UNIVERSITAS INDONESIA

ANALISA PENERAPAN DISTRIBUSI NORMAL DAN *LEARNING VECTOR QUANTIZATION* DALAM SISTEM DETEKSI PLAGIARISME MAKALAH DWIBAHASA BERBASIS METODE *LATENT SEMANTIC ANALYSIS*

SKRIPSI

DARIEN JONATHAN WINATA
1206261806

FAKULTAS TEKNIK UNIVERSITAS INDONESIA
DEPARTEMEN TEKNIK ELEKTRO
TEKNIK KOMPUTER
DEPOK
DESEMBER
2015

UNIVERSITAS INDONESIA

ANALISA PENERAPAN DISTRIBUSI NORMAL DAN *LEARNING VECTOR QUANTIZATION* DALAM SISTEM DETEKSI PLAGIARISME MAKALAH DWIBAHASA BERBASIS METODE *LATENT SEMANTIC ANALYSIS*

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana Teknik

DARIEN JONATHAN WINATA
1206261806

FAKULTAS TEKNIK UNIVERSITAS INDONESIA
DEPARTEMEN TEKNIK ELEKTRO
TEKNIK KOMPUTER
DEPOK
DESEMBER
2015

HALAMAN PERNYATAAN ORISINALITAS

Skripsi ini adalah hasil karya saya sendiri,

dan semua sumber baik yang dikutip maupun dirujuk

telah saya nyatakan dengan benar.

Nama	**:** Darien Jonathan Winata
NPM	**:** 1206261806
Tanda Tangan	**:**
Tanggal	**:** 16 Desember 2015

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh	:	
Nama	:	Darien Jonathan Winata
NPM	:	1206261806
Program Studi	:	Teknik Komputer
Judul Skripsi	:	Analisa Penerapan Distribusi Normal dan *Learning Vector Quantization* Dalam Sistem Deteksi Plagiarisme Makalah Dwibahasa Berbasis Metode *Latent Semantic Analysis*

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Teknik pada Program Studi Teknik Komputer, Fakultas Teknik, Universitas Indonesia

DEWAN PENGUJI

Pembimbing	: Dr. Ir. Anak Agung Putri Ratna M.Eng.	(.....)
Penguji	:	(.....)
Penguji	:	(.....)

Ditetapkan di : Depok

Tnggal : 16 Desember 2015

KATA PENGANTAR

Penulis mengucapkan syukur kepada Tuhan yang Maha Esa, sebab hanya karena anugerah-Nya, penulis dapat menyelesaikan makalah ini dengan baik. Dalam kesempatan ini, penulis ingin mengucapkan terima kasih kepada pihak-pihak yang telah membantu terselesaikannya skripsi ini, yakni:

1. Ibu Dr. Ir. Anak Agung Putri Ratna M.Eng, selaku pembimbing skripi yang telah memberikan pengarahan, koreksi, dukungan, dan waktu selama penulis mengerjakan skripsi ini.
2. Bapak Boma Anantasatya Adhi S.T., selaku mentor teknik skripsi yang telah memberikan bantan dan arahan selama penulis mengerjakan skripsi.
3. Orang tua dan keluarga penulis yang telah memberikan dukugnan baik moril maupun materiil sehingga penulis dapat menyelesaikan skripsi ini.
4. Mardiyah, Vebryanti Siahaan, dan Dian Rismawati sebagai rekan pada riset ini.
5. Rekan sesama asisten laboratorium Jaringan Komputer 2011, 2012, dan 2013

Penulis berharap agar Tuhan yang Maha Esa memberkati semua pihak yang telah berkontribusi bagi penyelesaian skripsi ini, dan semoga skripsi ini dapat bermanfaat bagi pengembangan ilmu pengetahuan dan teknologi.

Depok, Desember 2015

Penulis

HALAMAN PERNYATAAN PERSETUJUAN

PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universitas Indonesia, saya yang bertanda tangan di bawah ini:

Nama	: Darien Jonathan Winata
NPM	; 1206261806
Program Studi	: Teknik Komputer
Departemen	: Teknik Elektro
Fakultas	: Teknik
Jenis Karya	: Skripsi

Demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Noneksklusif (*Non-exclusive Royalty Free Right*)** atas karya ilmiah saya yang berjudul:

ANALISA PENERAPAN DISTRIBUSI NORMAL DAN *LEARNING VECTOR QUANTIZATION* DALAM SISTEM DETEKSI PLAGIARISME MAKALAH DWIBAHASA BERBASIS METODE *LATENT SEMANTIC ANALYSIS*

Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalih media/formatkan, mengelola, merawat, dan mempublikasikan tugas saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenar-benarnya.

Dibuat di: Depok
Pada Tanggal: 16 Desember 2015

Yang Menyatakan

(Darien Jonathan Winata)

ABSTRAK

Nama	:	Darien Jonathan Winata
Program Studi	:	Teknik Komputer
Judul Skripsi	:	Analisa Penerapan Distribusi Normal dan *Learning Vector Quantization* Dalam Sistem Deteksi Plagiarisme Makalah Dwibahasa Berbasis Metode *Latent Semantic Analysis*
Pembimbing	:	D. Ir. Anak Agung Putri Ratna, M.Eng

Distribusi normal adalah salah satu jenis persebaran kelompok data yang didefinisikan berdasarkan rata-rata dan standar deviasi dari sekelompok data, yang dapat digunakan untuk mengelompokkan data berdasarkan posisinya terhadap standar deviasi dari kelompok data tersebut. *Learning Vector Quantization* adalah salah satu jenis *neural network* yang bisa mempelajari sendiri masukan yang ia terima kemudian memberi keluaran sesuai dengan masukan tersebut, dengan metode *supervised* dan *competitive learning*. Skripsi ini membahas penerapan dan analisa dari kedua sistem tersebut untuk menguji hasil deteksi plagiarisme oleh sistem deteksi plagiarisme berbasis *latent semantic analysis*, yang berasal dari program Simple-O. Beberapa modifikasi dilakukan untuk meningkatkan akurasi pengujian, antara lain dengan melakukan variasi parameter-parameter dari metode distribusi normal, yakni dengan mengubah batas standar deviasi maupun dengan mengubah koefisien pengali batas nilai pada standar deviasi tertentu, dimana hasilnya adalah standar deviasi maupun koefisien pengalinya berbanding lurus dengan aspek relevansi program (*recall*) namun tidak pada akurasi (*F-Measure*). Modifikasi juga dilakukan pada parameter percepatan belajar dari algoritma *learning vector quantization*, dimana hasilnya adalah parameter percepatan belajar berbanding terbalik dengan relevansi program maupun akurasi. Kemudian variasi dan analisa dilakukan pada tujuh jenis besaran hasil keluaran sistem deteksi plagiarisme berbasis *latent semantic analysis*, yakni *frobenius norm*, *slice*, dan *pad*, beserta kombinasinya, dimana hasilnya keberadaan *frobenius norm* diwajibkan untuk melakukan evaluasi kemiripan antara dua teks. Kemudian hasil pengujian menggunakan kedua metode digabungkan menggunakan operasi AND yang memberikan hasil yang beragam, dengan catatan perlunya keseimbangan antara *precision* dan *recall* dari masing pengujian yang akan dilakukan operasi AND untuk memberikan hasil yang baik.

Kata kunci: Latent Semantic Analysis, Learning Vector Quantization, Distribusi Normal, Plagiarisme, Simple-O

ABSTRACT

Nama	:	Darien Jonathan Winata
Program Studi	:	Computer Engineering
Judul Skripsi	:	Analysis of Normal Distribution and Learning Vector Quantization Implementation in Latent Semantic Analysis Based Bilingual Plagiarism Detection System
Pembimbing	:	D. Ir. Anak Agung Putri Ratna, M.Eng

Normal distribution is a type of data distributions which is defined from the average and standard deviation of the data cluster. It can be used to group datas based on its position from the standard deviation of the data cluster. Learning vector quantization is a type of neural networks that can learn from inputs it gets to give appropriate outputs, with supervised and competitive learning methods. This thesis discusses the implementation and analysis of both methods to verify the plagiarism detection results from detection plagiarism system based on latent semantic analysis, which is based on Simple-O program. Some modifications are made, such as by variating the parameters of normal distribution method, by changing the limits of standard deviation or by changing the factor of the number limit at a particular standard deviation. Both of them appear to be directly proportional to the relevance (*recall*), but not with accuracy (*F-Measure*). Modifications are also made at the learning acceleration parameters from the learning vector quantization algorithm, which sees the parameters being inversely proportional to both the relevance and accuracy. Then, variations and analysis are done to seven types of magnitude from the results of the plagiarism detection system, which are frobenius norm, slice, and pad, and their combinations, which suggest that frobenius norm is the most verifiable results, and must be included to be evaluated when text similarity analysis are conducted. Then, verification results using both methods are combined using AND operation which gives diverse results. However, it is needed to have a balance between precision and recall from each verifications to produce good results.

Keywords: Latent Semantic Analysis, Learning Vector Quantization, Normal Distribution, Plagiarism, Simple-O

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR GRAFIK

DAFTAR TABEL

BAB 1
PENDAHULUAN

1.1. Latar Belakang

Kemudahan akses informasi melalui internet pada bidang pendidikan di satu sisi memudahkan pelajar untuk mencari informasi dan bahan belajar. Namun, di sisi lain hal ini juga menyebabkan terjadinya tindakan tidak etis dengan melakukan tindakan penjiplakan karya-karya yang tersedia di internet, yang disebut sebagai plagiarisme, yakni tindakan mengambil tulisan dari orang lain [1]. Plagiarisme mencakup penjiplakan esai, karya tulis, maupun *paper*. Penjiplakan dapat dilakukan dengan berbagai modus, yakni secara langsung (tanpa perubahan) ataupun juga yang diikuti dengan mengubah bahasa dan tata cara penulisan.

Untuk itu, diperlukan tindakan-tindakan untuk mengurangi praktek plagiarisme, salah satunya dengan melakukan pengecekan pada karya ilmiah terhadap praktik plagiarisme untuk kemudian diberikan sanksi kepada penulis karya ilmiah yang terbukti sebagai hasil plagiarisme. Cara otomatis dilakukan dengan bantuan perangkat lunak pada komputer, atau disebut *computer assisted plagiarism detection*, yang dalam riset ini berfokus pada deteksi plagiarisme dua bahasa, dimana pengecekan dilakukan pada karya tulis berbahasa Indonesia dengan karya tulis referensi yang berbahasa Inggris.

Program yang dikembangkan pada riset ini menggunakan metode *Latent Semantic Analysis,* yakni metode berbasis semantik yang membuat representasi dokumen hanya berdasarkan kata dan frekuensi kemunculan kata tersebut. Karena tidak memperhatikan susunan, struktur, dan tata bahasa dari sebuah kata, algoritma *Latent Semantic Analysis* hanya membutuhkan proses translasi kata per kata, sehingga proses translasi yang digunakan relatif lebih mudah dibandingkan dengan metode deteksi plagiarisme yang berbasis tata bahasa atau yang lainnya.

Program yang dikembangkan pada riset ini, dengan metode *Latent Semantic Analysis*, akan menghasilkan sekumpulan angka. Penentuan apakah sebuah karya

tulis adalah plagiat dari karya tulis lainnya akan diputuskan dari angka-angka tersebut. Dalam menentukan plagiat atau tidaknya sebuah karya tulis, penulis menggunakan dua metode. Metode pertama yakni algoritma *Learning Vector Quantization* (LVQ), yaitu algoritma untuk melakukan pengelompokan data berdasarkan jarak antara vektor masukan dengan vektor klasifikasi (plagiat atau tidak). Metode kedua yakni distribusi normal, yaitu metode statistika yang dapat menentukan posisi sebuah data terhadap kelompoknya berdasarkan rata-rata dan standar deviasi dari kelompok data tersebut, dengan asumsi kelompok data tersebut memenuhi syarat-syarat distribusi normal (atau distribusinya tidak diketahui). Posisi data dari kelompoknya akan digunakan penulis untuk menentukan plagiat tidaknya sebuah karya tulis.

1.2. Tujuan Penelitian

Tujuan dari penulisan karya ilmiah ini adalah:

1. Melanjutkan sistem deteksi plagiarisme berbasis LSA yang telah ada sebelumnya agar dapat melakukan deteksi plagiarisme secara akurat dengan mengimplementasikan algoritma LVQ dan metode distribusi normal.
2. Menganalisa dan mengetahui parameter-parameter terbaik dalam implementasi algoritma LVQ dan metode distribusi normal.

1.3. Batasan Masalah

Sistem ini dirancang untuk dapat melakukan deteksi plagiarisme pada karya tulis berbahasa Indonesia yang mengacu pada karya tulis berbahasa Inggris. Berbagai modifikasi pada algoritma akan dilakukan untuk mendapatkan tingkat akurasi terbaik pada deteksi plagiarisme.

1.4. Metode Penelitian

Metode penelitian yang digunakan antara lain:

(1) Studi literatur, yaitu membaca materi yang berkaitan dengan plagiarisme, LSA, LVQ, dan distribusi normal.

(2) Perancangan program, yaitu algoritma LVQ dan distribusi normal untuk menentukan plagiat atau tidaknya sebuah karya tulis.
(3) Implementasi dan modifikasi program yang telah dirancang sebelumnya
(4) Analisa hasil pengujian implementasi program.

1.5. Sistematika Penulisan

Laporan ini terbagi dalam 5 bab, dimana pada tiap-tiap bab akan menjelaskan hal-hal berikut:

BAB 1 PENDAHULUAN

Bab ini menjelaskan tentang latar belakang, tujuan penulisan, batasan masalah, metode penelitian, dan sistematika penulisan.

BAB 2 DETEKSI PLAGIARISME DAN PENGELOMPOKAN DATA

Bab ini akan menjelaskan konsep dasar plagiarisme, algoritma deteksi plagiarisme dengan *Latent Semantic Analysis*, dan pengelompokan data menggunakan *Learning Vector Quantization* dan distribusi normal.

BAB 3 PROGRAM DETEKSI PLAGIARISME DAN PENGELOMPOKAN DATA HASIL KELUARAN PROGRAM

Bab ini akan menjelaskan tentang implementasi algoritma deteksi plagiarisme dengan *Latent Semantic Analysis*, pengelompokan data menggunakan *Learning Vector Quantization* dan distribusi normal, serta modifikasi-modifikasi yang akan dilakukan ada algoritma pengelompokan data.

BAB 4 HASIL UJI COBA DAN ANALISA

Bab ini akan menjelaskan tentang skenario implementasi algoritma dan hasil serta analisa dari hasil pengujian.

BAB 5 KESIMPULAN

Bab ini akan menyampaikan kesimpulan dari perancangan, implementasi, hasil pengujian serta analisa dari program deteksi plagiarisme.

BAB 2
DETEKSI PLAGIARISME DAN PENGELOMPOKAN DATA

2.1. Deteksi Plagiarisme Secara Otomatis

Plagiarisme dapat didefinisikan sebagai tindakan pencurian ide atau kata-kata atau hal lainnya yang terkait menjadi milik sendiri, menggunakan produksi orang lain tanpa memberikan kredit pada sumber, melakukan pencurian literatur, atau untuk memberikan ide yang baru dan orisinil yang diturunkan dari sumber lain. Dengan kata lain, plagiarisme adalah tindakan penipuan. Hal tersebut termasuk mencuri pekerjaan orang lain dan kemudian berbohong. [2]

Deteksi plagiarisme secara otomatis adalah proses untuk mengetahui indikasi plagiarisme pada sebuah karya tulis berbasus teks menggunakan perangkat lunak. Metode yang dapat digunakan untuk melakukan deteksi plagiarisme tekstual secara otomatis yaitu: [3]

- *Grammar-based*: metode ini berfokus kepada struktur tata bahasa dari dokumen.
- *Semantics-based*: metode ini menggunakan model *vector-space* untuk mendeteksi persamaan antara dua dokumen. Metode ini membuat "sidik jari" dari sebuah dokumen untuk kemudian dicocokkan dengan "sidik jari" dari dokumen lain.
- *Grammar-semantics hybrid*: metode ini merupakan perpaduan antara metode *grammar-based* dengan *semantics-based.*
- *External Plagiarism Detection*: metode ini menggunakan korpus referensi dari dokumen-dokumen yang isinya mungkin telah diplagiat.
- *Clustering*: metode ini digunakan untuk mengurangi waktu pencarian.

2.2. Latent Semantic Analysis

Latent Semantic Analysis (LSA) adalah teori dan metode yang digunakan untuk mengekstraksi dan melakukan representasi penggunaan kontekstual dari arti

kata-kata menggunakan komputasi statistik yang diterapkan pada kumpulan teks yang besar (Landauer and Dumais, 1997). [4]

Langkah-langkah dalam menjalankan metode LSA yaitu dengan melakukan representasi teks sebagai sebuah matriks dengan kata-kata sebagai baris dan dokumen (atau konteks lain) sebagai kolom yang disebut matriks *term-document*. Setiap *cell* berisi banyaknya sebuah kata pada baris keluar pada dokumen pada kolom terkait. Kemudian, setiap entri dimasukkan kepada transformasi awal untuk pembobotan. Berikut adalah contoh pembuatan matriks *term-document* sebagai bagian dari algoritma LSA:

Example of text data: Titles of Some Technical Memos

c1: *Human* machine *interface* for ABC *computer* applications
c2: A *survey* of *user* opinion of *computer system response time*
c3: The *EPS user interface* management *system*
c4: *System* and *human system* engineering testing of *EPS*
c5: Relation of *user* perceived *response time* to error measurement

m1: The generation of random, binary, ordered *trees*
m2: The intersection *graph* of paths in *trees*
m3: *Graph minors* IV: Widths of *trees* and well-quasi-ordering
m4: *Graph minors*: A *survey*

$\{X\} =$

	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	1	0	0	1	0	0	0	0	0
interface	1	0	1	0	0	0	0	0	0
computer	1	1	0	0	0	0	0	0	0
user	0	1	1	0	1	0	0	0	0
system	0	1	1	2	0	0	0	0	0
response	0	1	0	0	1	0	0	0	0
time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
survey	0	1	0	0	0	0	0	0	1
trees	0	0	0	0	0	1	1	1	0
graph	0	0	0	0	0	0	1	1	1
minors	0	0	0	0	0	0	0	1	1

r (human.user) = -.38

r (human.minors) = -.29

Gambar 2.1: Atas: Kumpulan *Passage*; Bawah: *Term-Document Matrix* Dari Kumpulan *Passage* Atas [4]

Kemudian, LSA akan melakukan *singular value decomposition* (SVD) pada matriks untuk analisa faktor. SVD mendekomposisi matriks menjadi perkalian dari tiga matriks, dimana salah satu dari tiga matriks hasil SVD adalah matriks diagonal

yang berisi nilai skala yang mengalikan matriks baris (matriks pertama) dan matriks kolom (matriks ketiga) kembali menjadi matriks awal. Teknik ini dapat digunakan untuk mengurangi dimensi dari matriks dengan memotong dimensi dari matriks diagonal, kemudian mengalikannya lagi dengan matriks lain (yang dimensinya juga telah dipotong) sehingga memunculkan kembali matriks yang dimensinya sama dengan dimensi matriks semula, namun dengan nilai yang berbeda sehingga membawa signifikansi yang berbeda dengan matriks sebelumnya. [4]

Metode asli LSA adalah dengan membandingkan dokumen yang tersimpan pada kolom dari kedua matriks tersebut dengan menggunakan koefisien korelasi *spearman's rho* (ρ), dengan rentang $-1 \leq \rho \leq 1$. Nilai *spearman's rho* yang tinggi menandakan korelasi yang sejalan antara kedua dokumen (fungsi yang dibentuk oleh kedua dokumen meningkat secara monoton), sedangkan nilai *spearman's rho* yang rendah menandakan korelasi yang berlawanan (fungsi yang dibentuk oleh kedua dokumen menurun secara monoton). [5]

$\{\hat{X}\} =$

	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	0.16	0.40	0.38	0.47	0.18	-0.05	-0.12	-0.16	-0.09
interface	0.14	0.37	0.33	0.40	0.16	-0.03	-0.07	-0.10	-0.04
computer	0.15	0.51	0.36	0.41	0.24	0.02	0.06	0.09	0.12
user	0.26	0.84	0.61	0.70	0.39	0.03	0.08	0.12	0.19
system	0.45	1.23	1.05	1.27	0.56	-0.07	-0.15	-0.21	-0.05
response	0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
time	0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
EPS	0.22	0.55	0.51	0.63	0.24	-0.07	-0.14	-0.20	-0.11
survey	0.10	0.53	0.23	0.21	0.27	0.14	0.31	0.44	0.42
trees	-0.06	0.23	-0.14	-0.27	0.14	0.24	0.55	0.77	0.66
graph	-0.06	0.34	-0.15	-0.30	0.20	0.31	0.69	0.98	0.85
minors	-0.04	0.25	-0.10	-0.21	0.15	0.22	0.50	0.71	0.62

r (human.user) = .94

r (human.minors) = -.83

Gambar 2.2: Matriks Pada Gambar (2.1) Setelah Diproses Menggunakan SVD Dengan Pemotongan Menyisakan 2 Dimensi [4]

Dibandingkan dengan gambar 2.1, korelasi antara kata *human* dan *user* naik dari -0.38 menjadi 0.94. Artinya, pada *passage* yang ada, kata *human* dan *user* sebetulnya memiliki korelasi yang kuat, namun tidak tampak pada *term-document matrix* sebelum dilakukan SVD. Demikian juga dengan pasangan kata *human* dan

minors yang turun dari -0.29 menjadi -0.93 yang menandakan bahwa korelasi antara kedua kata tersebut sebetulnya sangat tidak signifikan.

Proses ini dimodifikasi oleh Departemen Teknik Elektro Universitas Indonesia, dengan memperkecil ruang vektor tiap dokumen, menjadi per 10 kata, bukan per dokumen. Perbandingan kemiripan dua dokumen tidak dilakukan menggunakan *spearman's rho,* namun menggunakan matriks diagonal hasil dekomposisi, yang kemudian dibentuk menjadi vektor menggunakan *Frobenius Norm*, yakni salah satu bentuk normalisasi matriks sesuai dengan persamaan berikut:

$$\|A\|_F \equiv \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2} \tag{2.1}$$

$$\|A\|_F \equiv \sqrt{Tr(AA^H)} \tag{2.2}$$

Kemudian membandingkan masing-masing matriks *frobenius norm* untuk menemukan korelasi antara kedua dokumen.

