan pengoreksi ejaan (*spelling correction*) dapat meningkatkan tingkat akurasi sebesar 35,28 % dan memperkecil *RMSE* sebesar 3,80.

6.2 Saran

Beberapa saran yang dapat dilakukan untuk penelitian selanjutnya adalah

1. Dilakukan perbaikan pada proses *spelling correction* untuk mendeteksi kesalahan pengetikan berupa adanya spasi pada sebuah kata misalnya “Iri tabilitas” yang seharusnya “Iritabilitas” tanpa spasi dan kata-kata yang tidak dipisahkan oleh spasi misalnya “berkembangbiak” yang seharusnya “berkembang biak”.
2. Dikembangkan untuk sistem penilaian esai selain pertanyaan “sebutkan” atau berupa soal perhitungan dan *code program*.

DAFTAR PUSTAKA

Arifin, Z. 2017. *Evaluasi Pembelajaran : Prinsip, Teknik, dan Prosedur*. Bandung : PT Remaja Rosdakarya.

Asian, J. 2007. *Effective Techniques for Indonesian Text Retrieval*. Thesis. Melbourne: RMIT University.

Astiti, K. A. 2017. *Evaluasi Pembelajaran*. Yogyakarta : Penerbit Andi (Anggota IKAPI).

Berry, M. W. & Kogan, J. 2010. *Text Mining Aplication and Theory*. West Sussex : John Wiley & Sons Ltd.

Braddley, M. O., Fachrurrozi, M & Yusliani, N. 2017. *Pengoreksian Ejaan Kata Berbahasa Indonesia Menggunakan Algoritma Levensthein Distance*. Prosiding Annual Research Seminar 2017 Computer Science and ICT.

Crocetti, G. 2015. *Textual Spatial Cosine Similarity*. White Plains : Pace University Seidenberg School of CSIS.

Darsono, A. D. B., & Wijono, S. H. 2014. *Sistem Penilaian Essay Jawaban Berbahasa Indonesia dengan Metode K-Nearest Neighbor (k-NN) dan Latent Semantic Analysis*. Prosiding Seminar RiTekTra 2014.

Djafar, I. M. 2014. Deteksi *Kemiripan Dokumen Teks Menggunakan Algoritma Manber*. Program Studi S1 Teknologi Informasi Fakultas Ilmu Komputer Dan Teknologi Informasi Universitas Sumatera Utara.

Dwitiyastuti, R. N., Muttaqin, A. & Aswin M. 2013. *Pengoreksi Kesalahan Ejaan Bahasa Indonesia Menggunakan Metode Levenshtein Distance*. Jurnal Mahsasiswa Teknik Elektro Universitas Brawijaya, http://elektro.studentjournal.ub.ac.id/ Diakses tanggal 20 November 2017.

Feldman, R & Sanger, J. 2007. *The Text Mining Handbook : Advanced Approaches in Analyzing Unstructured Data*. New York : Cambridge University Press.

Fitri, R. & Asyikin, A. N. 2015. *Aplikasi Penilaian Ujian Essay Otomatis Menggunakan Metode Cosine Similarity.* Jurnal POROS TEKNIK, Volume 7 No. 2, Desember 2015 :54-105.

Haldar, R. & Mukhopadhyay, D. 2011. *Levenshtein Distance Technique in Dictionary Lookup Methods: An Improved Approach*. Calcutta : Web Intelligence & Distributed Computing Research Lab Green Tower.

Hamzah, A. 2014. *Sentiment Analysis Untuk Memanfaatkan Saran Kuesioner dalam Evaluasi Pembelajaran dengan menggunakan Naive Bayes Classifier (NBC)*. Yogyakarta : Prosiding Seminar Nasional Aplikasi Sains & Teknologi (SNAST) 2014.

Jody, Wibowo, A. T., & Arifianto, A. 2015. *Analsis dan Implementasi Algoritma Winnowing dengan Synonym Recognition pada Deteksi Plagiarisme untuk Dokumen Teks Berbahasa Indonesia.* Bandung : Teknik Informatika Fakultas Teknik Informatika Telkom University

Lahitani, A. R., Permanasari, A. E. & Setiawan, N.A. 2016. *Cosine Similarity To Determine Similarity Measure: Study Case In Online Essay Assessment*. Proceedings of 2016 4th International Conference on Cyber and IT Service Management, CITSM 2016.