2.3. *Learning Vector Quantization*

Learning Vector Quantization (selanjutnya disebut LVQ) adalah algoritma pelatihan yang memodifikasi sedikit bagian dari algoritma *Self Organizing Map* (SOM), dimana LVQ bersifat *supervised,* yang berarti algoritma ini didesain untuk belajar dengan dibantu oleh target/tujuan sebagai *supervisor* untuk melakukan pengelompokan. Algoritma ini digunakan untuk untuk membuat representasi terpisah dari masing-masing dimensi output. Hal ini berarti setiap input yang diharapkan masuk ke kelompok yang sama harus memiliki *common features* agar bisa dikelompokkan sesuai dengan *common features* yang dimiliki. Tujuan dibuatnya algoritma LVQ adalah untuk meningkatkan performa algoritma SOM (*recognition rate*) dengan bantuan target untuk mensupervisi proses, namun dengan kecepatan yang tetap tinggi, seperti ciri utama SOM. [6]

Sistem LVQ juga berbasis pada *competitive learning,* dimana hanya terdapat satu pemenang yang "diaktivasi" setiap waktunya, yang disebut sebagai *winning neuron.* Perbedaan antara LVQ dan SOM adalah sistem pengecekan

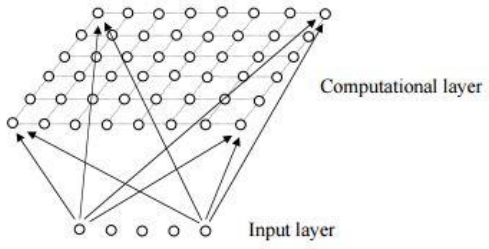
apakah *winning neuron* memang sesuai dengan target yang diharapkan. Sesuai tidaknya *winning neuron* dengan target yang diharapkan menentukan *update* bobot yang dialami oleh bobot-bobot yang terkait dengan *winning neuron* tersebut.

Penentuan neuron pemenang dilakukan dengan mencari *euclidean distance* antara input dengan masing-masing neuron. Neuron yang memiliki jarak terkecil pada input tersebut akan dianggap sebagai *winning neuron*.

Sistem LVQ diadaptasi dari hasil studi biologi saraf yang mengindikasikan bahwa sensor-sensor input yang berbeda jenis akan dipetakan menuju kelompok-kelompok sesuai area masing-masing pada *cerebral cortex.* Peta ini disebut sebagai *topographic map,* serta mengadaptasi *Kohonen network.* LVQ memiliki komponen inisialisasi, kompetisi, kooperasi, dan adaptasi.

2.3.1. *Kohonen Network*

Kohonen Network adalah salah satu jenis SOM, dimana struktur ini memiliki struktur *feed forward* dengan lapisan komputasi tunggal yang diatur dalam baris dan kolom. Setiap neuron komputasi terknoneksi secara penuh dengan neuron sumber di lapisan input.



Gambar 2.3: *Kohonen Network*

2.3.2. *Variabel-Variabel yang Mempengaruhi LVQ*

Berikut in adalah parameter-parameter yang mempengaruhi keluaran LVQ:

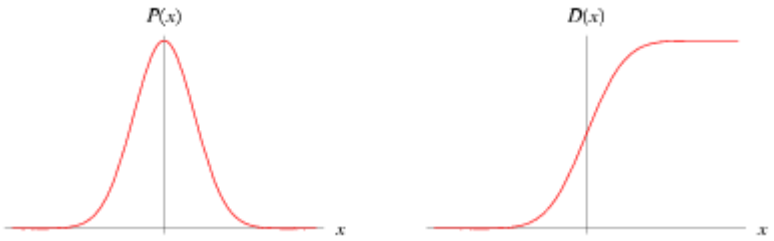
- input x dengan dimensi k dan jumlah data i (x_{i1} hingga x_{ik}, 1 *epoch* terdiri dari i input).
- target y dengan jumlah data i (y_i), dimana setiap input x_i memiliki pasangan yang bersesuaian yakni y_i.

- sejumlah j neuron output
- bobot w yang menghubungkan masing-masing neuron output dengan input. Bobot ini memiliki dua dimensi (w_{jk})
- *Euclidean distance* (D_j) antara masing-masing output dengan input tertentu.
- Laju pemahaman α
- *recognition rate* sebagai parameter keberhasilan *learning process*

2.4. Distribusi Normal

Distribusi normal adalah salah satu jenis distribusi statistik, dimana sebuah kelompok data dengan rata-rata μ, variansi σ^2, dan *variate* X akan memiliki *probability density function* (PDF) P(x) sebagai berikut: [7]

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2} \quad (2.3)$$



Gambar 2.4: *Probability Density Function* (Kiri) dan *Cummulative Distribution Function* (Kanan) Dari Distribusi Normal

Pada domain $x \in (-\infty, \infty)$. Dengan banyak properti yang spesial yang dimiliki oleh distribui normal, aplikasi dari distribusi normal menjadi sangat luas, sehingga banyak jenis distribusi lain bisa didekati dengan menggunakan distribusi normal, dari distribusi binomial hingga distribusi *poisson*.

Distribusi normal standar adalah distribusi normal dengan rata-rata 0 dan standar deviasi 1. Setiap distribusi normal dapat dikonversi menjadi distribusi normal standar dengan mengganti nilai X menjadi Z (atau sering disebut *z-score*), dimana

$$Z = \frac{(x-\mu)}{\sigma} \quad (2.4)$$

Sehingga dz = dx/σ, yang menghasilkan

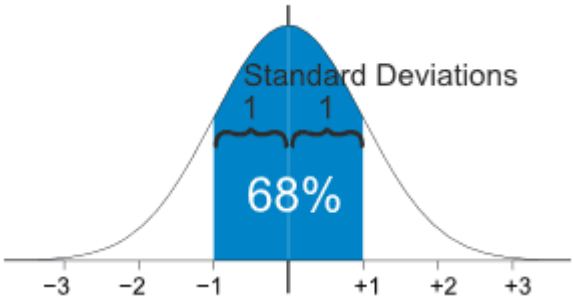
$$P(x)dx = \frac{1}{\sqrt{2\pi}} e^{-(z)^2/2} dz. \quad (2.5)$$

Untuk mengetahui berapa bagian dari data yang merentang dari interval [0,z], maka digunakanlah fungsi distribusi normal, yakni:

$$\Phi(z) = \frac{1}{\sqrt{2\pi}} \int_0^z e^{\frac{-(x)^2}{2}} dx = \frac{1}{2} \text{erf}(\frac{z}{2}). \quad (2.6)$$

Normal distribusi juga dikenal dengan aturan 68-95-99.7%. Aturan ini menyatakan bahwa setiap distribusi normal akan memenuhi properti di bawah ini yang juga disebut dengan *empirical rule* (aturan empiris), yakni: [8]

- 68% dari seluruh observasi akan jatuh di antara 1 standar deviasi dari rata-rata (di antara μ – σ hingga μ + σ)
- 95% dari seluruh observasi akan jatuh di antara 2 standar deviasi dari rata-rata (di antara μ – 2σ hingga μ + 2σ)
- 99.7% dari seluruh observasi akan jatuh di antara 3 standar deviasi dari rata-rata (di antara μ – 3σ hingga μ + 3σ)



Gambar 2.5: Gambaran Aturan Empiris Untuk 1 Standar Deviasi [9]

Properti di atas dapat digunakan untuk melakukan pengelompokan data, misalnya pengelompokan berdasarkan data-data yang berada di dalam dan di luar standar deviasi, atau menggunakan batas bawah atau batas atas standar deviasi menjadi batas dari kedua kelompok yang ada.

2.5. *Precision, Recall,* dan *F-Measure*

Precision dan *Recall* adalah dua perhitungan yang biasanya digunakan dalam mengukur performa *natural language processing* yang digunakan untuk melakukan pembelajaran terkait dokumen. *Precision* mengukur banyaknya dokumen yang ditandai benar oleh *natural language processing*, betul-betul benar (seberapa tepat prediksi dari *natural language processing*), sedangkan *Recall* mengukur banyaknya dokumen yang ditandai benar oleh *human reader* yang berhasil ditandai benar oleh *natural language processing* (sebarapa mampu *natural language processing* mengembalikan dokumen yang seharusnya dikembalikan). Secara umum, *Precision* dan *Recall* dapat ditunjukkan dengan tabel berikut:

	Prediksi Betul	**Prediksi Salah**
Kenyataan Betul	A	B
Kenyataan Salah	C	D

Tabel 2.1: Tabel Konsep Perhitungan *Precision* dan *Recall*

Rumus *Precision* dan *Recall* sesuai dengan tabel di atas adalah:

$$Precision = \frac{A}{(A+C)}, Recall = \frac{A}{(A+B)} \tag{2.7}$$

Precision digunakan untuk mengetahui seberapa banyak *false positives* yang terjadi dalam pemrosesan oleh *natural language processing*, ditandai dengan adanya variabel C pada perhitungan di atas (Prediksi betul pada kenyataan yang salah memberikan *false positive*), sedangkan *Recall* digunakan untuk mengetahui seberapa banyak *false negatives* yang terjadi, ditandai dengan adanya variabel B pada perhitungan di atas (Prediksi salah pada kenyataan yang betul memberikan *false negative*).

Perhitungan hanya menggunakan *Precision* saja atau *Recall* saja tidak mampu memberikan pandangan yang seimbang terhadap *false positives* dan *false negatives*. Untuk mengakomodasi pandangan yang seimbang dan sesuai kebutuhan terhadap *false positives* dan *false negatives*, digunakanlah properti *F-Measure*. *F-Measure* adalah rata-rata harmonik berbobot β dari *Precision* dan *Recall*, dengan rumus:

$$F_\beta = (1+\beta^2)\frac{percision.recall}{\beta^2.precision+recall} \tag{2.8}$$

Penggunaan nilai 1 pada bobot β menghasilkan *F-Measure* tradisional atau *F-Measure* seimbang, dimana nilai *Precision* dan *Recall* dianggap sama pentingnya dalam penilaian performa *natural language processing.* Semakin besar nilai bobot β berarti *F-Measure* memberikan penekanan yang lebih tinggi pada *Recall*, sedangkan nilai bobot β yang lebih rendah berarti *F-Measure* memberikan penekanan yang lebih tinggi pada *Precision.*

Penggunaan nilai *Precision*, *Recall*, dan *F-Measure* berguna untuk menentukan apakah algoritma *natural language processing* yang digunakan berhasil memberikan keluaran yang sesuai dengan keinginan. Nilai yang tinggi pada *Precision* berarti algoritma mampu menekan nilai *false positives*, sehingga nilai positif yang dikembalikan oleh algoritma betul-betul positif. Nilai yang tinggi pada *Recall* berarti algoritma mampu menekan nilai *false negatives*.

2.6. Simple-O

Simple-O adalah sistem peniilaian esai otomatis yang sedang dikembangkan oleh Departemen Teknik Elektro Universitas Indonesia. Metode yang digunakan oleh Simple-O adalah dengan mengambil setiap kata dalam esai dan mencocokkannya dengan kata-kata dari kunci jawaban. Proses perhitungan dilakukan dengan algoritma Latent Semantic Analysis dan Singular Value Decomposition, dan ditulis dengan bahasa pemrograman PHP karena diharapkan sistem ini dapat ditempatkan dalam *cloud*.

Struktur basis data dari sistem ini adalah tabel dengan ID yang menandakan soal yang ingin diuji, bobot lokal dan global untuk menentukan bobot niladalm perhitngan LSA, dengan keluaran akhir adalah nilai dari jawaban yang dievaluasi. Terdapat juga tabeluntuk menyimpan kumpulan persamaan kata, untuk menyelesaikan kasus dari kata-kata yang berbeda namun bermakna sama.

Proses pengecekan adalah dengan menggunakan ID sebagai referensi untuk mengambil kunci jawaban, yang terdiri dari jawaban asli, kata kunci, dan bobot yang telah ditentukan oleh *human rater*. Kata-kata dari teks kemudian disimpan bentuk *array*. *Array* yang telah melalui proses sinonim akan dijadikan input untuk fungsi yang memetakan *array* menjadi matriks *term-document* yang akan diproses

menggunakan SVD. Panjang vektor dari matriks akhir perhitungan LSA akan dijadikan pembanding untuk menilai jawaban uji. Perbandingan dilakukan dengan membagi panjang vektor matriks jawaban dengan panjang vektor matriks kunci, kemudian diberi skala 100.

2.7. PHP: Hypertext Preprocessor (PHP)

PHP adalah bahasa pemrograman yang digunakan untuk membuat konten dinamis yang berinteraksi dengan basis data. Sebagai bahasa pemrograman yang bersifat *server-side,* PHP, tidak seperti bahasa-bahasa di atas, tidak diterjemahkan oleh *web browser*, namun diterjemahkan oleh server tempat program PHP tersebut berada, sehingga program yang ditulis dengan bahasa PHP hanya bisa berjalan di dalam mesin yang telah dikonfigurasi untuk bisa menginterpretasikan program PHP. Dokumen yang berisi program PHP memiliki ekstensi .php.

Untuk menandai bahwa suatu program menggunakan PHP, digunakan *tag* pembuka `<?php` dan penutup `?>`. Kode PHP dapat digunakan secara bersamaan dengan kode HTML, baik dengan memiliki suatu dokumen yang berisi bahasa PHP serta *markup* HTML diluarnya, maupun menggunakan PHP untuk mengeluarkan *markup* HTML (baik dengan perintah `echo`, `print`, `print_r`, dan lain-lain.. Pemrograman PHP juga dapat dilakukan dengan berorientasi objek (*Object Oriented Programming*).

PHP dapat diimplementasikan secara *as-is* ataupun dengan *framework*, yakni sekumpulan *library* yang berbasis PHP, namun mengubah sistem dan paradigma pemrograman PHP yang ada, dengan tujuan untuk mempermudah pemrograman secara keseluruhan dan optimasi keluaran program. [10]

Program pada riset ini dibangun menggunakan bahasa PHP, dengan mempertimbangkan kelebihan berikut, yakni: [11]

- Simpel dan mudah dipelajari
- Memiliki banyak dokumentasi serta bantuan dari *blog*, *forum*, dan lain-lain
- Dapat dijalankan pada banyak OS dan tidak membutuhkan IDE spesifik dalam pengembngannya

- Gratis
- Dapat diintegrasikan dengan banyak sistem lain
- Banyak *framework* yang membantu pengembangan aplikasi berbsais PHP
- Masalah mudah dideteksi dan diperbaiki
- Mendukung skalabilitas
- Mendukung paradigma pemrograman berorientasi objek
- Kecepatan eksekusi yang tinggi

2.8. MATLAB®

MATLAB® adalah bahasa tingkat tinggi dan lingkungan interaktif yang digunakan oleh jutaan insinyur dan ilmuwan yang memugkinkan penggunanya mengeksplor dan memvisualisasikan ide dan berkolaborasi dengan ilmuwan multidisiplin. MATLAB® memiliki spesialisasi dalam komputasi matematis, statistik, dan lain-lain yang berhubungan dengan sains, sehingga banyak algoritma matematis dan statistika yang dikembangkan menggunakan MATLAB®. [12]

MATLAB® adalah bahasa tingkat tinggi dan lingkungan interaktif yang digunakan oleh jutaan insinyur dan ilmuwan. MATLAB® memugkinkan penggunanya mengeksplor dan memvisualisasikan ide dan berkolaborasi dengan ilmuwan multidisiplin. MATLAB® memiliki spesialisasi dalam komputasi matematis, statistik, dan lain-lain yang berhubungan dengan sains, sehingga banyak library yang sangat berguna untuk membangun program matematis dan statistika yang memudahkan pengguna. Misalnya, pengguna dapat menghitung standar deviasi dari sekumpulan data X hanya dengan memanggil `stdev(X)`. Sedangkan dengan bahasa pemrograman lain, pengguna perlu mengimplementasikan sendiri logika dari standar deviasi untuk dapat menghitung standar deviasi.

Program yang dibangun menggunakan MATLAB® menggunakan ekstensi .m. Iterasi dan logika yang digunakan dalam bahasa ini mirip dengan program lainnya (while loop, for loop, if, dan sebagainya). Perbedaannya, program berbasis MATLAB® tidak dibatasi menggunakan titik koma, dan fungsi atau iterasi tidak dibatasi dengan kurung kurawal.

2.9. Web Server

Web Server adalah sebuah *server* yang bertugas untuk menerima *request* dari klien dan memberikan respon kepada klien tersebut. *Request* dari klien akan berisi URI. URI tersebut akan diterjemahkan oleh *web server* menjadi nama file (untuk *request* statis), atau program/fungsi dalam sebuah program (untuk *request* dinamik). Hasil dari akses tersebut, baik berupa *file* maupun keluaran fungsi, kepada pihak yang me-*request*.

Salah satu bagian dari *web server* yang akan dibahas di laporan ini adalah *HTTP Server*. *HTTP Server* adah *web server* yang khusus bekerja untuk menerima dan memproses *request* dengan protokol *HTTP*.

Apache Web Server adalah *web server* yang diproduksi oleh *Apache*, sebuah lembaga perangkat lunak berbasis *open*-source. *Apache Web Server* adalah *web server* yang paling populer sampai saat ini. Selain *HTTP Server*, *Apache Web Server* juga menyediakan banyak servis lain, yakni *Simple Mail Transfer Protocol* (SMTP – TCP Port 25), *Domain Name Service* (DNS – TCP Port 53), *File Transfer Protocol* (FTP – TCP Port 21).

Apache menggunakan protokol TCP/IP untuk berkomunikasi dengan klien. Hal ini berarti klien dan server berkomunikasi menggunakan IP Address dan *Port number*. Apache *Web Server* memproses seluruh *request* yang masuk ke server berdasarkan file httpd.conf. [13]

2.10. MySQL

MySQL adalah sebuah Relational Database Management System (RDBMS) yang bersifat open source. MySQL dapat digunakan untuk membuat database sebagai tempat penyimpanan dan pemrosesan seluruh data yang akan digunakan dalam pekerjaan pada kerja praktek ini. MySQL dapat berinteraksi dengan PHP, yang berarti seluruh operasi pada MySQL dapat dilakukan melalui PHP, karena PHP dapat mengirim query pada MySQL. Manajemen database MySQL secara graphical user interface dapat dilakukan melalui phpmyadmin. Modul MySQL dan phpmyadmin sudah tersedia secara langsung ketika pengguna

membangun web server stack menggunakan LAMP (untuk Linux) atau XAMPP (untuk Windows)

BAB 3
PROGRAM DETEKSI PLAGIARISME DAN PENGELOMPOKAN DATA HASIL KELUARAN PROGRAM

3.1. Spesifikasi Perangkat

Perancangan sistem dilakukan dengan menggunakan beberapa perangkat keras dan lunak, yakni:

3.1.1. *Perangkat Keras*

Perangkat keras yang digunakan untuk mengimplementasikan sistem ini adalah laptop ASUS A43SD dengan spesifikasi sebagai berikut:

- *Processor:* Intel(R) Core(TM) i5-2450M CPU @ 2.50 GHz
- *Installed Memory* (RAM): 4,00 GB
- Kapasitas *Hard Disk*: 550 GB

3.1.2. *Perangkat Lunak*

Daftar perangkat lunak yang digunakan untuk mengimplementasikan sistem ini adalah:

- Sistem Operasi Windows 7 Ultimate 64 bit
- Sublime Text 2, yakni perangkat lunak *text editor* untuk menulis program berbasis PHP di sistem ini
- MATLAB®, yakni perangkat lunak untuk komputasi, baik matematis, statistik, dan lain-lain. Digunakan untuk menulis dan melakukan simulasi pembelajaran program *Learning Vector Quantization*
- Web Browser (Google Chrome), digunakan untuk menjalankan program berbasis PHP

3.2. Perancangan Sistem Deteksi Plagiarisme

Sistem deteksi plagiarisme yang dibangun adalah pengembangan dari sistem penilaian esai otomatis bernama Simple-O yang memanfaatkan algoritma yang sama, yaitu algoritma LSA.

Pertimbangan untuk menggunakan LSA didasari oleh kurangnya sistem *natural language processing* yang dimiliki oleh Bahasa Indonesia, sehingga cara terbaik yang dapat dilakukan adalah dengan memperhatikan hanya fitur-fitur leksikal dari deteksi plagiarisme, yang didukung oleh LSA. [14] LSA mampu memahami fitur leksikal dari sebuah dokumen dengan menggunakan konsep SVD, dimana SVD dapat mengungkap konsep dari dokumen, bukan kata-kata pada dokumen. *Term-document* matriks yang dibangun oleh LSA dapat digunakan untuk model probabilistik untuk melakukan observasi kemiripan dari dokumen. [15] Untuk itu, LSA sudah sering digunakan untuk membangun sistem berbasis pengetahuan di atas korpus paralel untuk menemukan kejadian plagiarisme. [16]

Dengan fitur-fitur di atas, maka sistem yang dibangun akan menguji orisinalitas karya tulis yang ditulis dalam Bahasa Indonesia, dengan referensi karya tulis yang ditulis dalam Bahasa Inggris.

Langkah awal yang perlu dilakukan dalam membangun sistem ini adalah dengan mengumpulkan sepuluh tulisan berbahasa Inggris dengan berbagai topik, yang kemudian disimpan dalam sebuah *database*. Setiap paragraf dari tulisan akan dikaitkan dengan kunci untuk mempermudah pemanggilannya. Tulisan bahasa Inggris itu kemudian ditranslasi menjadi bahasa Indonesia menggunakan translasi manual oleh manusia. Tulisan hasil translasi kemudian juga disimpan dalam *database*, yang juga dikaitkan dengan kunci. Kunci pada tulisan Bahasa Inggris tertentu akan sama dengan kunci pada tulisan Bahasa Indonesia yang merupakan hasil translasinya. Tulisan-tulisan tersebut akan dijadikan bahan pengujian.

3.3. Alur Sistem Pendeteksi Plagiarisme

Proses pendeteksi plagiarisme dimulai dengan membuang tanda baca serta *stopwords*, baik dari setiap paragraf dari karya tulis yang akan diuji maupun dari paragraf dari karya tulis referensi. *Stopwords* harus dibuang karena dengan natur dari *stopwords* yang tidak memiliki arti semantik yang penting [17], keberadaan *stopwords* yang sering pada setiap karya tulis akan mengacaukan algoritma LSA, membuat algoritma menjadi bias untuk menentukan konteks dari karya tulis yang ada.

Kemudian, program akan melakukan translasi pada karya tulis Bahasa Indonesia pada *database* menjadi Bahasa Inggris menggunakan *database* kamus yang akan melakukan translasi kata per kata tanpa memeprhatikan tata bahasa dan struktur, karena LSA tidak memperhatikan tata bahasa maupun struktur dari kalimat.

Setelah proses translasi dijalankan, sistem akan merujuk kepada paragraf dari karya tulis referensi, untuk kemudian mengambil setiap kata-kata yang unik dari paragraf tersebut (kata yang muncul dua kali tidak diambil kembali) untuk menjadi *keywords*. *Keywords* ini akan menjadi bagian *term* pada *term-document matrix* yang akan dibangun kemudian, baik bagi paragraf karya tulis referensi maupun bagi karya tulis yang diuji. Pada tahap ini, sistem telah memiliki daftar kata dari paragraf karya tulis uji dan referensi sebagai *document* dan kata-kata dari paragraf karya tulis referensi sebagai *term*.

Masing-masing kata-kata dari paragraf karya tulis uji dan referensi kemudian dibentuk menjadi sebuah *term-document matrix*. Untuk mengurangi beban sistem, definisi *document* dibuat menjadi 10 kata, bukan 1 paragraf. Kemudian, SVD akan dilakukan pada masing-masing *term-documen matrix*, baik milik paragraf karya tulis uji, maupun milik referensi. Dengan pengetahuan bahwa SVD akan mendekomposisi matriks sebagai berikut:

$$M = U\Sigma V^* \tag{3.1}$$

Maka sistem akan mengambil matriks diagonal Σ sebagai *fingerprint* dari masing-masing *term-document matrix*.

Kemudian, kedua matriks tersebut akan dibandingkan menggunakan dua cara, yakni:

- Membandingkan *Frobenius Norm* paragraf dari karya tulis uji dengan *Frobenius Norm* paragraf dari karya tulis referensi yang dibangun dari vektor pada matriks Σ milik masing-masing paragraf dengan skala 100. Semakin dekat nilai perbandingan dengan angka 100, maka kedua paragraf semakin mirip. Untuk keperluan pengolahan data, angka hasil perbandingan ini akan dikurangi

100. Hal ini berarti semakin dekat nilai perbandingan dengan angka 0, maka kedua paragraf semakin mirip [18].

- Menghitung sudut dari kedua vektor yang direpresentasikan dengan masing-masing matriks Σ. Semakin sempit sudut dari kedua vektor (semakin dekat nilai sudut dengan angka nol), semakin mirip pula kedua paragraf tersebut. Jika dimensi dari kedua vektor berbeda, maka:
  - Vektor dengan dimensi lebih banyak akan dipotong dimensi-dimensi yang lebihnya sampai jumlah dimensinya sama dengan vektor yang dimensinya lebih kecil. Langkah ini disebut dengan *slice*.
  - Vektor dengan dimensi lebih kecil akan mengalami penambahan dimensi dengan penambahan elemen *array* sampai jumlah dimensinya sama dengan vektor yang dimensinya lebih panjang. Elemen *array* yang ditambahkan bernilai 0. Langkah ini disebut dengan *pad.*

Hasil keluaran LSA dapat diobservasi melalui ketiga parameter di atas.

idptest	idpref	frobn	slice	pad
1001	1010	47.7119	34.4035	34.8333
1001	1011	95.3463	28.1482	28.1482
1001	1012	100	17.4361	22.3399
1001	1013	17.4078	19.3733	19.3733
1001	1014	95.3463	28.1482	28.1482
1001	1015	64.5497	25.3886	25.3886
1001	1016	21.4834	20.7583	20.7583
1001	1017	27.0501	31.1962	31.1962
1001	1018	24.6183	90	90
1002	1001	78.8811	22.1904	25.3824
1002	1002	100.816	9.28766	9.28766
1002	1003	54.2326	31.0747	31.0747
1002	1004	81.2404	26.0351	26.0351
1002	1005	49.4872	39.0666	39.0666

Gambar 3.1: Contoh Hasil Keluaran Sistem Deteksi Plagiarisme

```
FOR(counter = id_uji_awal TO id_uji_akhir)
{ //untuk setiap paragraf dari karya tulis uji yang dipilih
       INPUT paragraf_uji; //memasukkan paragraf uji
       CLEAN paragraf_uji; //menghilangkan tanda baca dari paragraf uji
       EXCLUDE STOPWORDS paragraf_uji; //menghilangkan stopwords dari
paragraf uji
       CREATE ARRAY array_uji; //memasukkan kata-kata pada paragraf uji
dalam array
       FOREACH(array_uji AS key)
       { //untuk setiap kata-kata pada paragraf uji
              TRANSLATE key; //lakukan translasi ke Bahasa Inggris
              PUSH key TO array_uji; // masukkan kembali hasil
translasi ke array
       }
       FOR(counter = id_ref_awal TO id_ref_akhir)
       { //untuk setiap paragraf dari karya tulis referensi yang
dipilih
              INPUT paragraf_ref; //memasukkan paragraf referensi
              CLEAN paragraf_ref; //menghilangkan tanda baca dari
paragraf referensi
              EXCLUDE STOPWORDS paragraf_ref; //menghilangkan stopwords
dari paragraf referensi
              CREATE ARRAY array_ref; //memasukkan kata-kata pada
paragraf referensi dalam array
              keywords = array_ref; //keywords yang akan digunakan
berasal dari array kata-kata paragraf referensi
              CREATE matriks_uji; //membuat term-document matrix pada
paragraf uji
              SVD matriks_uji; //melakukan SVD pada term-document
matrix paragraf uji yang telah dibuat
              CREATE matriks_ref; //membuat term-document matrix pada
paragraf referensi
              SVD matriks_ref; //melakukan SVD pada term-document
matrix paragraf referensi yang telah dibuat
              fnorm = (FROB NORM matriks_s_uji)/(FROB NORM
matriks_s_ref)*100; //menghitung perbandingan frobenius norm kedua
matriks
              slice_uji = SLICE matriks_s_uji; //memotong dimensi
matriks uji agar sama dengan referensi
              slice_ref = SLICE matriks_s_ref; //memotong dimensi
matriks referensi agar sama dengan uji
              slice = ARC COS (slice_uji*slice_ref/FROB NORM slice_uji
* FROB NORM slice_ref); //menghitung sudut dari kedua matriks
              pad_uji = PAD matriks_s_uji; //menambahkan dimensi
matriks uji agar sama dengan referensi
              pad_ref = PAD matriks_s_ref; //menambahkan dimensi
matriks referensi agar sama dengan uji
              pad = ARC COS (pad_uji*pad_ref/FROB NORM pad_uji * FROB
NORM pad_ref); //menghitung sudut dari kedua matriks
       }
}
```

Gambar 3.2: *Pseudocode* Sistem Pendeteksi Plagiarisme Berbasis LSA

3.4. Pengelompokan Hasil Keluaran Sistem Deteksi Plagiarisme

Hasil yang telah dikeluarkan oleh sistem pendeteksi plagiarisme berupa angka, sehingga angka-angka tersebut perlu dikelompokkan sebagai angka-angka yang menyatakan indikasi plagiarisme dan angka-angka yang tidak menyatakan

indikasi plagiarisme. Dalam skripsi ini, penulis akan membahas algoritma-algoritma yang dapat digunakan untuk melakukan pengelompokan tersebut.

3.4.1. *Penentuan Indikasi Plagiarisme Berdasarkan Distribusi Normal*

Metode distribusi normal akan mengelompokkan data berdasarkan posisinya terhadap nilai standar deviasi tertentu pada rentang distribusi normal (*z-score*), dengan batas yang ditentukan oleh nilai standar deviasi tertentu yang akan divariasikan. Sesuai sifat distribusi normal tradisional dengan rumus:

$$z = \frac{x-\mu}{\sigma} \tag{3.2}$$

$$\mu + z\sigma = x \tag{3.3}$$

Maka semakin kecil nilai *z-score* dari suatu data, berarti semakin kecil juga nilai dari data tersebut, sesuai dengan representasi grafik distribusi normal yang menyatakan semakin kecil *z-score*, posisi data akan semakin ke kiri.

Variasi akan dilakukan pada nilai *z*. Dengan menentukan nilai *z*, maka akan terbentuk sebuah nilai batas yang akan memisahkan antara nilai yang tidak terindikasi plagiat ($x \geq \mu + z\sigma$), dengan nilai yang terindikasi plagiat ($x < \mu + z\sigma$). Selain itu, sebagai percobaan yang heuristik, penulis menambahkan koefisien k untuk mengalikan nilai batas yang ada, sehingga nilai batas menjadi

$$x = k(\mu + z\sigma). \tag{3.4}$$

Nilai z dan k akan dimodifikasi untuk menemukan hasil terbaik dari metode distribusi normal.

Proses yang dikerjakan oleh metode ini yakni dengan mengambil suatu karya tulis uji, mengambil setiap paragraf yang berada di dalam paragraf tersebut, kemudian untuk masing-masing paragraf uji, mengambil setiap hasil perbandingan antara paragraf uji tersebut dengan setiap paragraf referensi yang pernah dibandingkan dengan algoritma LSA. Setelah itu masing-masing nilai (*fnorm, slice*, dan *pad*) akan dibandingkan dengan batas nilai tertentu yang ditentukan dengan metode distribusi normal. Jika nilai-nilai tersebut berada dibawah batas nilai yang ada, maka nilai tersebut terindikasi plagiat.

Kumpulan data yang digunakan untuk membentuk distribusi normal adalah nilai-nilai dari perbandingan satu paragraf uji dengan seluruh paragraf referensi yang pernah dibandingkan.

```
FOR(counter = id_uji_awal TO id_uji_akhir){ //untuk setiap paragraf dari karya
tulis uji yang dipilih
	GET id_ref, fnorm, slice, pad; //mengambil nilai-nilai dari paragraf
referensi yang telah di-generate nilainya oleh LSA
	fnorm = 100 - fnorm; //mendefinisikan ulang nilai fnorm agar definisi
plagiat menjadi nilai nol
	mean_fnorm = (SUM fnorm)/(COUNT fnorm); //menentukan nilai rata-rata
frobenius norm dari kumpulan data fnorm yang ada
	std_fnorm = sd(fnorm); //menentukan nilai standar deviasi frobenius
norm dari kumpulan data fnorm yang ada
	FOREACH(fnorm AS value){ //untuk setiap nilai fnorm
		IF(value < k*(mean_fnorm - z*std_fnorm)){ //jika nilai fnorm
lebih kecil dari batas tertentu
			status_fnorm = "plagiat"; //fnorm dari perbandingan
kedua paragraf terkait menandakan indikasi plagiat
		}
		ELSE{
			status_fnorm = "tidak"; //fnorm dari perbandingan kedua
paragraf terkait tidak menandakan indikasi plagiat
		}
	}
	mean_slice = (SUM slice)/(COUNT slice); //menentukan nilai rata-rata
slice dari kumpulan data slice yang ada
	std_slice = sd(slice); //menentukan nilai standar deviasi slice dari
kumpulan data slice yang ada
	FOREACH(slice AS value){ //untuk setiap nilai slice
		IF(value < k*(mean_slice - z*std_slice)){ //jika nilai slice
lebih kecil dari batas tertentu
			status_slice = "plagiat"; //slice dari perbandingan
kedua paragraf terkait menandakan indikasi plagiat
		}
		ELSE{
			status_slice = "tidak";//slice dari perbandingan kedua
paragraf terkait tidak menandakan indikasi plagiat
		}
	}
	mean_pad = (SUM pad)/(COUNT pad); //menentukan nilai rata-rata pad dari
kumpulan data pad yang ada
	std_pad = sd(pad); //menentukan nilai standar deviasi pad dari kumpulan
data pad yang ada
	FOREACH(pad AS VALUE){ //untuk setiap nilai pad
		IF(value < k*(mean_pad - z*std_pad)){ //jika nilai pad lebih
kecil dari batas tertentu
			status_pad = "plagiat"; //pad dari perbandingan kedua
paragraf terkait menandakan indikasi plagiat
		}
		ELSE{
			status_pad = "tidak"; //pad dari perbandingan kedua
paragraf terkait tidak menandakan indikasi plagiat
		}
	}
}
```

Gambar 3.3: *Pseudocode* Penentuan Indikasi Plagiarisme Berdasarkan Distribusi Normal

Hasil pengelompokan data kemudian dibandingkan dengan hasil pembacaan manual oleh penulis, apakah hasil deteksi oleh metode distribusi normal sesuai dengan hasil pembacaan manual oleh penulis, menghasilkan nilai *Precision*, *Recall*, dan *F-Measure*.

3.4.2. *Penentuan Indikasi Plagiarisme Berdasarkan Algoritma Learning Vector Quantization*

Untuk menentukan indikasi plagiarisme menggunakan algoritma *Learning Vector Quantization*, algoritma LVQ akan melakukan *training*. *Training* dilakukan untuk mendapatkan nilai bobot *neural network* yang sesuai, agar program dapat mengenali setiap masukan dan mengelompokkannya dengan benar. *Training* yang dilakukan sebanyak satu kali, dimana data untuk *training* diambil dari seluruh nilai perbandingan yang telah dilakukan. Hal ini berbeda dengan metode distribusi normal dimana metode distribusi normal dijalankan untuk setiap satu paragraf uji.

Dalam fase *training*, algoritma *Learning Vector Quantization* akan melakukan inisialisasi bobot dengan mengambil nilai input, dengan asumsi bahwa bobot yang berkorespondensi dengan input akan memiliki jarak nol, sehingga bobot yang ada sudah mendekati bobot yang dibutuhkan. Namun karena inisialisasi bobot hanya menggunakan sedikit bagian dari input, bobot tetap harus dilatih untuk dapat menyesuaikan diri dengan input lainnya. Penyesuaian bobot dilakukan dengan mencari *euclidean distance* terdekat antara input dengan masing-masing neuron, yang rumusnya dijelaskan pada rumus (3.4). Indeks neuronang memiliki *euclidean distance* terdekat akan dicek terhadap target, apakah neuron tersebut yang seharusnya menang. Jika memang neuron tersebut sesuai, maka bobot akan dimodifikasi unuk mengurangi jarak antara input dengan neuron tersebut. Jika neuron tersebut ternyata taidak sesuai, maka bobot akan modifikasi untuk menambah jarak antara input dengan neuron tersebut. [6]. Modifikasi bobot dilakukan menggunakan variabel *alpha*, yang akan semakin kecil seiring berjalannya algoritma agar modifikasi bobot menjadi lebih presisi. Modifikasi bobot akan terus dilakukan hingga kondisi henti dicapai.

Dalam fase *testing*, algoritma LVQ menggunakan bobot yang didapat pada fase *training*. Proses berjalannya algoritma sama seperti fase *training*, namun fase *testing* berhenti sampai penentuan neuron yang menang (neuron yang memiliki *euclidean distance* terpendek dengan input tertentu). Keluaran dari fase *testing* ini adalah pasangan input dengan masing-masing indeks neuron pemenangnya.

Dengan parameter-parameter yang telah dijelaskan pada bab 2, proses algoritma LVQ adalah sebagai berikut:

Training:

- 0. Inisialisasi bobot w
- 1. Cek kondisi henti. Jika kondisi henti gagal, lakukan langkah 2 – 7. Kondisi henti yang dimaksud adalah jika variabel yang menjadi kriteria pemberhentian pelatihan (*epoch* atau *alpha*) sudah sesuai dengan angka yang telah ditentukan.

  Langkah 2 hingga 7, kemudian 1 dilakukan per *epoch*
- 2. Untuk setiap vektor masukan x_i berdimensi k (x_{i1} hingga x_{ik}), lakukan langkah 3 hingga 7.

  Langkah 3 hingga 6 dilakukan tiap input x_i.
- 3. Untuk setiap neuron output dengan dimensi j, hitung *euclidean distance* antara bobot yag terkorespondensi dengan output tertentu dengan x_i

$$D_j = \sqrt{\sum_{k=1}^{m}(w_{kj} - x_{ik})^2} \quad (3.5)$$

- 4. Tentukan indeks j hingga Dj minimum
- 5. Cek target x_i yang seharusnya (y_i).
- 6. Untuk setiap bobot yang yang menghubungkan input tersebut dengan neuron pemenang, mengupdate bobot dengan
  - $W_{ji}(new) = W_{ji}(old) + \alpha(x_i - W_{ji}(old))$ jika indeks j **sesuai** dengan target (indeks j yang terkait dengan x_i sama dengan y_i)
  - $W_{ji}(new) = W_{ji}(old) - \alpha(x_i - W_{ji}(old))$ jika indeks j **tidak sesuai** dengan target (indeks j yang terkait dengan x_i sama dengan y_i)
- 7. Memodifikasi laju pemahaman dengan $\alpha(t+1) = n\alpha(t)$

  Testing:
- 8. Mengambil bobot yang telah didapat dari proses testing
- 9. Untuk setiap vektor masukan x_i (x_{i1} hingga x_{ik}), lakukan langkah 10 hingga 12.
- 10. Untuk setiap neuron output dengan dimensi j, hitung *euclidean distance* antara bobot yag terkorespondensi dengan output tertentu dengan x_i

$$D_j = \sqrt{\sum_{k=1}^{m}(w_{kj} - x_{ik})^2} \quad (3.6)$$

- 11. Tentukan indeks j hingga Dj minimum

```
GET input,alpha,n,dim_out,kondisi_henti,target; //mengambil seluruh parameter
yang dibutuhkan
dim_in = DIMENSI input; //mengambil dimensi dari input
i = COUNT input; //mengambil jumlah input yang ada
FOR(counter = 1 TO dim_out){ //untuk setiap dimensi output
        FOR(counter2 = 1 TO dim_in){ //untuk setiap dimensi input
                w(counter,counter2) = input(counter,counter2); //inisialisasi
beban dengan input sejumlah dimensi output
        }
}
WHILE(kondisi_henti IS FALSE){ //jika kondisi henti belum terpenuhi
        FOR(counter = 1 TO dim_in){ //untuk setiap dimensi input
                FOR(counter2 = 1 to DIM_out){ //untuk setiap dimensi output
                        dist = EUCLIDEAN DISTANCE BETWEEN w AND in //cari
euclidean distance antara beban dengan input
                }
                j_win = INDEX OF MIN(dist); //tentukan indeks dari jarak
terpendek
                IF(j_win == target){ //jika indeks jarak terpendek sesuai target
                        FOR(counter2 = 1 to DIM_OUT){
                                w(counter2,counter) = w(counter2,counter) +
(alpha*(input(counter) - w(counter2,counter))); //DEKATKAN kelas output dengan
input
                        }
                }
                ELSE{ //jika indeks jarak terpendek TIDAK sesuai target
                        FOR(counter2 = 1 to DIM_OUT){
                                w(counter2,counter) = w(counter2,counter) -
(alpha*(input(counter) - w(counter2,counter))); //JAUHKAN kelas output dengan
input
                        }
                }
        }
        alpha = n*alpha; //koreksi nilai alpha
        epoch = epoch + 1; //counter epoch
}
```

Gambar 3.4: Fase *Training* Pada Algoritma *Learning Vector Quantization*

```
GET input,w; //mengambil seluruh parameter yang dibutuhkan
dim_in = DIMENSI input; //mengambil dimensi dari input
i = COUNT input; //mengambil jumlah input yang ada
FOR(counter = 1 TO dim_in){ //untuk setiap dimensi input
        FOR(counter2 = 1 to DIM_out){ //untuk setiap dimensi output
                dist = EUCLIDEAN DISTANCE BETWEEN w AND in //cari euclidean
distance antara beban dengan input
        }
        j_win = INDEX OF MIN(dist); //tentukan indeks dari jarak terpendek
}
```

Gambar 3.5: *Pseudocode* Fase *Testing* Algoritma *Learning Vector Quantization*

Dalam penggunaan LVQ untuk melakukan deteksi indikasi plagiarisme, dimensi neuron ditentukan, yaitu 2, yakni “plagiat” dan “tidak plagiat”. Keluaran dari fase *testing* pada LVQ, yakni pasangan input dengan masing-masing indeks neuron pemenangnya, dapat menentukan indikasi terhadap nilai input tersebut

(dimana nilai input adalah hasil perbandingan antara paragraf uji dan paragraf referensi tertentu) , apakah nilai tersebut menandakan indikasi plagiat atau tidak.

Hasil indeks pemenang kemudian dibandingkan dengan hasil pembacaan manual oleh penulis, apakah hasil deteksi oleh algoritma LVQ sesuai dengan hasil pembacaan manual oleh penulis, menghasilkan nilai *Precision*, *Recall*, dan *F-Measure*.

3.5. Kalibrasi Metode Pengelompokan Hasil Keluaran Sistem Deteksi Plagiarisme

Untuk mendapatkan penentuan indikasi plagiarisme yang paling sesuai, maka penulis melakukan kalibrasi pada metode distribusi normal) melalui parameter-parameter yang dapat diubah pada masing-masing metode/algoritma. Proses kalibrasi ini dilakukan dengan metode heuristik, yakni dengan mencoba berbagai metode yang memungkinkan dan melihat keluarannya. Keluaran dari masing-masing percobaan akan menjadi dasar bagi penulis untuk mencoba metode lainnya.

3.5.1. *Kalibrasi Pada Metode Distribusi Normal*

Untuk mengetahui potensi akurasi metode distribusi normal dalam melakukan deteksi indikasi plagiarisme, maka penulis mencoba untuk melakukan perubahan-perubahan pada parameter-parameter yang bisa diubah. Berdasarkan rumus (3.4), yakni $x = k(\mu + z\sigma)$, maka parameter yang dapat diubah oleh penulis adalah nilai *z-score* dan nilai koefisien k. Nilai *default* yang digunakan oleh penulis adalah $z = 1$ dan $k = 1$.

Kalibrasi dilakukan dengan melakukan percobaan mengganti nilai *z* dan k, masing-masing melalui rentang [0.8,1.3]. Dengan pengetahuan bahwa nilai *z* dan k berbanding lurus dengan x, maka hipotesis penulis adalah sebagai berikut: semakin kecil nilai *z* dan k akan membuat semakin sedikit paragraf yang terdeteksi sebagai plagiat, karena nilai pembatas antara terindikasi dan tidak terindikasi akan semakin kecil, yang menyebabkan nilai-nilai yang sebelumnya cukup kecil untuk diindikasi plagiat, akan menjadi lebih besar daripada nilai batas tersebut sehingga jadi tidak

terindikasi plagiat.. Demikian juga sebaliknya, semakin besar nilai *z* dan k akan membuat semakin banyak paragraf yang terdeteksi sebagai plagiat, karena nilai pembatas antara terindikasi dan tidak terindikasi akan semakin besar, yang menyebabkan nilai-nilai yang sebelumnya cukup besar untuk tidak diindikasi plagiat, akan menjadi lebih kecil daripada nilai batas tersebut sehingga jadi terindikasi plagiat.

Hasil pemilihan nilai *z* dan k yang memberikan nilai *Precision*, *Recall*, dan *F-Measure* yang tinggi akan kemudian disimpan oleh penulis sebagai parameter-parameter terbaik dalam menggunakan metode distribusi normal.

3.5.2. *Kalibrasi Pada Algoritma Learning Vector Quantization*

Kalibrasi pada algoritma LVQ dapat dilakukan pada fase *training*, dimana algoritma ini memiliki parameter *alpha* dan *n* yang mempengaruhi kecepatan dan ketepatan pembelajaran. Penggunaan parameter *alpha* dan *n* yang tepat sangat esensial untuk dapat menghasilkan bobot terbaik, agar fase *testing* dapat menghasilkan indeks-indeks yang sesuai dengan pembacaan manual oleh penulis.