Mahalakshmi, & Kavitha S. 2016. *Software Program Plagiarism Detection Using Longest Common Subsequence Method*. International Journal of Computer Techniques – Volume 3 Issue 4, July - Aug 2016.

Mahendra, I P. A. K., Arifin, A. Z. & Ciptaningtyas, H. T. 2008. *Penggunaan Algoritma Semut dan Confix Stripping Stemmer untuk Klasifikasi Dokumen Berita Berbahasa Indonesia.* Surabaya : Seminar Tugas Akhir Jurusan Teknik Informatika, Fakultas Teknologi Informasi Institut Teknologi Sepuluh Nopember (ITS).

Murray, P. W., Agard, B. & Barajas, M. A. 2018. *Forecast of individual customer's demand from a large and noisy dataset*. Computers & Industrial Engineering 118 (2018) 33–43 . ScienceDirect.

Mustapha, S. S.M.F.D., Idris, N. dan Abdullah, R. 2005. *Embedding Information Retrieval and Nearest-Neighbour Algorithm into Automated Essay Grading System*. Proceedings of the Third International Conference on Information Technology and Applications (ICITA'05).

Nazief, B. & Adriani, M. 1996. *Confix-Stripping: Approach to Stemming Algorithm for Bahasa Indonesia*. Faculty of Computer Science University of Indonesia.

Nicholas, F. 2016. *Identifikasi Tipe File Dari File Fragment Menggunakan Longest Common Subsequences (LCS)*. Medan : Fakultas Ilmu Komputer Dan Teknologi Informasi Universitas Sumatera Utara. http://repository.usu.ac.id/handle/123456789/49856 diakses tanggal 8 Mei 2018.

Ratna A. A. P., Sanjaya R., Wirianata T., & Purnamasari P. D. 2017. *Word Level Auto-correction for Latent Semantic Analysis Based Essay Grading System*. International Conference on Quality in Research (QiR) : International Symposium on Electrical and Computer Engineering 15 th, 24-27 July 2017.

Robertson, S. 2004. *Understanding Inverse Document Frequency: On theoretical arguments for IDF*. Journal of Documentation, Vol.60, no.5, pp. 503-520.

Saadah, M. N., Atmagi, R. W., Rahayu, D.S. & Arifin, A.Z. 2013. *Sistem Temu Kembali Dokumen Teks dengan Pembobotan Tf-Idf Dan LCS*. JUTI, Volume 11, Nomor 1, Januari 2013 : 17 – 20.

Saputra, W. S. J., Muttaqin, F. 2013. *Pengenalan Karakter pada Proses Digitalisasi Dokumen Menggunakan Cosine Similarity*. Seminar Nasional Teknik Informatika (SANTIKA) 2013, 18 September 2013, pp 51-56.

Sary, Y. N. E. 2015. *Buku Mata Ajar Evaluasi Pendidikan*. Yogyakarta : Penerbit Deepublish (Grup Penerbitan CV BUDI UTAMA).

Sugiyamto, Surarso, B. & Sugiharto, A. 2014. *Analisa Performa Metode Cosine dan Jacard pada Pengujian Kesamaan Dokumen*. Jurnal Masyarakat Informatika, Volume 5, Nomor 10.

Sumaryanta. 2015. *Pedoman Penskoran*. Indonesian Digital Journal of Mathematics and Education Volume 2 Nomor 3.

Tala, F. Z. 2003. *A Study of Stemming Efects on Information Retrieval in Bahasa Indonesia.* Institute for Logic, Language and Computation Universiteit van Amsterdam The Netherlands.

Tuffery, S. 2011. *Data Mining and Statistics for Decision Making*. West Sussex : John Wiley & Sons Ltd.

Weiss, S. M., Indurkhya, N., Zhang, T., Damerau, F.J. 2005. *Text Mining : Predictive Methods of Analyzing Unstructered Information*. New York : Springer Science Business Media, Inc.

Yusuf, A. M. 2015. *Esesmen dan Evaluasi Pendidikan : Pilar Penyedia Informasi dan Kegiatan Pengendalian Mutu Pendidikan*. Jakarta : Penerbit Kencana.