Secara teoritis, semakin besar nilai *alpha* dan *n*, maka pembelajaran akan berjalan semakin lambat, karena program akan semakin lambat dalam mencapai *stopping criteria*. Sisi positif dari fenomena ini adalah pembelajaran yang terjadi akan semakin teliti, karena nilai *alpha* yang besar akan menyebabkan perubahan bobot setiap *epoch* nya akan mengecil, sedangkan nilai *n* yang besar akan menyebabkan perubahan *alpha* setiap *epoch* nya akan semakin mengecil. Permasalahan dari nilai *alpha* dan *n* yang besar adalah terdapat kemungkinan bahwa koreksi bobot tidak bisa mencapai ideal karena koreksi yang dilakukan cenderung sedikit, sehingga bobot tidak dapat berubah secara signifikan.

Semakin kecil nilai *alpha* dan *n*, maka pembelajaran akan berjalan semakin cepat, karena program akan semakin cepat dalam mencapai *stopping criteria*. Sisi negatif dari fenomena ini adalah pembelajaran yang terjadi akan semakin tidak teliti, karena nilai *alpha* yang kecil akan menyebabkan perubahan bobot setiap *epoch* nya akan membesar, sedangkan nilai *n* yang besar akan menyebabkan perubahan *alpha* setiap *epoch* nya akan semakin membesar, yang membuka

kemungkinan nilai bobot yang ideal akan terlompati oleh algoritma. Sisi positif dari nilai *alpha* dan *n* yang kecil adalah terdapat kemungkinan bahwa koreksi bobot dapat mencapai nilai ideal karena perubahan yang besar memungkinan algoritma untuk melakukan koreksi bobot secara signifikan.

Penilaian nilai *alpha* dan *n* terbaik akan dilihat melalui seberapa banyak indeks yang dikembalikan oleh algoritma yang sesuai dengan nilai hasil pembacaan manual oleh penulis, pada fase *testing*. Nilai *alpha* dan *n* yang terbaik akan disimpan oleh penulis sebagai parameter-parameter terbaik dalam menggunakan metode LVQ.

3.5.3. *Variasi Implementasi Metode Pengelompokan Hasil Deteksi Plagiarisme*

Selain menentukan parameter-parameter terbaik yang diperlukan untuk memastikan performa dari masing-masing metode maupun algoritma, peningkatan performa sistem deteksi plagiarisme dapat dilakukan dengan melakukan variasi implementasi metode pengelompokan hasil deteksi plagiarisme, untuk menentukan metode implementasi yang terbaik, untuk kemudian diimplementasikan menggunakan metode dan algoritma yang sudah memiliki parameternya yang terbaik. Berikut adalah daftar variaisi metode implementasi yang dapat dilakukan:

- Melakukan variasi pada penggunaan data. Dengan tersedianya 3 nilai yang perlu dikelompokkan (*frobenius norm*, sudut *slice*, dan sudut *pad*), maka implementasi metode algoritma dapat divariasikan sebagai berikut:
    - Untuk metode distribusi normal, terdapat 7 variasi keluaran data yang dapat dicoba, yakni: operasi AND dari ketiga nilai (*frobenius norm*, sudut *slice*, dan sudut *pad*), hanya menggunakan salah satu dari tiga nilai yang ada (hanya *frobenius norm*, hanya sudut *slice*, atau hanya sudut *pad*), atau menggunakan operasi AND pada dua dari tiga nilai yang ada (*frobenius norm* AND *slice*, *frobenius norm* AND *pad*, atau *slice* AND *pad*).
    - Untuk algoritma LVQ juga dapat 7 variasi keluaran data yang dapat dicoba. Hal ini dilakukan dengan memvariasikan parameter input

pada algoritma. Sama seperti metode distribusi normal, variasi yang dapat dicoba yaitu: input 3 dimensi (*frobenius norm*, sudut *slice*, dan sudut *pad*), input hanya 1 dimensi (hanya *frobenius norm*, hanya sudut *slice*, atau hanya sudut *pad*), atau input 2 dimensi (*frobenius norm* dan *slice*, *frobenius norm* dan *pad*, atau *slice* dan *pad*).

- Menggabung hasil dari metode distribusi normal dengan algoritma LVQ. Penggabungan metode ini dapat dilakukan dengan beberapa cara, yakni:
    - Melakukan operasi AND pada hasil metode distribusi normal dan algoritma LVQ, sehingga hanya jika kedua metode mengeluarkan hasil "plagiat" untuk perbandingan paragraf yang sama, barulah metode ini memberi hasil "plagiat". Jika untuk perbandingan paragraf yang sama pada kedua metode, salah satunya memberikan hasil "tidak", maka metode ini akan memberi hasil "tidak". Dasar dari variasi ini adalah untuk mengurangi nilai-nilai *false positives*, karena jika salah satu metode memberikan nilai "plagiat" namun salah satunya lagi memberikan nilai "tidak", maka keluaran dari metode ini akan memberikan nilai "tidak". Namun metode ini harus memastikan agar jangan sampai *true positives* pada kedua metode bertemu, agar jumlah *true positives* yang berubah menjadi *false negatives* dapat ditekan.
    - Menjalankan algoritma LVQ terlebih dahulu, kemudian berusaha mengurangi nilai *false positives* yang terjadi dengan memasukkan kembali seluruh keluaran yang bernilai "plagiat" pada metode distribusi normal, dengan harapan nilai-nilai *false positives* akan berubah menjadi *true negatives*.
    - Menjalankan metode distribusi normal terlebih dahulu, kemudian berusaha mengurangi nilai *false negatives* yang terjadi dengan memasukkan kembali seluruh keluaran yang bernilai "tidak" pada metode LVQ, dengan harapan nilai-nilai *false negatives* akan berubah menjadi *true positives*.

```
/* OPERASI AND PADA DISTRIBUSI NORMAL DAN LVQ */
GET input; //mengambil seluruh input
out_normdist = NORMAL DIST input; //memproses input dengan metode distribusi
normal
out_lvq = LVQ input; //memproses input dengan algoritma LVQ
out_final = out_normdist AND out_lvq; //keluaran final adalah hasil AND dari
keluaran masing-masing metode
_____________________________________________________________________________

/* LVQ, KEMUDIAN DISTRIBUSI NORMAL UNTUK NILAI-NILAI "PLAGIAT" */
GET input; //mengambil seluruh input
out = LVQ input; //memproses input dengan algoritma LVQ
FOREACH(out AS value){ //untuk setiap nilai output oleh LVQ
        IF(value == "plagiat"){ //jika ada nilai plagiat
                in_normdist = ARRAY_PUSH value; //masukkan ke kelompok data yang
akan melalui distribusi normal
        }
}
out_normdist = NORMAL DIST in_normdist; //memproses input "plagiat" dengan
distribusi normal
out_final = UPDATE out SET out_normdist WHERE out.key = out_normdist.key;
//memperbaharui daftar keluaran dengan daftar keluaran oleh distribusi normal
_____________________________________________________________________________

/* DISTRIBUSI NORMAL, KEMUDIAN LVQ UNTUK NILAI-NILAI "TIDAK" */
GET input; //mengambil seluruh input
out = NORMAL DIST input; //memproses input dengan metode distribusi normal
FOREACH(out AS value){ //untuk setiap nilai output oleh distribusi normal
        if(value == "tidak"){ //jika ada nilai tidak
                in_lvq = ARRAY_PUSH value; //masukkan ke kelompok data yang akan
melalui LVQ
        }
}
out_lvq = LVQ in_lvq; //memproses input "ya" algoritma LVQ
out_final = UPDATE out SET out_lvq WHERE out.key = out_lvq.key; //memperbaharui
daftar keluaran dengan daftar keluaran oleh LVQ
```

Gambar 3.6: *Pseudocode* Variasi Implementasi Metode Pengelompokan Data Hasil Deteksi Plagiarisme

BAB 4
HASIL UJI COBA DAN ANALISA

4.1. Proses Penerapan Program Deteksi Plagiarisme dan Hasil Pengujian Program Deteksi Plagiarisme

Proses yang dilakukan dalam menerapkan dan menganalisa hasil pengujian program deteksi plagiarisme dimulai dengan mengumpulkan sejumlah karya tulis yang akan digunakan untuk dimasukkan pada sistem, diproses oleh sistem deteksi plagiarisme sehingga menghasilkan keluaran yang akan dinilai oleh program penguji. Proses pengumpulan karya tulisnya adalah sebagai berikut:

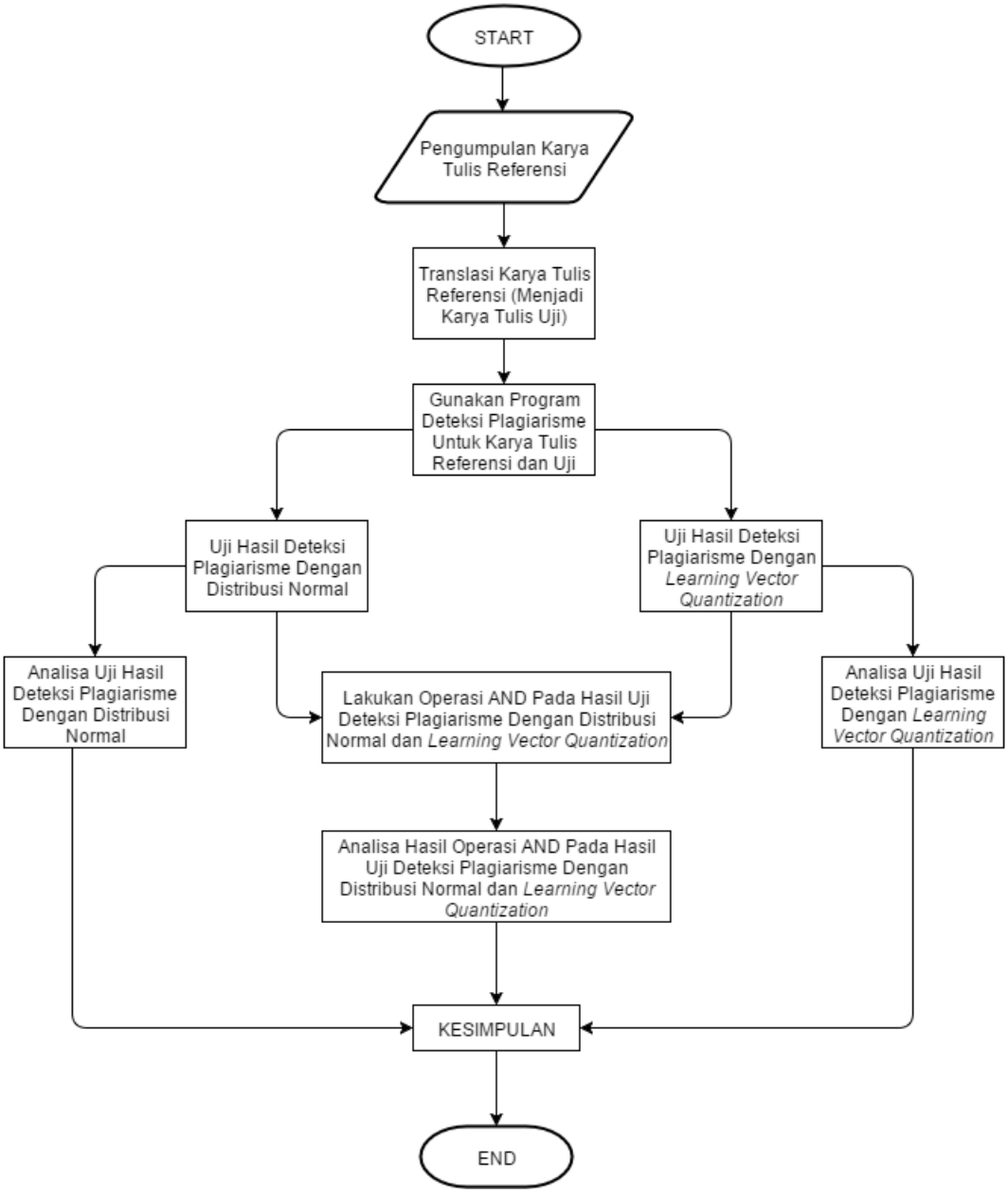
Penulis mengumpulkan 10 karya tulis berbahasa Inggris sebagai karya tulis referensi. Kemudian kesepuluh karya tulis tersebut akan diterjemahkan ke bahasa Indonesia secara manual (tidak menggunakan mesin penerjemah, misalnya Google Translate). Kemudian, penulis akan melakukan penilaian secara manual terhadap seluruh paragraf-paragraf karya tulis tersebut, apakah ada potensi plagiarisme antara paragraf yang satu dengan paragraf lainnya. Pengecekan oleh program akan dilakukan antara karya tulis referensi dengan karya tulis hasil translasi langsungnya yang dijadikan karya tulis uji. Sehingga, terdapat 10 hasil perbandingan karya tulis.

Tema dari kesepuluh karya tulis tersebut adalah: desain *dual band-pass filter*, pendidikan jarak jauh, pemanasan global, simulasi *cloud computing* dengan MATLAB, *online game*, sistem penilai essay otomatis dengan *Simple-O*, sistem penilai essay otomatis dengan *Latent Semantic Analysis* sejumlah 2 karya tulis, gempa bumi, dan *healing music*.

Proses deteksi plagiarisme dilakukan menggunakan program berbasis *Latent Semantic Analysis*. Program ini akan memunculkan tiga besaran, yakni *Frobenius Norm*, sudut dua vektor (*slice*), dan sudut dua vektor (*pad*) yang kemudian akan digunakan oleh program penguji hasil deteksi plagiarisme (distribusi normal dan *learning vector quantization*).

Pengujian hasil deteksi plagiarisme dilakukan dalam tiga cara, yakni menggunakan distribusi normal saja, menggunakan *learning vector quantization*

saja, dan menggunakan operasi AND untuk menggabungkan hasil pengujian oleh distribusi normal dan *learning vector quantization*.



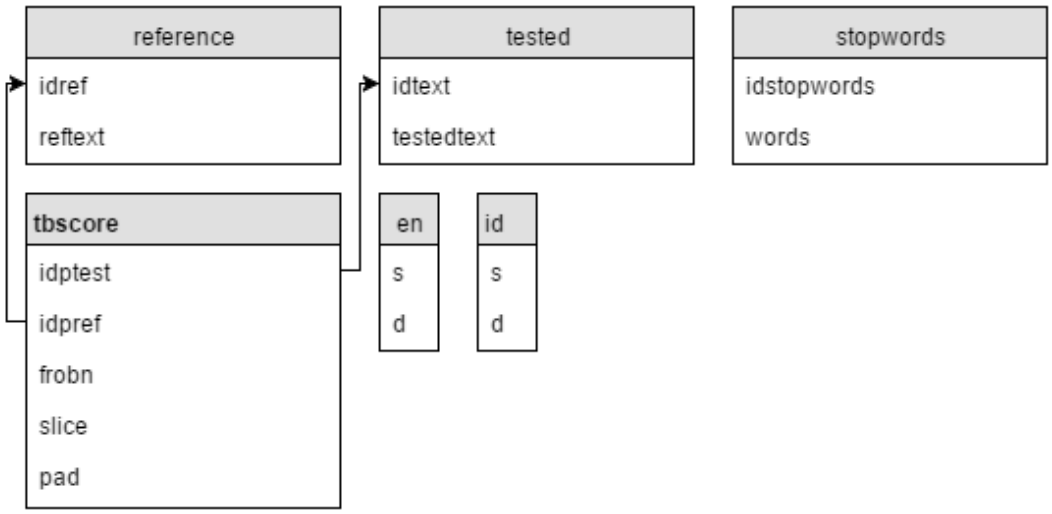
Gambar 4.1: Diagram Proses Penerapan Program Deteksi Plagiarisme dan Hasil Pengujian Program Deteksi Plagiarisme

Hasil pengecekan oleh program akan dibandingkan degan hasil pengecekan manual, yang akan menghasilkan tiga nilai yang kemudian akan dianalisa dan dijadikan acuan untuk mengambil kesimpulan. Ketiga nilai tersebut adalah *Precision, Recall*, dan *F-Measure* dengan bobot 1. *Precision* akan menjelaskan tentang kemampuan program penguji hasil deteksi dalam meminimalisir kejadian

false positive, *Recall* akan menjelaskan tentang kemampuan program penguji hasil deteksi dalam meminimalisir kejadian *false negative*, sedangkan *F-Measure*, rata-rata harmonik berbobot antara *Recall* dan *Precision*, dengan bobot 1 akan memberikan bobot yang sama baik bagi *Recall* dan *Precision*.

4.2. Program Deteksi Plagiarisme Menggunakan Latent Semantic Analysis

Program deteksi plagiarisme menggunakan *latent semantic analysis* dibangun menggunakan bahasa pemrograman PHP yang dijalankan di *web browser* di atas *Apache Web Server*. Seluruh data (karya tulis, kamus translasi, daftar *stopwords*, dan hasil pemrosesan program) disimpan menggunakan MySQL sebagai *Relational Database Management System*. *Apache Web Server*, PHP, dan MySQL Server adalah bagian dari XAMPP, yakni kumpulan perangkat lunak yang membangun *stack* yang berfungsi untuk menjalankan *web services*.



Gambar 4.2: Skema Relasi *Database* Program Deteksi Plagiarisme

Gambar di atas menjelaskan tentang penggunaan *database* dalam program deteksi plagiarisme. Tabel *reference* berisi paragraf-paragraf dari karya tulis referensi, dimana paragraf disimpan di kolom *reftext* dengan identitasnya pada *idref*. Tabel *tested* berisi paragraf-paragraf dari karya tulis uji, dimana paragraf disimpan di kolom *testedtext* dengan identitasnya pada *idtext*. Kolom *idref* dan *idtext* memiliki ketentuan sebagai berikut: tiga digit terakhir menentukan nomor paragraf pada karya tulis tersebut, sedangkan digit-digit di belakangnya

mengidentifikasikan karya tulis. Misalnya entri dengan *id* 1000 berarti paragraf yang berkorespondensi dengan *id* tersebut adalah paragraf ke nol dari karya tulis ke 1.

Tabel *stopwords* menyimpan daftar *stopwords* pada kolom *words* dan identifikasi bahasanya pada kolom *idstopwords*.

Kamus translasi disimpan di dua tabel, yakni tabel *en* dan tabel *id*. Tabel *en* menyimpan daftar translasi dari bahasa Inggris (sumber) ke bahasa Indonesia (tujuan), sedangkan tabel *id* menyimpan daftar translasi dari bahasa Indonesia (sumber) ke bahasa Inggris (tujuan). Kolom *s* berarti *source*, yakni kata dalam bahasa sumber (misalnya kolom *s* pada *en* berarti daftar kata bahasa Inggris sebagai bahasa sumber), sedangkan kolom *d* berarti *destination*, yakni kata dalam bahasa tujuan (misalnya kolom *d* pada *en* berarti daftar kata bahasa Indonesia sebagai bahasa tujuan).

Tabel *tbscore* berisi nilai keluaran hasil program deteksi plagiarisme yang kemudian akan digunakan oleh program penguji hasil deteksi plagiarisme. Kolom *idptest* mengidentifikasikan paragraf karya tulis uji, kolom *idpref* mengidentifikasikan paragraf karya tulis referensi. Pasangan *idptest* dan *idpref* menunjukkan pasangan paragraf yang telah diuji indikasi plagiarismenya oleh program deteksi plagiarisme. Kolom *fnorm* berisi nilai perbandingan panjang vektor keluaran hasil program deteksi plagiarisme, kolom *slice* berisi nilai sudut hasil perhitungan sudut kedua vektor diagonal hasil SVD dengan pemotongan dimensi vektor yang lebih besar, dan kolom *pad* berisi nilai sudut hasil perhitungan sudut kedua vektor diagonal hasil SVD dengan penambahan dimensi vektor yang lebih kecil dengan nilai nol. Kolom *idpref* dan *idptest* pada tabel *tbscore* memiliki *foreign key constraint*, dimana *idpref* memiliki referensi pada kolom *idref* pada tabel *reference*, dan *idptest* memiliki referensi pada kolom *idtext* pada tabel *tested*. *Foreign key constraint* pada kedua kolom tersebut memiliki ketentuan ON DELETE RESTRICT, sehingga data pada tabel *tbscore* tidak bisa dihapus sebelum data pada tabel terkait dihapus terlebih dahulu.

Eksekusi program dilakukan dengan Berikut ini adalah contoh hasil deteksi plagiarisme menggunakan *latent semantic analysis* pada satu karya tulis dengan 9 paragraf:

Idptest	Idpref	Fnorm	Slice	pad
3001	**3001**	60.6977	5.95616	5.95616
3001	**3002**	35.3553	47.6318	47.6318
3001	**3003**	0	90	90
3001	**3004**	0	90	90
3001	**3005**	28.4747	7.26953	7.26953
3001	**3006**	0	90	90
3001	**3007**	0	90	90
3001	**3008**	0	90	90
3001	**3009**	14.2857	58.706	58.706
3002	**3001**	53.8028	21.0574	21.0574
3002	**3002**	55.9017	24.8395	24.8395
3002	**3003**	0	90	90
3002	**3004**	32.1634	35.0656	35.0656
3002	**3005**	0	90	90
3002	**3006**	0	90	90
3002	**3007**	0	90	90
3002	**3008**	0	90	90
3002	**3009**	0	90	90
3003	**3001**	0	90	90
3003	**3002**	0	90	90
3003	**3003**	34.641	27.564	27.564
3003	**3004**	18.5695	54.04	54.04
3003	**3005**	16.4399	42.0606	42.0606
3003	**3006**	0	90	90
3003	**3007**	0	90	90
3003	**3008**	41.7029	46.246	46.246
3003	**3009**	0	90	90
3004	**3001**	0	90	90
3004	**3002**	0	90	90
3004	**3003**	0	90	90
3004	**3004**	89.0564	6.74606	25.498
3004	**3005**	23.2495	27.0031	27.0031
3004	**3006**	0	90	90
3004	**3007**	22.9416	43.4915	43.4915
3004	**3008**	41.7029	46.246	46.246
3004	**3009**	31.9438	47.1346	47.1346
3005	**3001**	16.2221	44.2492	44.2492
3005	**3002**	0	90	90
3005	**3003**	0	90	90
3005	**3004**	32.1634	35.0656	35.0656
3005	**3005**	91.5335	12.3201	20.2599
3005	**3006**	0	90	90

3005	**3007**	22.9416	43.4915	43.4915
3005	**3008**	0	90	90
3005	**3009**	20.2031	44.9014	44.9014
3006	**3001**	0	90	90
3006	**3002**	0	90	90
3006	**3003**	0	90	90
3006	**3004**	0	90	90
3006	**3005**	23.2495	42.0606	42.0606
3006	**3006**	53.8816	11.7788	22.6378
3006	**3007**	0	90	90
3006	**3008**	41.7029	46.246	46.246
3006	**3009**	14.2857	58.706	58.706
3007	**3001**	0	90	90
3007	**3002**	0	90	90
3007	**3003**	0	90	90
3007	**3004**	0	90	90
3007	**3005**	16.4399	42.0606	42.0606
3007	**3006**	0	90	90
3007	**3007**	51.2989	1.50848	26.6047
3007	**3008**	0	90	90
3007	**3009**	14.2857	56.8207	58.706
3008	**3001**	0	90	90
3008	**3002**	0	90	90
3008	**3003**	0	90	90
3008	**3004**	0	90	90
3008	**3005**	0	90	90
3008	**3006**	17.9605	48.0955	48.0955
3008	**3007**	0	90	90
3008	**3008**	85.9727	7.88509	7.88509
3008	**3009**	0	90	90
3009	**3001**	16.2221	44.2492	44.2492
3009	**3002**	0	90	90
3009	**3003**	0	90	90
3009	**3004**	37.1391	90	90
3009	**3005**	16.4399	42.0606	42.0606
3009	**3006**	0	90	90
3009	**3007**	22.9416	43.4915	43.4915
3009	**3008**	0	90	90
3009	**3009**	85.7143	15.4776	21.1637

Tabel 4.1: Tabel *tbscore* Hasil Program Deteksi Plagiarisme Menggunakan *Latent Semantic Analysis*

Entri yang di-*highlight* berwarna kuning adalah hasil deteksi plagiarisme antara paragraf referensi dengan paragraf uji yang merupakan hasil translasi langsung dari paragraf referensi terkait (dimana kolom *idpref* dan *idptest* dari entri tersebut bernilai sama). Secara umum, entri-entri yang di-*highlight* tersebut memiliki kencenderungan seperti berikut: 1) nilai *fnorm* yang tinggi dibandingkan entri lainnya dan 2) nilai *slice* dan *pad* yang lebih rendah dibandingkan entri lainnya. Hal ini sesuai dasar teori dimana jika nilai *fnorm* semakin mendekati 100, maka kedua paragraf cenderung terindikasi plagiat, dan jika nilai sudut semakin mendekati nol, maka kedua paragraf cenderung terindikasi plagiat.

Namun ada satu paragraf yang memiliki nilai yang tidak sesuai dasar teori, yakni paragraf 3003. Nilai *fnorm* yang hanya 34.641 dianggap terlalu jauh dari nilai ideal (100), dan kedua nilai sudut mencapai 27.564^{o}, yang dianggap terlalu jauh dari nilai ideal (0).

Berikut ini adalah daftar kata-kata pada paragraf referensi dengan ID 3003, dan hasil translasi paragraf uji dari bahasa Indonesia ke bahasa Inggris:

Paragraf referensi	**Paragraf uji yang sudah ditranslasi**
generally	in, in, general
lot	common, public
reasons	alasan
make	membuat
global	
warming	pemanasan
use	
excessive	berlebihan
fossil	fossil
fuels	burn, fire, bake, roast
much	
vehicles	vehicle
increasing	
main	principal, prime
reason	alasan
besides	

livestock	animal
agriculture	husbandry, agriculture
also	
contribution	memiliki
using	menggunakan
excessive	berlebihan
methane	metana
gas	gas, gasoline
fertilizers	manure, fertilizer
-	Matter, substance, material, material (berasal dari kata "bahan" dari bahan bakar)

Tabel 4.2: Tabel Perbandingan Paragraf Referensi dan Hasil Translasi Paragraf Uji Pada Paragraf dengan *ID* 3003

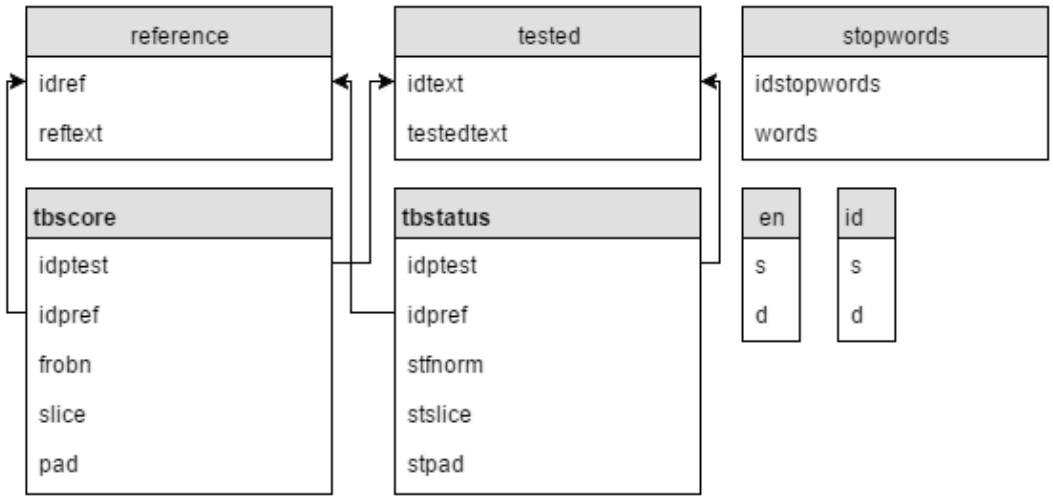
Dari tabel di atas dapat disimpulkan bahwa proses translasi yang sangat tidak akurat (hanya 3 kata yang melewati proses translasi dengan benar, yakni *agriculture*, *gas*, dan *fossil*) menyebabkan tidak terbentuknya matriks *term-document* yang sesuai antara paragraf referensi dan paragraf uji, sehingga hasil *singular value decomposition* menghasilkan matriks diagonal yang cukup berbeda, sehingga pada akhirnya menghasilkan nilai *fnorm*, *slice*, dan *pad* yang tidak sesuai harapan.

Namun, secara umum, dari 9 paragraf yang seharusnya dideteksi plagiat, 8 paragraf lainnya mengembalikan hasil yang cukup baik untuk kemudian diuji oleh program penguji hasil deteksi plagiarisme.

4.3. Pengujian Hasil Deteksi Plagiarisme Menggunakan Distribusi Normal

Hasil keluaran program deteksi plagiarisme menggunakan *latent semantic analysis* kemudian akan diuji menggunakan metode distribusi normal. Untuk itu, *database* memerlukan tabel tambahan, yakni tabel *tbstatus*. Tabel *tbstatus* berisi nilai keluaran hasil uji program deteksi plagiarisme. Kolom *idptest* mengidentifikasikan paragraf karya tulis uji, kolom *idpref* mengidentifikasikan

paragraf karya tulis referensi. Pasangan *idptest* dan *idpref* menunjukkan pasangan paragraf yang telah diuji program uji hasil deteksi plagiarisme. Kolom *stfnorm* berisi hasil pengujian berdasarkan nilai *fnorm* yang ada pada tabel *tbscore.* Kolom *stslice* berisi hasil pengujian berdasarkan nilai *slice* yang ada pada tabel *tbscore*, sedangkan kolom *stpadd* berisi hasil pengujian berdasarkan nilai *pad* yang ada pada tabel *tbscore*. Entri dari ketiga kolom ini dapat berisi dua hal, yakni "plagiat" atau "tidak", yang menyatakan indikasi plagiat dari hasil deteksi plagiarisme paragraf dengan *idptest* dan *idpref* terkait.



Gambar 4.3: Skema Relasi Database Program Penguji Hasil Deteksi Plagiarisme

Kolom *idpref* dan *idptest* pada tabel *tbscore* memiliki *foreign key constraint*, dimana *idpref* memiliki referensi pada kolom *idref* pada tabel *reference*, dan *idptest* memiliki referensi pada kolom *idtext* pada tabel *tested*. *Foreign key constraint* pada kedua kolom tersebut memiliki ketentuan ON DELETE RESTRICT, sehingga data pada tabel *tbscore* tidak bisa dihapus sebelum data pada tabel terkait dihapus terlebih dahulu.

Proses pengujian hasil deteksi plagiarisme dengan metode distribusi normal dimulai dengan memilih karya tulis uji yang ingin diuji indikasi plagiarismenya. Kemudian, program akan membandingkan karya tulis uji tersebut dengan karya tulis referensi yang menjadi sumber karya tulis uji tersebut. Karena terdapat 10 karya tulis, maka pengujian akan diulang sebanyak 10 kali. Hasil pengujian akan mengembalikan indikasi plagiat pada tiga parameter, yakni *frobenius norm*, *slice*, dan *pad*.

4.3.1. *Analisa Hasil Variasi Nilai Pengali z-score*

Berdasarkan rumus (3.4), nilai pengali *z-score* yang dimaksud pada rumus penentuan batas distribusi $x = k(\mu + z\sigma)$ adalah variabel *k*. Penulis memvariasikan variabel *k* dari 0.8 hingga 1.3 dengan selisih nilai tiap variasinya sebesar 0.1. Dalam melakukan penerapan variasi, parameter-parameter lain memiliki ketetapan sebagai berikut:

- Hasil pengujian yang digunakan adalah operasi AND dari *frobenius norm*, *slice*, dan *pad*.
- Nilai *z-score* yang digunakan adalah -1.
- Jumlah karya tulis yang dicoba sejumlah 10 yang memberikan 4050 hasil pengujian.
- Dari 4050 hasil pengujian, terdapat 194 paragraf yang seharusnya terdeteksi plagiat, dan 3856 paragraf yang seharusnya tidak terdeteksi plagiat.

Tabel di bawah ini adalah hasil pengujian dengan memvariasikan nilai *k* pada metode distribusi normal:

k	**0.8**	**0.9**	**1**	**1.1**	**1.2**	**1.3**
True Negative	3831	3804	3765	3667	3546	3391
True Positive	75	90	105	112	123	134
True Negative + True Positive	3906	3894	3870	3779	3669	3525
False Negative	119	104	89	82	71	60
False Positive	25	52	91	189	310	465
False Negative + False Positive	144	156	180	271	381	525
Negatives	3950	3908	3854	3749	3617	3451
Positives	100	142	196	301	433	599
Precision	0.75	0.633803	0.535714	0.372093	0.284065	0.223706
Recall	0.386598	0.463918	0.541237	0.57732	0.634021	0.690722
F-Measure	0.510204	0.535714	0.538462	0.452525	0.392344	0.337957

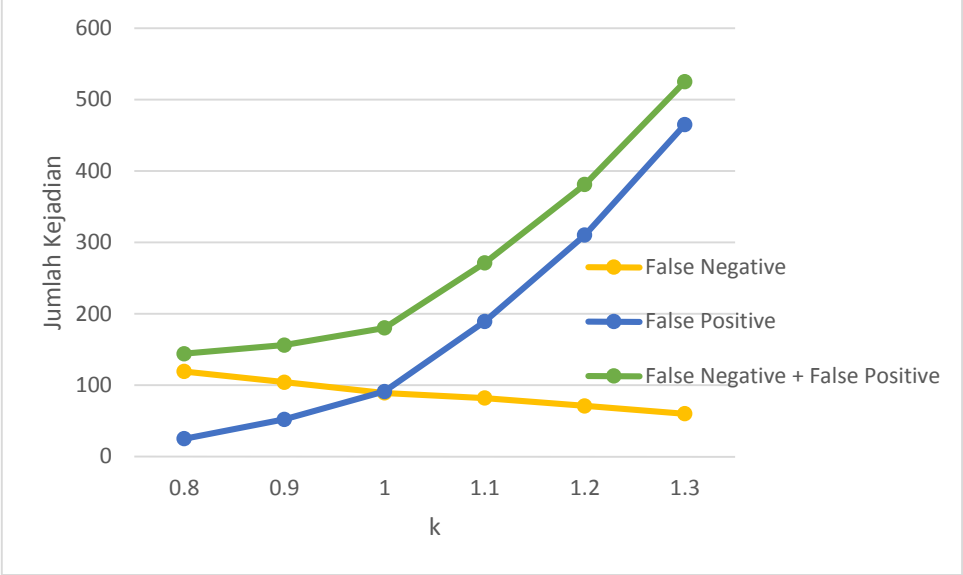
Tabel 4.3: Hasil Variasi Nilai k Pada Pengujian Hasil Deteksi Plagiarisme Metode Distribusi Normal

Tabel di atas memperlihatkan bahwa semakin besar nilai *k*, semakin sedikit nilai *true positives + true negatives*, yang berarti akurasi dari program penguji semakin berkurang. Selain itu, semakin besar nilai *k*, semakin banyak juga hasil

deteksi yang bernilai positif, sedangkan hasil deteksi yang bernilai negatif semakin berkurang.

Namun, akurasi program tidak dapat disimpulkan hanya dari nilai *true positives* dan *true negatives*, maupun jumlah *positives* dan *negatives* yang dihasilkan oleh program terutama dengan tidak seimbangnya nilai *positives* dan *negatives* yang sebenarnya (194 untuk *positives* dan 3856 untuk *false negatives*, perbandingannya yakni 1 : 19.876). Untuk itu, diperlukan parameter lain untuk mengetahui akurasi program.

Representasi grafis dari beberapa besaran pada tabel di atas dapat dilihat pada kedua grafik dibawah ini:

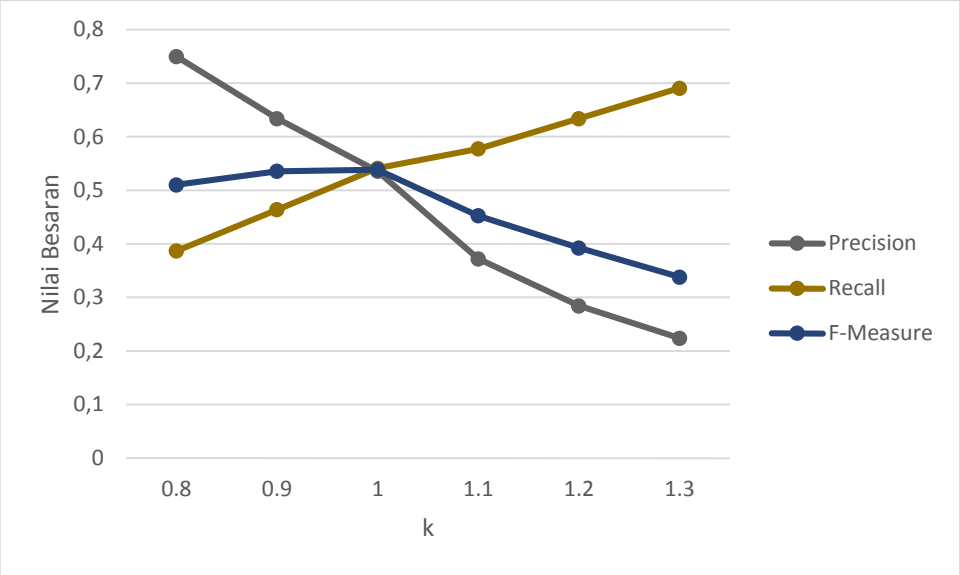


Grafik 4.1: Pengaruh Variasi Nilai k Pada Nilai False Positives dan False Negatives Pada Metode Distribusi Normal

Dari grafik di atas, pengaruh variabel pengali *z-score* (*k*) memberikan pengaruh sebagai berikut:

- Semakin besar nilai *k*, nilai *false negatives* turun, namun penurunannya semakin kecil seiring naiknya nilai *k*.
- Semakin besar nilai *k*, nilai *false positives* naik, dan kenaikannya cenderung semakin besar seiring naiknya nilai *k*.

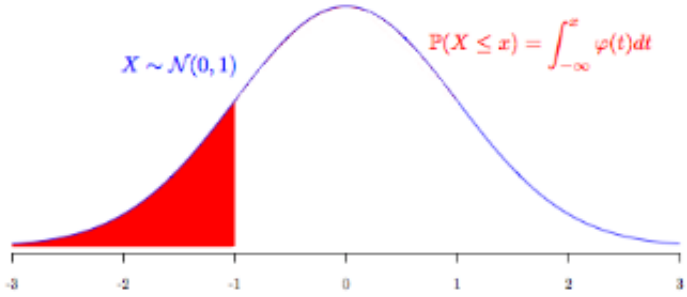
- Sehingga, semakin besar nilai *k*, akan semakin banyak hasil pengujian yang salah (*false*).



Grafik 4.2: Pengaruh Variasi Nilai k Pada Besaran Pengukuran Performa Information Retrieval Pada Metode Distribusi Normal

Grafik di atas menjelaskan pengaruh lebih jauh dari variabel pengali *z-score* (*k*) terkait dengan nilai *false positives* dan *false negatives* melalui nilai besaran *information retrieval* (*Precision*, *Recall*, dan *F-Measure*). Penjelasannya adalah sebagai berikut:

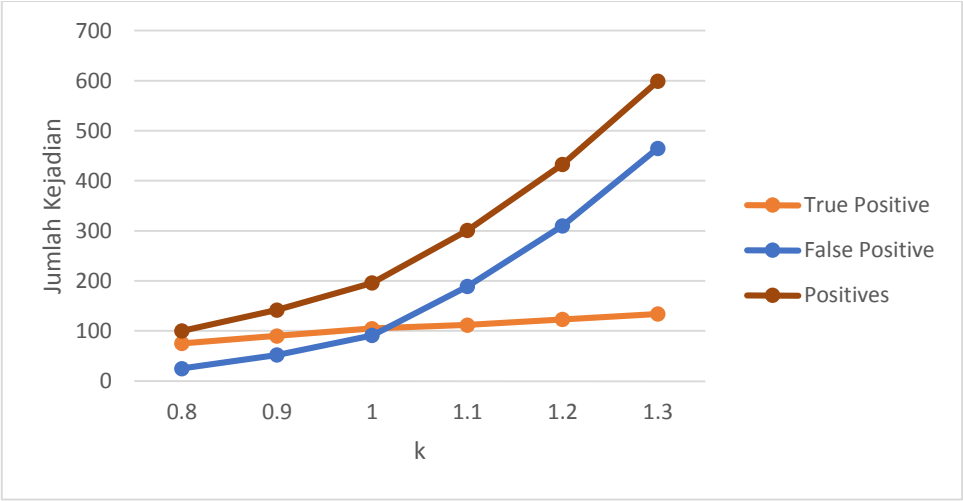
- Semakin besar nilai *k*, nilai *Precision* turun, dengan sedikit kecenderungan penurunannya semakin besar seiring naiknya nilai *k*.
- Semakin besar nilai *k*, nilai *Recall* semakin tinggi, dan kenaikannya cenderung semakin sedikit seiring naiknya nilai *k*.
- Akibat turunnya nilai *Precision* yang cenderung semakin tajam, sedangkan kenaikan nilai *Recall* yang semakin landai, nilai *F-Measure* mengalami penuruan seiring naiknya nilai *k*. Bahkan terlihat bahwa grafik *F-Measure* cenderung mengikuti grafik *Precision*.



Gambar 4.4: Daerah Terindikasi Plagiat (Merah) dan Tidak Terindikasi Plagiat (Putih) Pada Batas *Z-Score* Sebesar -1 dan Faktor Pengali (*k*) Sebesar 1

Analisa terkait hasil variasi variabel *k* di atas dapat dijelaskan sebagai berikut: Sesuai rumus (3.4), yakni $x = k(\mu + z\sigma)$, maka semakin besar nilai *k*, maka nilai batas *x* akan semakin besar, sehingga dapat disimpulkan bahwa daerah yang nilai-nilainya terindikasi plagiat akan semakin besar, sehingga jumlah paragraf yang dideteksi sebagai plagiat (nilai *positives*) akan semakin banyak. Analisa di atas diperkuat dengan gambar 4.4 yang menyatakan bahwa kenaikan nilai *k* akan memperluas daerah yang berwarna merah, sehingga akan semakin memperbanyak jumlah paragraf yang terdeteksi sebagai plagiat.

Namun, kenaikan nilai *positives* selain meningkatkan jumlah paragraf yang terindikasi *true positive* (dimana hal ini diinginkan), hal ini juga berdampak pada semakin banyaknya paragraf yang terindikasi *false positive* (tidak diinginkan). Bahkan kenaikan nilai *false positives* lebih tinggi daripada kenaikan jumlah paragraf yang teridinikasi *true positives*, seperti pada grafik di bawah ini.



Grafik 4.3: Pengaruh Variasi Nilai k Pada Nilai *Positives* Pada Metode Distribusi Normal

Kesimpulannya, kenaikan nilai *k* berdampak positif pada aspek relevansi (*Recall*) program, namun berdampak negatif pada aspek presisi (*Precision*) dari program. Karena dampak negatif pada *Precision* lebih besar daripada dampak positif pada *Recall*, secara keseluruhan, kenaikan nilai *k* berdampak negatif pada keakuratan program pengujian sistem deteksi plagiarisme berdasarkan metode distribusi normal.

Keseimbangan terbaik antara *Recall* dan *Precision* dicapai pada nilai *k* sebesar 1, dimana pada nilai *k* tersebut didapatkan nilai *F-Measure* tertinggi, yakni 0.538462. Jika pengujian berhenti hingga tahap ini, maka nilai *k* 1 adalah nilai terbaik untuk akurasi program. Namun, Karena nilai-nilai di atas akan kembali digunakan bersama dengan hasil uji program deteksi menggunakan LVQ dengan operasi AND, maka program membutuhkan nilai *Recall* yang tinggi. Pada nilai *k* tersebut, nilai *Recall* yang didapatkan sangat rendah, yakni 0.541237, sehingga nilai tersebut juga tidak dapat dikatakan sebagai nilai terbaik untuk kemudian digabungkan dengan uji deteksi menggunakan LVQ.

4.3.2. *Analisa Hasil Perubahan Batas Z-Score*

Berdasarkan rumus (3.4), nilai *z-score* yang dimaksud pada rumus penentuan batas distribusi $x = k(\mu + z\sigma)$ adalah variabel *z.* Penulis memvariasikan variabel *z* dari -0.8 hingga -1.1 dengan selisih nilai tiap variasinya sebesar 0.1. Dalam melakukan penerapan variasi, parameter-parameter lain memiliki ketetapan sebagai berikut:

- Hasil pengujian yang digunakan adalah operasi AND dari *frobenius norm*, *slice*, dan *pad.*
- Nilai batas *z-score* (*k*) yang digunakan adalah 1.1
- Jumlah karya tulis yang dicoba sejumlah 10 yang memberikan 4050 hasil pengujian.
- Dari 4050 hasil pengujian, terdapat 194 paragraf yang seharusnya terdeteksi plagiat, dan 3856 paragraf yang seharusnya tidak terdeteksi plagiat.

Tabel di bawah ini adalah hasil pengujian dengan memvariasikan nilai *z* pada metode distribusi normal:

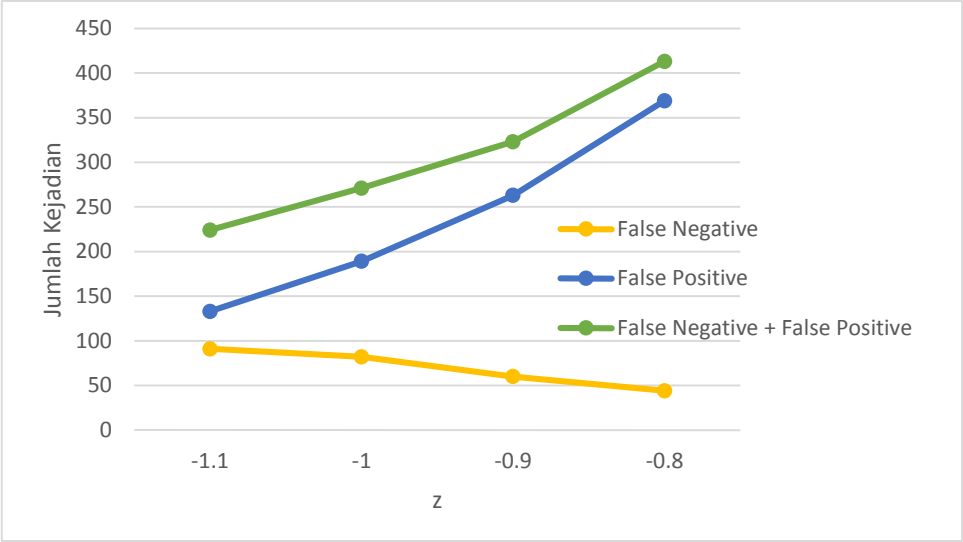
z	-1.1	-1	-0.9	-0.8
True Negative	3723	3667	3593	3487
True Positive	103	112	134	150
True Negative + True Positive	3826	3779	3727	3637
False Negative	91	82	60	44
False Positive	133	189	263	369
False Negative + False Positive	224	271	323	413
Negatives	3814	3749	3653	3531
Positives	236	301	397	519
Precision	0.436441	0.372093	0.337531	0.289017
Recall	0.530928	0.57732	0.690722	0.773196
F-Measure	0.47907	0.452525	0.453469	0.420757

Tabel 4.4: Hasil Variasi Nilai *z-score* Pada Pengujian Hasil Deteksi Plagiarisme Metode Distribusi Normal

Secara umum, kecenderungan hasil variasi nilai *z-score* sama seperti kecenderungan pada hasil variasi nilai *k.* Tabel di atas memperlihatkan bahwa semakin besar nilai *z-score*, semakin sedikit nilai *true positives* + *true negatives*, yang berarti akurasi dari program penguji semakin berkurang. Selain itu, semakin besar nilai *z-score*, semakin banyak juga hasil deteksi yang bernilai positif, sedangkan hasil deteksi yang bernilai negatif semakin berkurang.

Namun, akurasi program tidak dapat disimpulkan hanya dari nilai *true positives* dan *true negatives*, maupun jumlah *positives* dan *negatives* yang dihasilkan oleh program terutama dengan tidak seimbangnya nilai *positives* dan *negatives* yang sebenarnya (194 untuk *positives* dan 3856 untuk *false negatives*, perbandingannya yakni 1 : 19.876). Untuk itu, diperlukan parameter lain untuk mengetahui akurasi program.

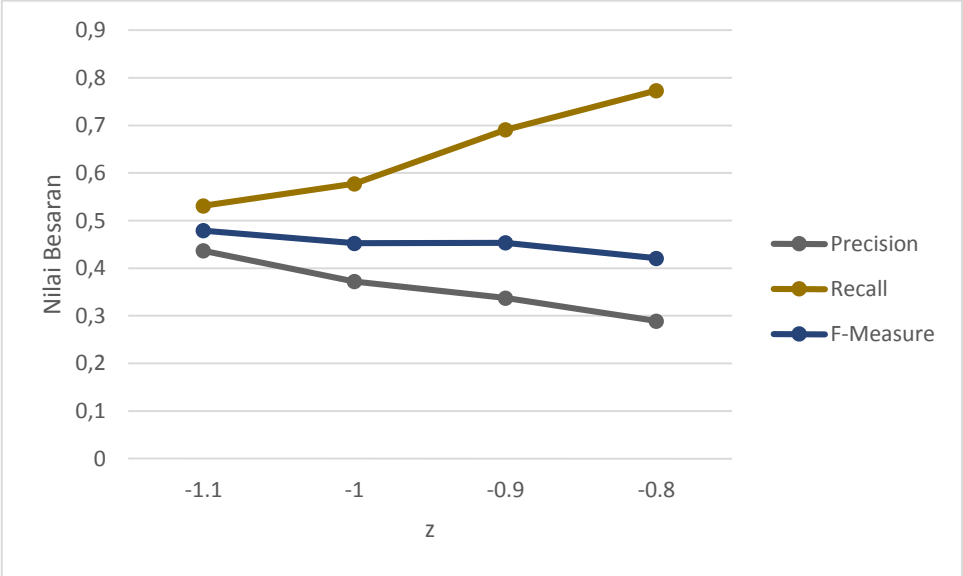
Representasi grafis dari beberapa besaran pada tabel di atas dapat dilihat pada kedua grafik dibawah ini:



Grafik 4.4: Pengaruh Variasi Nilai *z-score* Pada Nilai False Positives dan False Negatives Pada Metode Distribusi Normal

Dari grafik di atas, pengaruh variabel *z-score* (*z*) memberikan pengaruh sebagai berikut:

- Semakin besar nilai *z*, nilai *false negatives* turun.
- Semakin besar nilai *z*, nilai *false positives* naik, dan kenaikannya jauh lebih tinggi daripada penurunan nilai *false negatives*.
- Sehingga, semakin besar nilai *k*, akan semakin banyak hasil pengujian yang salah (*false*).



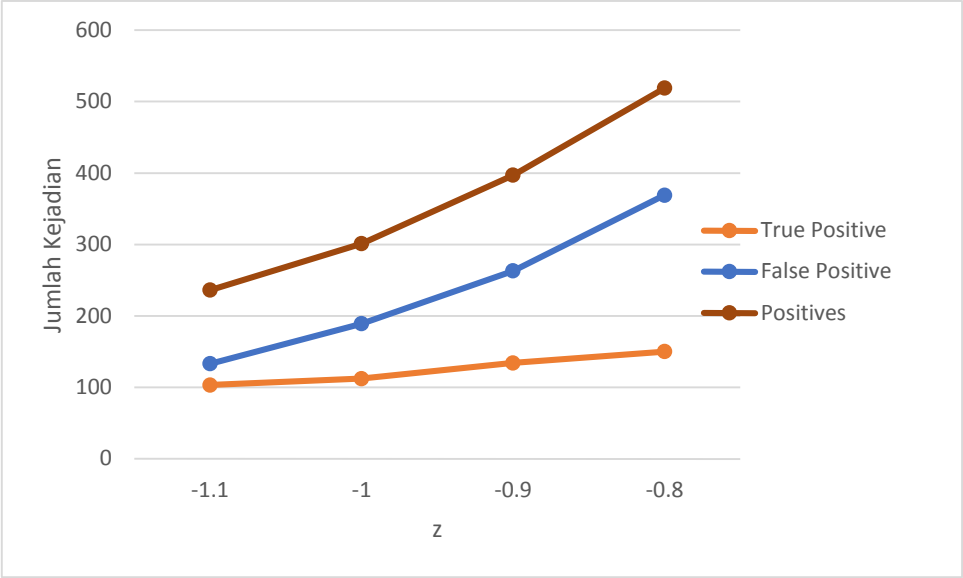
Grafik 4.5: Pengaruh Variasi Nilai *z-score* Pada Besaran Pengukuran Performa *Information Retrieval* Pada Metode Distribusi Normal

Grafik di atas menjelaskan pengaruh lebih jauh dari variabel *z-score* (*z*) terkait dengan nilai *false positives* dan *false negatives* melalui nilai besaran *information retrieval* (*Precision*, *Recall*, dan *F-Measure*). Penjelasannya adalah sebagai berikut:

- Semakin besar nilai *z*, nilai *Precision* turun tanpa kecenderungan tertentu.
- Semakin besar nilai *z*, nilai *Recall* naik tanpa kecenderungan tertentu, dan kenaikannya lebih besar dibandingkan dengan penurunan nilai *Precision*.
- Meskipun nilai *Recall* memiliki kenaikan yang lebih signifikan daripada penurunan nilai *Precision*, akibat dari ciri-ciri rerata harmonik yang dimiliki oleh *F-Measure*, maka kenaikan nilai *z* akan menurunkan nilai *F-Measure*.

Analisa terkait hasil variasi variabel *z* di atas dapat dijelaskan sebagai berikut: Sesuai rumus (3.4), yakni $x = k(\mu + z\sigma)$, maka semakin besar nilai *z*, maka nilai batas *x* akan semakin besar, sehingga dapat disimpulkan bahwa daerah yang nilai-nilainya terindikasi plagiat akan semakin besar, sehingga jumlah paragraf yang dideteksi sebagai plagiat (nilai *positives*) akan semakin banyak. Analisa di atas diperkuat dengan gambar 4.4 yang menyatakan bahwa kenaikan nilai *z* akan memperluas daerah yang berwarna merah, sehingga akan semakin memperbanyak jumlah paragraf yang terdeteksi sebagai plagiat.

Namun, kenaikan nilai *positives* selain meningkatkan jumlah paragraf yang terindikasi *true positive* (dimana hal ini diinginkan), hal ini juga berdampak pada semakin banyaknya paragraf yang terindikasi *false positive* (tidak diinginkan). Bahkan kenaikan nilai *false positives* lebih tinggi daripada kenaikan jumlah paragraf yang teridinikasi *true positives*, seperti pada grafik di bawah ini.



Grafik 4.6: Pengaruh Variasi Nilai z-score Pada Nilai *Positives* Pada Metode Distribusi Normal

Kesimpulannya, kenaikan nilai *z* berdampak positif pada aspek relevansi (*Recall*) program, namun berdampak negatif pada aspek presisi (*Precision*) dari program. Penurunan nilai *F-Measure* seiring naiknya nilai *z* dianggap tidak terlalu signifikan.

4.3.3. *Analisa Karakteristik Nilai Frobenius Norm, Slice, dan Pad Pada Uji Hasil Deteksi Plagiarisme*

Setiap hasil pengujian menggunakan sistem deteksi plagiarisme dengan metode *latent semantic analysis*, terdapat tiga besaran yang dapat dianalisa dan digunakan lebih jauh untuk melakukan uji hasil deteksi plagiarisme, yakni *frobenius norm* (jarak antara dua vektor), dan besaran sudut antara dua vektor, yang terdiri dari *slice* (berdasarkan dimensi sudut terkecil) dan *pad* (berdasarkan dimensi sudut terbesar). Ketiga besaran tersebut memiliki karakteristik yang berbeda-beda.

Dalam melakukan uji hasil deteksi plagiarisme menggunakan metode distribusi normal, terdapat tujuh besaran yang merupakan kombinasi dari ketiga besaran yanada di atas, yakni:

- Hasil operasi AND dari *frobenius norm*, *slice*, dan *pad*. Ditandai dengan nama kolom *fsp*.
- *Frobenius norm*. Ditandai dengan nama kolom *f*.

- *Slice*. Ditandai dengan nama kolom *s*.
- *Pad*. Ditandai dengan nama kolom *p*.
- Hasil operasi AND dari *frobenius norm* dan *slice*. Ditandai dengan nama kolom *fs*.
- Hasil operasi AND dari *frobenius norm* dan *pad*. Ditandai dengan nama kolom *fp*.
- Hasil operasi AND dari *slice* dan *pad*. Ditandai dengan nama kolom *sp*.

Pengujian untuk menganalisa karakteristik ketujuh besaran di atas dilakukan bersamaan dengan pengujian nilai pengali *z-score* (*k*). Berikut ini adalah prosedur pengujian untuk menganalisa ketujuh besaran di atas:

- Pengujian dilakukan 6 kali, yakni dengan nilai *k* dari 0.8 hingga 1.3 dengan selisih antar pengujian sebesar 0.1.
- Nilai *z-score* yang digunakan adalah -1.

  Tabel di bawah ini adalah hasil pengujiannya.

Param	***f***	***s***	***p***
True Negative	3159.667	3406.667	3318.667
True Positive	171.3333	132.5	134.3333
True Negative + True Positive	3331	3539.167	3453
False Negative	22.66667	61.5	59.66667
False Positive	696.3333	449.3333	537.3333
False Negative + False Positive	719	510.8333	597
Negatives	3182.333	3468.167	3378.333
Positives	867.6667	581.8333	671.6667
Precision	**0.279979**	**0.249086**	**0.222216**
Recall	**0.883162**	**0.68299**	**0.69244**
F-Measure	**0.398894**	**0.356555**	**0.325177**

Tabel 4.5: Rata-Rata Hasil Uji Deteksi Plagiarisme dengan Metode Distribusi Normal Melalui Perspektif *Frobenius Norm*, *Slice*, dan *Pad*

Berdasarkan tabel di atas, terlihat bahwa besaran *frobenius norm* memiliki nilai *positives* yang sangat banyak, yang berimbas pada tingginya juga nilai *false positive*. Namun, tingginya nilai *positives* juga berarti *frobenius norm* mampu mendeteksi indikasi plagiat dengan lebih baik, ditandai dengan jumlah *true positive* yang tertinggi dibandingkan dengan *slice* dan *pad*.

Sedangkan besaran *slice* dan *pad* memiliki karakteristik yang mirip, karena keduanya berasal dari perhitungan sudut antara dua vektor. Keduanya tidak mencatat nilai *positives* yang banyak, yang berarti besaran *slice* dan *pad* sulit untuk mendeteksi indikasi plagiat dengan baik (132.5 untuk *slice* dan 134.5 untuk *pad*, jauh dibandingkan dengan 171.3 untuk *frobenius norm*). Namun, dibandingkan dengan *frobenius norm*, keduanya lebih baik dalam menghindari kejadian *false positives*, dimana angka *false positive* pada *slice* dan *pad* mencapai 449.3 untuk *slice* dan 537.3 untuk *pad*, dibandingkan dengan 69.3 untuk *frobenius norm*.

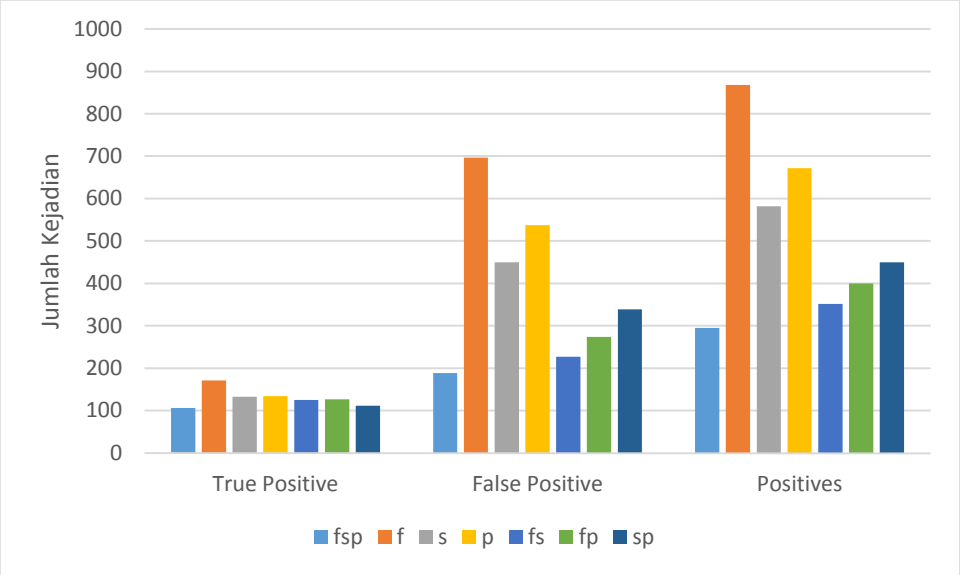
Karena kemampuan besaran *frobenius norm* untuk mendeteksi *true positives* jauh lebih baik dibandingkan dengan kemampuan besaran *slice* dan *pad* untuk menghindari *false positives*, maka nilai besaran performa *information retrieval* pada *frobenius norm* lebih tinggi daripada nilai besaran performa *slice* dan *pad*. Meskipun *frobenius norm* mencatat nilai *false positives* yang tinggi, namun kemampuan *frobenius norm* yang jauh lebih baik dibandingkan besaran sudut dalam mendeteksi nilai yang terindikasi plagiat (*true positives* - dimana *true positives* menjadi bilangan penyebut dalam *Precision* maupun *Recall*) membuat besaran *frobenius norm* mampu memberikan nilai tertinggi pada seluruh nilai besaran performa *information retrieval.*

Tabel di bawah ini adalah hasil pengujian dimana ketiga nilai besaran (*frobenius norm, slice, pad*) dikombinasikan menggunakan operasi AND:

Param	***fsp***	***fs***	***fp***	***sp***
True Negative	3667.333	3629.333	3582.5	3517.333
True Positive	106.5	125	126.3333	111.3333
True Negative + True Positive	3773.833	3754.333	3708.833	3628.667
False Negative	87.5	69	67.66667	82.66667
False Positive	188.6667	226.6667	273.5	338.6667
False Negative + False Positive	276.1667	295.6667	341.1667	421.3333
Negatives	3754.833	3698.333	3650.167	3600
Positives	295.1667	351.6667	399.8333	450
Precision	**0.466563**	**0.458869**	**0.431904**	**0.276952**
Recall	**0.548969**	**0.64433**	**0.651203**	**0.573883**
F-Measure	**0.461201**	**0.498773**	**0.469416**	**0.358126**

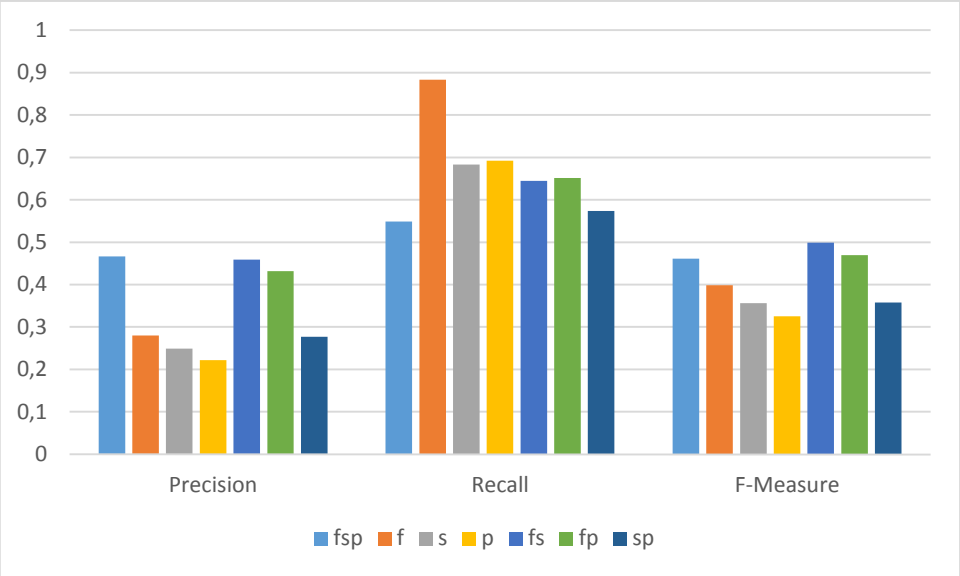
Tabel 4.6: Rata-Rata Hasil Uji Deteksi Plagiarisme dengan Metode Distribusi Normal Melalui Perspektif Hasil Kombinasi Operasi AND dari *Frobenius Norm, Slice, dan Pad*

Sesuai dengan analisa sebelumnya dimana besaran *frobenius norm* merupakan besaran yang terbaik dibandingkan dengan *slice* dan *pad*, hasil kombinasi tanpa melibatkan *frobenius norm* (*slice* AND *pad*) memberikan performa terburuk, sehingga nilai *slice* AND *pad* tidak diikutkan pada analisa selanjutnya.



Grafik 4.7: Nilai *Positives* Hasil Uji Deteksi Plagiarisme dengan Metode Distribusi Normal Pada Besaran *Frobenius Norm*, *Slice*, dan *Pad* beserta Kombinasinya

Kombinasi ketiga besaran (*frobenius norm*, *slice*, dan *pad*) memberikan nilai *positives* terendah. Hal ini disebabkan oleh sifat dari operasi AND, dimana seluruh besaran harus bernilai *positive* untuk menghasilkan nilai *positive*. Namun hal tersebut berarti nilai *false positives* dapat diminimalisir. Hal ini terlihat dari nilai *Precision* yang tertinggi pada kombinasi ketiga besaran tersebut, nilai *positive* yang rendah mengakibatkan rendahnya juga nilai *true positives* yang dimiliki oleh kombinasi ketiga besaran. Sedangkan kombinasi antara dua besaran (*frobenius norm* AND *slice* atau *frobenius norm* AND *pad*) mampu meningkatkan nilai *Precision* secara signifikan. Hal ini mengindikasikan bahwa banyak nilai *positive* pada *frobenius norm* dan nilai *positive* pada nilai sudut tidak sesuai satu sama lain, yang menyebabkan operasi AND membuat nilai-nilai tersebut hilang dan menjadi *negatives*, dan besar kemungkinan nilai-nilai *positive* yang hilang tersebut adalah *false positives*.



Grafik 4.8: Nilai *Information Retrieval* Hasil Uji Deteksi Plagiarisme dengan Metode Distribusi Normal Pada Besaran *Frobenius Norm*, *Slice*, dan *Pad* Beserta Kombinasinya

Nilai *F-Measure* yang didapatkan oleh kombinasi *frobenius norm* dengan *slice* dan *pad* mencapai nilai-nilai tertinggi, karena nilai *Recall* yang tinggi oleh *frobenius norm* diseimbangkan dengan operasi AND yang terjadi sehingga menghilangkan banyak kejadian *false positives*, yang pada akhirnya meningkakan nilai *Precision*, sehingga nilai *F-Measure* yang didapatkan menjadi paling optimal.

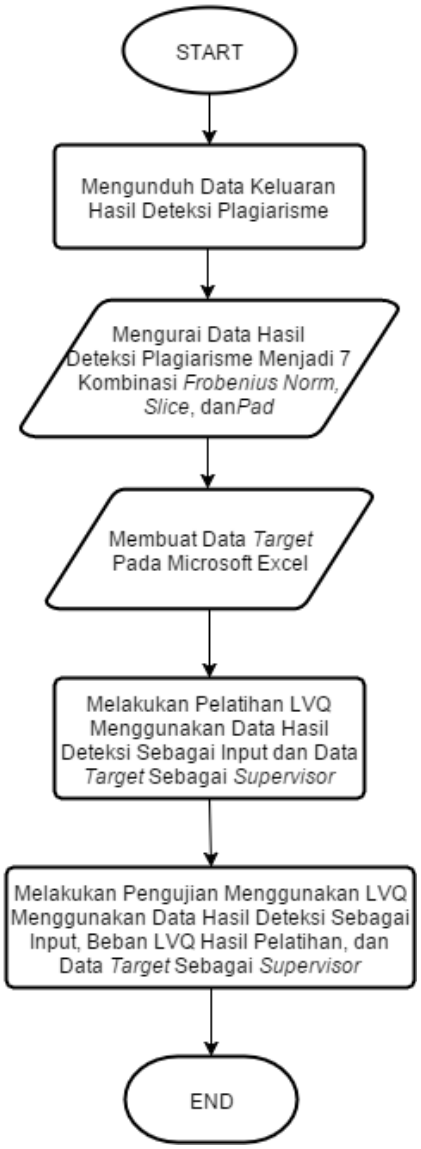
Namun, untuk dapat dikombinasikan dengan hasil keluaran lain (*learning vector quantization*), seperti yang telah dibahas sebelumnya, dibutuhkan nilai *Recall* yang tinggi agar operasi AND tidak memotong banyak nilai *true positives*, sehingga hasil dari *frobenius norm* sendiri tanpa dikombinasikan dengan yang lainnya dianggap cukup baik unutk dikombinasikan dengan hasil uji deteksi plagiarisme oleh *learning vector quantization*.

Kesimpulannya, dari ketujuh besaran di atas, hanya besaran-besaran yang melibatkan *frobenius norm* sebagai komponennya yang dapat digunakan untuk melakukan uji deteksi plagiarisme. Hal ini disebabkan oleh kemampuan *frobenius norm* yang sangat tinggi dalam aspek relevansi (*Recall*), dan kombinasinya dengan besaran-besaran lain dapat digunakan untuk meningkatkan *Precision*.

4.4. Pengujian Hasil Deteksi Plagiarisme Menggunakan Learning Vector Quantization

Seperti uji deteksi plagiarisme menggunakan distrbusi normal, uji deteksi plagiarisme menggunakan *learning vector quantization* juga memanfaatkan hasil program deteksi plagiarisme yang keluarannya berupa besaran pada *frobenius norm, slice,* dan *pad.* Pengerjaan uji deteksi plagiarisme menggunakan *learning vector quantization* dilakukan menggunakan perangkat lunak MATLAB®, dimana program *learning vector quantization* dibangun didalamnya.

Diagram proses pengujian deteksi plagiarisme menggunakan *learning vector quantization* adalah sebagai berikut:



Gambar 4.5: Diargam Proses Uji Deteksi Plagiarisme Menggunakan *Learning Vector Quantization*

Karena proses pengujian menggunakan MATLAB®, penulis perlu terlebih dahulu mengunduh hasil deteksi plagiarisme dalam bentuk yang bisa dibaca oleh MATLAB®, misalnya program excel (.xlsx, .xls, .csv, dan sebagainya). Kemudian, karena pengujian melibatkan besaran *frobenius norm, slice, pad,* dan kombinasinya, maka hasil deteksi plagiarisme dibagi menjadi 7 data yang merepresentasikan masing-masing besaran dan kombinasinya tersebut. Ketujuh data ini akan menjadi masukan bagi program *learning vector quantization*. Hasil pengujian dari program ini harus sesuai dengan hasil pengecekan manual oleh penulis, dimana hasil pengecekan ini juga dimasukkan pada program yang bisa dibaca oleh MATLAB®, dan dijadikan sebagai data *supervisor* untuk program *learning vector quantization*.

Setelah data masukan dan data *supervisor* telah dipersiapkan, maka program *learning vector quantization* dapat dijalankan pada fase pelatihan. Setelah fase pelatihan menghasilkan bobot yang diinginkan sesuai dengan data masukan yang dipilih penulis, maka program dapat melakukan pengujian. Hasil pengujian inilah yang dijadikan data keluaran untuk dianalisa.

4.4.1. *Analisa Variasi Nilai Alpha dan n*

Sesuai dengan dasar teori bahwa nilai *alpha* dan *n* memiliki fungsi yang sama, yaitu mempercepat proses pembelajaran, maka variasi dari kedua variabel ini dilakukan secara bersama-sama. Dengan menggunakan nilai *default* 1 untuk *alpha* dan 5 untuk *n*, variasi dilakukan dengan mengalikan bilangan tersebut dengan nilai 10^{-1}, 10^{-2}, dan 10^{-3}. Dengan demikian, ketiga variasi yang ada yakni: nilai *alpha* 0.1 dan *n* 0.5, *alpha* 0.01 dan *n* 0.05, dan *alpha* 0.001 dan *n* 0.005.

Selain variasi-variasi yang ada, parameter-parameter lain yang perlu ditentukan dalam menjalankan *learning vector quantization* adalah sebagai berikut:

- Jumlah *input neuron*: sesuai dengan data yang dimasukkan (misalnya data yang dimasukkan adalah seluruh tiga besaran keluaran program deteksi plagiarisme - *frobenius norm, slice,* dan *pad,* maka akan terdapat 3 *input neuron*).

- Jumlah *output neuron*: 2 (plagiat atau tidak). Jumlah *output neuron* ini ditandai dengan variasi pada data *supervisor*. Data *supervisor* akan berisi 1 kolom dengan nilai 1 (untuk plagiat) atau 2 (tidak plagiat).
- Rasio (untuk pelatihan): 1. Artinya 1 x 100% dari seluruh data masukan akan digunakan untuk pelatihan.

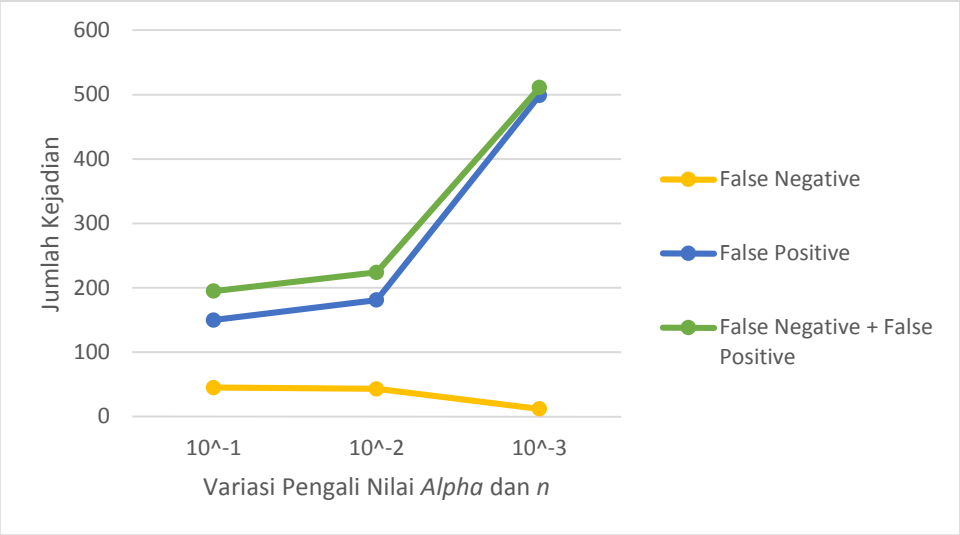
Tabel di bawah ini adalah hasil dari pengujian sesuai dengan kriteria di atas:

Pengali	**10^{-1}**	**10^{-2}**	**10^{-3}**
True Negative	3706	3675	3357
True Positive	149	151	182
True Negative + True Positive	3855	3826	3539
False Negative	45	43	12
False Positive	150	181	499
False Negative + False Positive	195	224	511
Negatives	3751	3718	3369
Positives	299	332	681
Precision	**0.498328**	**0.454819**	**0.267254**
Recall	**0.768041**	**0.778351**	**0.938144**
F-Measure	**0.604462**	**0.574144**	**0.416**

Tabel 4.7: Hasil Variasi Nilai *Alpha* dan *n* Pada Pengujian Hasil Deteksi Plagiarisme dengan Algoritma *Learning Vector Quantization*

Berdasarkan tabel di atas, terlihat bahwa semakin kecil nilai *alpha* dan *n*, maka nilai *true negative* + *true positive* akan semakin kecil. Namun, seiring dengan mengecilnya nilai *alpha* dan *n*, nilai *positive* semakin besar sedangkan nilai *negative* menjadi semakin kecil. Nilai *true* yang semakin kecil mengesankan bahwa program semakin tidak akurat. Namun, berikut akan dibahas beberapa parameter lain terkait hasil pengujian.

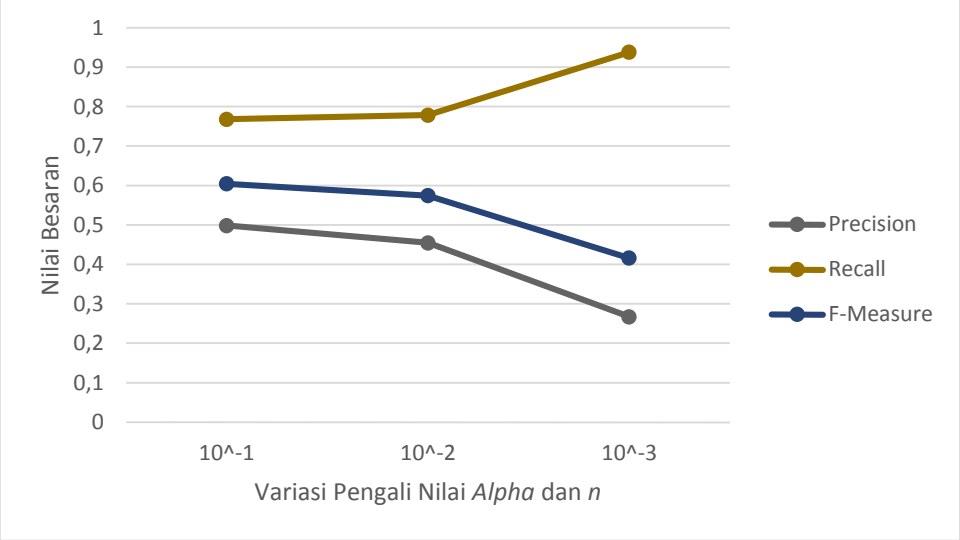
Representasi grafis dari beberapa besran pada tabel di atas dapat dilihat pada kedua grafik dibawah ini:



Grafik 4.9: Pengaruh Variasi Nilai alpha dan n Pada Nilai False Positives dan False Negatives Pada Algoritma Learning Vector Quantization

Dari grafik di atas, pengaruh variabel *alpha* dan *n* sebagai variabel untuk mempercepat proses pembelajaran adalah sebagai berikut:

- Semakin kecil nilai *alpha* dan *n*, nilai *false negatives* turun.
- Semakin kecil nilai *alpha* dan *n*, nilai *false positives* naik, dan kenaikannya cenderung semakin besar bahkan tajam seiring turunnya nilai *alpha* dan *n.*
- Sehingga, semakin kecil nilai *alpha* dan *n*, akan semakin banyak hasil pengujian yang salah (*false*), karena kenaikan *false positives* jauh lebih tinggi daripada penurunan *false negatives*.



Grafik 4.10: Pengaruh Variasi Nilai *Alpha* dan *n* Pada Besaran Pengukuran *Performa Information Retrieval* Pada Algoritma *Learning Vector Quantization*

Grafik di atas menjelaskan pengaruh lebih jauh dari variabel pengali *alpha* dan *n* terkait dengan nilai *false positives* dan *false negatives* melalui nilai besaran *information retrieval* (*Precision*, *Recall*, dan *F-Measure*). Penjelasannya adalah sebagai berikut:

- Semakin kecil nilai *alpha* dan *n*, nilai *Precision* turun, dengan sedikit kecenderungan penurunannya semakin besar seiring naiknya turunnya nilai *alpha* dan *n.*
- Semakin kecil nilai *alpha* dan *n*, nilai *Recall* naik, dan kenaikannya cenderung semakin besar seiring naiknya nilai *alpha* dan *n.*
- Meskipun kecenderungan kenaikan *Recall* sama besarnya dengan kecenderungan turunnya nilai *Precision*, akibat dari ciri-ciri rerata harmonik yang dimiliki oleh *F-Measure*, maka penurunan nilai *alpha* dan *n* akan menurunkan nilai *F-Measure*.

Analisa terkait hasil variasi variabel *alpha* dan *n* di atas dapat dijelaskan sebagai berikut: Nilai *alpha* dan *n* yang besar menyebabkan proses pembelajaran akan semakin banyak (sehingga semakin lama). Proses pembelajaran yang semakin banyak menyebabkan nilai bobot akan semakin sering terkoreksi. Karena data yang digunakan untuk pelatihan adalah data sebenarnya, dengan jumlah *negatives* yang jauh lebih banyak daripada jumlah *positives* (194 untuk *positives* dan 3856 untuk *false negatives*, perbandingannya yakni 1 : 19.876), menyebabkan algoritma lebih banyak "belajar untuk mengenal hasil *negative* (tidak terindikasi plagiat) daripada hasil *positive* (terindikasi plagiat)". Sebagai perbandingan, dengan *alpha* dan *n* masing-masing sebesar 0.1 dan 0.5, pengulangan pelatihan akan terjadi sebanyak 24 kali, sedangkan untuk *alpha* dan *n* masing-masing sebesar 0.01 dan 0.05, pelatihan hanya terjadi sebanyak 5 kali, dan untuk *alpha* dan *n* masing-masing sebesar 0.001 dan 0.005, pelatihan hanya terjadi sebanyak 3 kali. Selain itu, nilai *alpha* dan *n* yang kecil menyebabkan nilai bobot tidak banyak terkoreksi setiap kali pelatihan, sehingga nilai *alpha* dan *n* yang kecil tidak dapat merubah sifat bobot yang awalnya hanya entri pertama dan kedua dari data masukan, dimana hal tersebut tidak akan cukup untuk mempelajari data yang seharusnya tidak diindikasi sebagai plagiat (*negative*).

Banyaknya hasil *negative* pada nilai *alpha* dan *n* 0.1 dan 0.5 menyebabkan tidak banyak kejadian *false positives*, sehingga nilai *Recall* yang didapatkan cukup tinggi. Meskipun lebih banyak "mendapatkan pelatihan tentang data yang seharusnya tidak terindikasi plagiat", secara umum algoritma tetap dapat mendeteksi entri-entri yang terindikasi plagiat (*true positives*).

Kesimpulannya, penurunan nilai *alpha* dan *n* berdampak positif pada aspek relevansi (*Recall*) program, namun berdampak negatif pada aspek presisi (*Precision*) dari program. Karena dampak negatif pada *Precision* lebih besar daripada dampak positif pada *Recall*, secara keseluruhan, penurunan nilai *alpha* dan *n* berdampak negatif pada keakuratan program pengujian sistem deteksi plagiarisme berdasarkan metode distribusi normal.

4.4.2. *Analisa Karakteristik Nilai Frobenius Norm, Slice, dan Pad Pada Uji Hasil Deteksi Plagiarisme*

Seperti uji hasil deteksi plagiarisme dengan metode distribusi normal, terdapat uji hasil deteksi plagiarisme juga melibatkan besaran *frobenius norm*, *slice*, dan *pad*, karena ketiga besaran ini adalah keluaran dari sistem deteksi plagiarisme berbasis *latent semantic analysis*. Terdapat pula tujuh besaran yang sama, yang merupakan kombinasi dari tiga besaran di atas, yakni:

- Hasil operasi AND dari *frobenius norm*, *slice*, dan *pad*. Ditandai dengan nama kolom *fsp*.
- *Frobenius norm*. Ditandai dengan nama kolom *f*.
- *Slice*. Ditandai dengan nama kolom *s*.
- *Pad*. Ditandai dengan nama kolom *p*.
- Hasil operasi AND dari *frobenius norm* dan *slice*. Ditandai dengan nama kolom *fs*.
- Hasil operasi AND dari *frobenius norm* dan *pad*. Ditandai dengan nama kolom *fp*.
- Hasil operasi AND dari *slice* dan *pad*. Ditandai dengan nama kolom *sp*.

Pengujian untuk menganalisa karakteristik ketujuh besaran di tas dilakukan bersamaan dengan pgnujian nilai *alpha* dan *n.* Berikut ini adalah prosedur pengujian untuk menganalisa ketujuh besaran di atas.

- Jumlah *input neuron*: sesuai dengan data yang dimasukkan (misalnya data yang dimasukkan adalah seluruh tiga besaran keluaran program deteksi plagiarisme - *frobenius norm, slice,* dan *pad,* maka akan terdapat 3 *input neuron*).
- Jumlah *output neuron*: 2 (plagiat atau tidak). Jumlah *output neuron* ini ditandai dengan variasi pada data *supervisor*. Data *supervisor* akan berisi 1 kolom dengan nilai 1 (untuk plagiat) atau 2 (tidak plagiat).
- Rasio (untuk pelatihan): 1. Artinya 1 x 100% dari seluruh data masukan akan digunakan untuk pelatihan.

Tabel di bawah ini adalah hasil pengujiannya:

Param	***f***	***s***	***p***
True Negative	3630.667	3373.667	3462.667
True Positive	127	64.33333	63.33333
True Negative + True Positive	3757.667	3438	3526
False Negative	67	129.6667	130.6667
False Positive	225.3333	482.3333	393.3333
False Negative + False Positive	292.3333	612	524
Negatives	3697.667	3503.333	3593.333
Positives	352.3333	546.6667	456.6667
Precision	0.381253	0.280162	0.158213
Recall	0.654639	0.331615	0.32646
F-Measure	0.465479	0.123266	0.142965

Tabel 4.8 Rata-Rata Hasil Uji Deteksi Plagiarisme dengan Algoritma *Learning Vector Quantization* Melalui Perspektif *Frobenius Norm, Slice,* dan *Pad*

Melalui hasil pengujian di atas, terlihat bahwa karakteristik *frobenius norm, slice,* dan *pad* pada pengujian menggunakan algoritma *learning vector quantization* tidak sama dengan karakteristik pada pengujian menggunakan metode distribusi normal.

Besaran *frobenius norm* memiliki nilai *positives* yang paling sedikit, sedangkan besaran *slice* dan *pad* mencatat nilai *positives* yang sangat banyak.

Namun, banyaknya nilai *positives* yang dimiliki oleh *slice* dan *pad* ternyata sangat tidak akurat. Hal ini terlihat pada jumlah deteksi tepat (*true positives*) yang diperoleh besaran *slice* dan *pad* jauh lebih sedikit dibandingkan dengan *frobenius norm* (hanya sekitar setengahnya).

Namun, jika melihat pada hasil pengujian per variasi nilai *alpha* dan *n*, terlihat karakteristik yang sebelumnya telah dibahas pada metode distribusi normal, dimana besaran *frobenius norm* memiliki nilai *positives* yang sangat banyak, yang berimbas pada tingginya juga nilai *false positive*. Bahkan, untuk nilai *alpha* dan *n* sebesar 0.1 dan 0.5, besaran *slice* dan *pad* sama sekali tidak mampu mencatat nilai *positives*. Pada nilai *alpha* dan *n* sebesar 0.01 dan 0.05, besaran *slice* dan *pad* hanya mampu mencatat nilai *positives* sebanyak 9 dan 28. Sehingga karakteristik yang dijelaskan pada paragraf sebelumnya, seluruhnya dipengaruhi oleh hasil penerapan pada variasi nilai *alpha* dan *n* sebesar 0.001 dan 0.005.

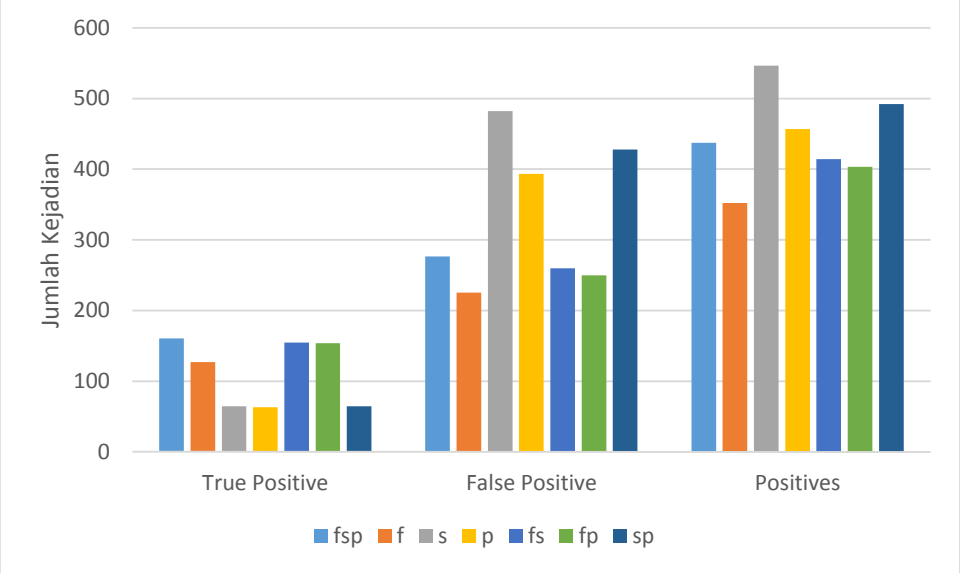
Namun, secara umum dapat disimpulkan bahwa dengan kemampuan *frobenius norm* yang jauh lebih baik untuk mendeteksi *true positives*, nilai besaran performa *information retrieval* pada *frobenius norm* lebih tinggi daripada nilai besaran performa pada *slice* dan *pad*.

Tabel di bawah ini adalah hasil pengujian dimana ketiga nilai besaran (*frobenius norm*, *slice*, *pad*) dikombinasikan menggunakan operasi AND:

Param	***fsp***	***fs***	***fp***	***sp***
True Negative	3579.333	3596.333	3606.333	3428.333
True Positive	160.6667	154.6667	153.6667	64.66667
True Negative + True Positive	3740	3751	3760	3493
False Negative	33.33333	39.33333	40.33333	129.3333
False Positive	276.6667	259.6667	249.6667	427.6667
False Negative + False Positive	310	299	290	557
Negatives	3612.667	3635.667	3646.667	3557.667
Positives	437.3333	414.3333	403.3333	492.3333
Precision	0.4068	0.406962	0.408638	0.172015
Recall	0.828179	0.797251	0.792096	0.333333
F-Measure	0.531536	0.526073	0.528116	0.141452

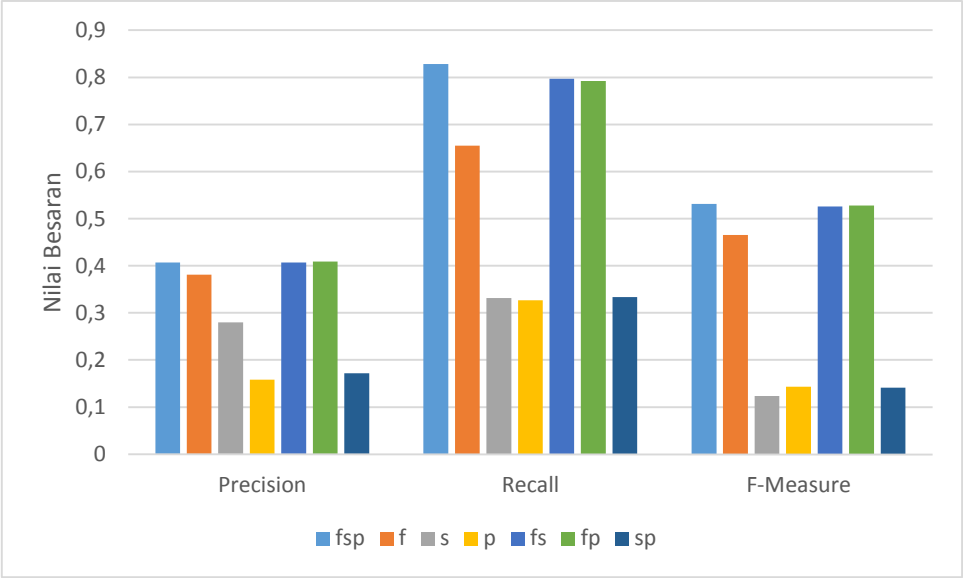
Tabel 4.9: Rata-Rata Hasil Uji Deteksi Plagiarisme dengan Algoritma *Learning Vector Quantization* Melalui Perspektif Hasil Kombinasi Operasi AND dari *Frobenius Norm*, *Slice*, dan *Pad*

Sesuai dengan analisa-analisa sebelumnya, baik terkait besaran *frobenius norm* maupun dengan analisa pada metode distribusi normal, besaran *frobenius norm* merupka besaran yang terbaik dibandingkan dengan *slice* dan *pad*, hasil kombinasi tanpa melibatkan *frobenius norm* (*slice* dan *pad*) memberikan performa terburuk, sehingga nilai *slice* dan *pad* tidak diikutkan pada analisa selanjutnya.



Grafik 4.11: Nilai Positives Hasil Uji Deteksi Plagiarisme dengan Algoritma *Learning Vector Quantization* Pada Besaran *Frobenius Norm*, *Slice*, dan *Pad* beserta Kombinasinya

Tidak seperti metode distribusi normal dimana kombinasi besaran-besaran (*frobenius norm*, *slice*, dan *pad*) memberikan nilai *positives* yang rendah akibat dari sifat operasi AND, sehingga nilai *positives* tidak akan tergantung dengan jumlah *positives* terendah. Kombinasi dari besaran-besaran tersebut akan memberikan nilai *positives* sesuai dengan karakteristik dari kombinasi besaran-besaran tersebut, bukan dari karakteristik besaran-besaran tersebut secara individu yang kemudian di AND kan dengan besaran lain. Seperti gabungan dari *frobenius norm*, *slice*, dan *pad* tidak memberikan nilai *positives* yang terendah (keempat terbanyak), bahkan justru memberikan nilai *true positive* tertinggi.



Grafik 4.12: Nilai *Information Retrieval* Hasil Uji Deteksi Plagiarisme dengan Algoritma *Learning Vector Quantization* Pada Besaran *Frobenius Norm, Slice, dan Pad* Beserta Kombinasinya

Nilai *F-Measure* yang didapatkan oleh kombinasi *frobenius norm* dengan *slice* dan *pad* mencapai nilai-nilai tertinggi. Hal ini karena nilai *frobenius norm* yang berdiri sendiri tidak menghasilkan banyak nilai *positives*, yang berimbas pada terlalu sedikitnya nilai *true positive* yag dihasilkan oleh *frobenius norm*, yang pada akhirnya membuat *Recall* dan *Precision* yang dihasilkan menjadi agak rendah.

Kesimpulannya, dari ketujuh besaran di atas, hanya besaran-besaran yang melibatkan *frobenius norm* sebagai komponennya yang dapat digunakan untuk melakukan uji deteksi plagiarisme. Hal ini disebabkan oleh sifat *slice* dan *pad*, yang, tanpa *frobenius norm*, sama sekali tidak mampu untuk melakukan pengujian hasil deteksi plagiarisme dengan baik (antara tidak mampu mencatat *positives* atau terlalu banyak *false positives*).

4.5. Pengujian Hasil Deteksi Plagiarisme Menggunakan Operasi AND Antara Hasil Distribusi Normal dan Learning Vector Quantization

Untuk melakukan operasi AND antara hasil distribusi normal dan *learning vector quantization*, penulis terlebih dahulu mengambil hasil-hasil terbaik dari masing-masing metode uji untuk menghemat tenaga dan waktu, dan untuk memastikan bahwa hasil operasi AND memberikan hasil-hasil yan terbaik.

Dalam mengambil hasil-hasil terbaik dari masing-masing metode uji, penulis mendasarkan kriterianya pada besaran *information retrieval*. Sebagai syarat utama, [enulis mensyaratkan nilai *Recall* yang tinggi, karena sebuah program deteksi plagiarisme harus mampu mendeteksi dengan tingkat presisi yang tinggi (program harus dapat menekan jumlah *false negatives*). Selain itu, nilai *F-Measure* juga menjadi persyaratan lainnya, agar selain memiliki tingkat presisi yang tinggi, program tidak terlalu mencatat terlalu banyak *false positives* yang berpotensi menghasilkan *false accusation*.

Berikut ini adalah hasil-hasil terbaik yang dihimpun oleh penulis pada metode distribusi normal:

k	**0.8**	**0.8**	**0.8**	**0.8**	**0.9**
z	**-0.8**	**-0.8**	**-1**	**-1.1**	**-0.8**
tipe	***f***	***fs***	***f***	***f***	***f***
Precision	0.447802	0.643564	0.510135	0.554717	0.350806
Recall	0.840206	0.670103	0.778351	0.757732	0.896907
F-Measure	0.584229	0.656566	0.616327	0.640523	0.504348
k	**0.9**	**0.9**	**0.9**	**0.9**	**1**
z	**-0.8**	**-0.8**	**-1**	**-1.1**	**-0.8**
tipe	***fs***	***fp***	***f***	***f***	***f***
Precision	0.52381	0.552	0.415423	0.623596	0.232283
Recall	0.737113	0.71134	0.860825	0.572165	0.912371
F-Measure	0.61242	0.621622	0.560403	0.596774	0.370293
k	**1**	**1**	**1.1**	**1.1**	**1.1**
z	**-1**	**-1.1**	**-0.8**	**-0.8**	**-0.8**
tipe	***f***	***f***	***f***	***fs***	***fp***
Precision	0.31216	0.345528	0.153978	0.258842	0.24961
Recall	0.886598	0.876289	0.927835	0.829897	0.824742
F-Measure	0.461745	0.495627	0.264123	0.394608	0.383234
k	**1.1**	**1.1**	**1.1**	**1.2**	**1.3**
z	**-0.9**	**-1**	**-1.1**	**-1**	**-1**
tipe	***f***	***f***	***f***	***f***	***f***
Precision	0.177052	0.202532	0.2341	0.137091	0.102535
Recall	0.92268	0.907216	0.891753	0.927835	0.938144
F-Measure	0.297095	0.331138	0.370847	0.238885	0.184865

Tabel 4.10: Daftar Hasil Uji Terbaik dengan Metode Distribusi Normal

Terdapat 20 hasil pada metode distribusi normal yang berpotensi menghasilkan nilai yang baik ketika dilakukan operasi AND dengan hasil pengujian

dengan algoritma *learning vector quantization.* Angka-angka yang dinyatakan dengan huruf *k* dan *z* adalah parameter-parameter yang divariasikan pada uji menggunakan distribusi normal, sesuai dengan rumus (3.4): $x = k(\mu + z\sigma)$. Sedangkan baris yang dinyatakan dengan "tipe" adalah jenis besaran keluaran program deteksi plagiarisme yang diuji. Tipe *f* berarti keluaran *frobenius norm* yang diuji. Sebagai contoh, entri dengan *k* = 0.9, *z* = -0.8, dan tipe = *fs* berarti menandakan hasil uji deteksi plagiarisme menggunakan metode distribusi normal dengan batas nilai $x = 0.9(\mu - 0.8\sigma)$, dan pengujian memanfaatkan operasi AND dari keluaran *frobenius norm* dan *slice*.

Berikut ini adalah hasil-hasil terbaik yang dihimpun oleh penulis pada algoritma *learning vector quantization*:

param	**10^{-1}**	**10^{-1}**	**10^{-1}**	**10^{-2}**	**10^{-2}**
tipe	***fsp***	***fs***	***fp***	***fsp***	***fs***
Precision	0.498328	0.491228	0.489362	0.454819	0.449848
Recall	0.768041	0.721649	0.71134	0.778351	0.762887
F-Measure	0.604462	0.584551	0.579832	0.574144	0.565966
param	**10^{-2}**	**10^{-3}**	**10^{-3}**	**10^{-3}**	**10^{-3}**
tipe	***fp***	***fsp***	***f***	***fs***	***fp***
Precision	0.442136	0.267254	0.307263	0.279809	0.294416
Recall	0.768041	0.938144	0.850515	0.907216	0.896907
F-Measure	0.561205	0.416	0.451436	0.427704	0.443312

Tabel 4.101: Daftar Hasil Uji Terbaik dengan Algoritma *Learning Vector Quantization*

Terdapat 10 hasil pada algoritma *learning vector quantization* yang berpotensi menghasilkan nilai yang baik ketika dilakukan operasi AND dengan hasil pengujian dengan metode distribusi normal. Angka-angka yang dinyatakan dengan "param" adalah koefisien pengali nilai *alpha* dan *n.* Seperti telah dibahas sebelumnya, variasi nilai *alpha* dan *n* dilakukan dengan memvariasikan nilai pengalinya. Dengan nilai *default* 1 dan 5 untuk *alpha* dan *n*, nilai param sebesar 0.1 berarti *alpha* = 0.1 dan *n* = 0.5. Nilai param sebesar 0.01 berarti *alpha* = 0.01 dan *n* = 0.05. Nilai param sebesar 0.001 berarti *alpha* = 0.001 dan *n* = 0.005. Baris yang dinyatakan dengan "tipe" adalah jenis besaran keluaran program deteksi plagiarisme yang diuji. Tipe *f* berarti keluaran *frobenius norm* yang diuji. Sebagai contoh, entri dengan param = 10^{-3}, dan tipe = *fs* berarti menandakan hasil uji deteksi

plagiarisme menggunakan algoritma *learning vector quantization* dengan parameter percepatan belajar *alpha* = 0.001 ($1x10^{-3}$) dan *n* = 0.005 ($5x10^{-3}$), serta *input neuron* menerima masukan dari dua besaran, yakni *frobenius norm* dan *slice*.

Seperti yang telah dibahas sebelumnya, seluruh hasil terbaik, baik dari metode distribusi normal maupun algoritma *learning vector quantization*, selalu melibatkan besaran *frobenius norm*, karena keberadaan besaran *frobenius norm* mampu untuk meningkatkan nilai besaran *information retrieval*.

Dari 20 hasil terbaik pada metode distribusi normal dan 10 hasil terbaik pada algoritma *learning vector quantization*, dilakukan operasi AND antara masing-masing hasil metode distribusi normal dengan masing-masing hasil metode *learning vector quantization*, sehingga didapatkan sejumlah 20 x 10 = 200 hasil pengujian. Dari 200 hasil pengujian tersebut, berikut adalah beberapa hasil terbaik:

DISTRIBUSI NORMAL	***k***	**0.8**		**0.8**		
	z	**0.8**		**0.8**		
	tipe	***f***		***fs***		
LEARNING VECTOR QUANTIZATION	***param***	**10^-1**	**10^-3**	**10^-3**	**10^-3**	**10^-3**
	tipe	***fsp***	***fsp***	***fsp***	***fs***	***fp***
True Negative		3755	3688	3790	3790	3791
True Positive		136	161	128	124	122
True Negative + True Positive		3891	3849	3918	3914	3913
False Negative		58	33	66	70	72
False Positive		101	168	66	66	65
False Negative + False Positive		159	201	132	136	137
Precision		0.5738	0.4894	0.6598	0.6526	0.6524
Recall		0.7010	0.8299	0.6598	0.6392	0.6289
F-Measure		0.6310	0.6157	0.6598	0.6458	0.6404
DISTRIBUSI NORMAL	***k***	**0.8**		**0.8**		
	z	**1**		**1.1**		
	tipe	***f***		***f***		
LEARNING VECTOR QUANTIZATION	***param***	**10^-1**	**10^-3**	**10^-1**	**10^-3**	**10^-3**
	tipe	***fsp***	***fsp***	***fsp***	***fsp***	***fs***
True Negative		3776	3729	3786	3750	3747
True Positive		129	150	125	146	143
True Negative + True Positive		3905	3879	3911	3896	3890
False Negative		65	44	69	48	51
False Positive		80	127	70	106	109
False Negative + False Positive		145	171	139	154	160

Precision		0.6172	0.5415	0.6410	0.5794	0.5675
Recall		0.6649	0.7732	0.6443	0.7526	0.7371
F-Measure		0.6401	0.6369	0.6427	0.6547	0.6413
DISTRIBUSI NORMAL	***k***	**0.9**	**0.9**	**1.1**		
	z	**0.8**	**1**	**0.8**		
	tipe	***fp***	***f***	***fs***		
LEARNING VECTOR QUANTIZATION	***param***	**10^-3**	**10^-1**	**10^-1**	**10^-2**	
	tipe	***fsp***	***fsp***	***fsp***	***fp***	
True Negative		3768	3758	3786	3784	
True Positive		133	136	127	126	
True Negative + True Positive		3901	3894	3913	3910	
False Negative		61	58	67	68	
False Positive		88	98	70	72	
False Negative + False Positive		149	156	137	140	
Precision		0.6018	0.5812	0.6447	0.6364	
Recall		0.6856	0.7010	0.6546	0.6495	
F-Measure		0.6409	0.6355	0.6496	0.6429	

Tabel 4.12: Daftar Hasil Terbaik Operasi AND Antara Hasil Uji Metode Distribusi Normal dan Hasil Uji Algoritma *Learning Vector Quantization*

Kriteria pemilihan hasil-hasil terbaik di atas adalah memiliki *F-Measure* di atas 0.63, atau *F-Measure* di atas 0.6 dengan nilai *recall* yang sangat tinggi.

Dari hasil-hasil di atas, dapat disimpulkan bahwa uji terbaik dapat dilakukan dengan mempertimbangkan keseimbangan pada kemampuan sebuah uji dalam sisi presisi (*precision*) dan relevansi (*recall*). Hal ini dapat dicontohkan pada hasil operasi AND antara metode distribusi normal (k = 0.8, z = 0.8, tipe = *frobenius norm*) yang awalnya memiliki *recall* yang sangat tinggi (0.844) dengan *precision* yang tidak terlalu rendah (0.447), dengan hasil algoritma *learning vector quantization* (koefisien pengali = 0.1, tipe = *frobenius norm, slice, pad*) dengan *recall* sebesar 0.768 dan *precision* sebesar 0.498, dapat menghasilkan sebuah hasil uji dengan *precision* yang meningkat (0.5738) tanpa mengurangi *recall* terlalu banyak (0.710), dengan nilai *F-Measure* sebesar 0.63109.

Tidak diperhatikannya keseimbangan antara *precision* dan *recall* menghasilkan nilai yang tidak terlalu baik. Misalnya hasil operasi AND antara metode distribusi normal (k = 0.9, z = 0.8, tipe = *frobenius norm*) yang awalnya memiliki *recall* yang sangat tinggi (0.896) dengan *precision* yang rendah (0.35),

dengan hasil algoritma *learning vector quantization* (koefisien pengali = 0.3, tipe = *frobenius norm, pad*) dengan *recall* yang sangat tinggi pula (0.90) dan *precision* yang juga rendah (0.279), menghasilkan sebuah hasil uji dengan precision yang tetap rendah (0.426) meskipun mendapatkan nilai *recall* yang sangat tinggi (0.829), sehingga nilai *F-Measure* nya tidak sampai 0.6 (yakni 0.5629).

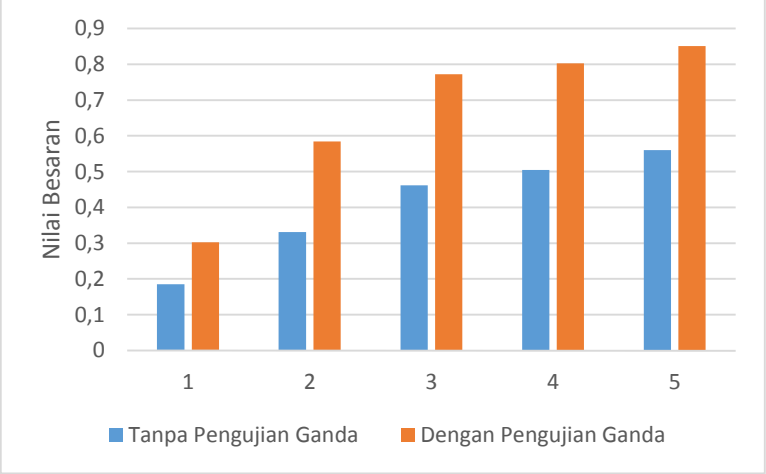
4.6. Analisa Kekurangan dan Perancangan Optimalisasi Sistem

Dari berbagai metode pengujian yang dilakukan, tidak ada metode yang mencapai tingkat akurasi (*F-Measure*) sebesar 70%. Salah satu hal kekurangan yang paling besar dalam sistem ini adalah kurangnya kamus translasi yang baik, sehingga, sebuah paragraf yang seharusnya merupakan plagiat dari paragraf lainnya, menjadi terkesan berbeda setelah proses translasi dijalankan. Untuk itu, diperlukan *database* kamus yang lebih akurat, misalnya menggunakan lisensi atau *Application Programming Interface* (API) berbayar, yang mampu melakukan proses translasi lebih baik.

Pengembangan sistem juga dapat dilakukan dengan melakukan pengujian ganda. Pengujian ganda yaitu menguji paragraf uji terhadap paragraf referensi, kemudian melakukan pengujian dengan menempatkan paragraf referensi sebagai paragraf uji, dan paragraf uji menjadi paragraf referensi, dan menentukan indikasi plagiat jika dan hanya jika kedua hasil pengujian terindikasi plagiat. Hal ini terbukti mampu meningkatkan tingkat akurasi pengujian secara signifikan. Berikut ini adalah contoh hasil pengujian beberapa paragraf dengan metode pengujian ganda:

	k	1.3	1.1	1	0.9	0.9
	z	1	1	1	0.8	1
	Param	f	f	f	f	f
SEBELUM	**True Negative**	2263	3163	3477	3534	3621
	True Positive	182	176	172	174	167
	True Negative + True Positive	2445	3339	3649	3708	3788
	False Negative	1593	693	379	322	235
	False Positive	12	18	22	20	27
	False Negative + False Positive	1605	711	401	342	262
	Negatives	3856	3856	3856	3856	3856
	Positives	194	194	194	194	194
	Precision	0.102535	0.202532	0.31216	0.350806	0.415423
	Recall	0.938144	0.907216	0.886598	0.896907	0.860825
	F-Measure	0.184865	0.331138	0.461745	0.504348	0.560403
SESUDAH	**True Negative**	3032	3626	3778	3792	3826
	True Positive	181	175	171	173	166
	True Negative + True Positive	3213	3801	3949	3965	3992
	False Negative	13	19	23	21	28
	False Positive	824	230	78	64	30
	False Negative + False Positive	837	249	101	85	58
	Negatives	3045	3645	3801	3813	3854
	Positives	1005	405	249	237	196
	Precision	0.1801	0.432099	0.686747	0.729958	0.846939
	Recall	0.93299	0.902062	0.881443	0.891753	0.85567
	F-Measure	0.301918	0.584307	0.772009	0.802784	0.851282

Tabel 4.11: Hasil Pengujian Ganda



Grafik 4.13: Grafik Perbandingan *F-Measure* Sebelum dan Setelah Pengujian Ganda

Dari grafik dan tabel di atas, pengujian ganda mampu meningkatkan akurasi pengujian secara signifikan. Bahkan ada pengujian yang mampu mencapai tingkat akurasi sebesar 85%. Pengujian ganda mampu mempertahankan nilai *recall* sambil meningkatkan nilai *precision* sekitar 2 kali lipat, sehingga nilai *F-Measure* dapat meningkat secara signifikan.

Namun, untuk dapat melakukan pengujian ganda, diperlukan kamus translasi yang sangat baik, karena agar paragraf uji dapat bertindak sebagai paragraf referensi, diperlukan proses translasi bahasa Indonesia ke bahasa Inggris yang baik, kemudian agar paragraf referensi dapat bertindak sebagai paragraf uji, diperlukan proses translasi bahasa Inggris ke bahasa Indonesia yang baik.

BAB 5
KESIMPULAN

1. Metode *Latent Semantic Analysis* dapat dimanfaatkan untuk melakukan evaluasi terhadap kemiripan antara dua teks dengan membandingkan panjang vektor maupun sudut antar vektor matriks hasil LSA.
2. Pada kedua metode pengujian (distribusi normal dan *learning vector quantization*), keberadaan panjang vektor diwajibkan untuk melakukan evaluasi kemiripan antara dua teks. Nilai sudut antar vektor tidak dapat berdiri sendiri untuk melakukan evaluasi.
3. Pada pengujian dengan metode distribusi normal, nilai pengali batas standar deviasi (*k*) dan nilai batas standar deviasi (*z*) berbanding terbalik dengan nilai *false negatives* (berbanding lurus dengan *recall*), namun berbanding lurus dengan nilai *false positives* (berbanding terbalik dengan *precision*).
4. Pada pengujian dengan algoritma *learning vector quantization*, nilai pemercepat pembelajaran (*alpha* dan *n*) berbanding lurus dengan nilai *false negatives* (berbanding terbalik dengan *recall*), namun berbanding terbalik dengan nilai *false positives* (berbanding lurus dengan *precision*).
5. Hasil Operasi AND secara umum meningkatkan nilai *precision*, namun menurunkan nilai *recall*, serta sedikit meningkatkan nilai *F-Measure*.
6. Untuk mendapatkan hasil pengujian yang baik dengan operasi AND, diperlukan keseimbangan antara *precision* dan *recall* dari masing-masing hasil pengujian.

DAFTAR PUSTAKA

[1] "Kamus versi online/daring (dalam jaringan)," [Online]. Available: kbbi.web.id/plagiarisme. [Diakses 23 November 2015].

[2] "Plagiarism.org," iParadigm, [Online]. Available: http://www.plagiarism.org/plagiarism-101/what-is-plagiarism/. [Diakses 11 November 2015].

[3] Asim M. El Tahir Ali, Hussam M Dahwa Abdulla, Vaclav Snasel, "Overview and Comparison of Plagiarism Detection Tools," dalam *Dateso 2011*, Pisek, Czech Republic, 2011.

[4] Landauer, T.K., Foltz, P.W., Laham, D., "An Introdution to Latent Semantic Analysis," *Discourse Processes,* vol. 25, no. 2-3, pp. 259-284, 1998.

[5] "Spearman's Correlation," [Online]. Available: http://www.statstutor.ac.uk/resources/uploaded/spearmans.pdf. [Diakses 24 November 2015].

[6] "Learning Vector Quantization," [Online]. Available: http://ccy.dd.ncu.edu.tw/~chen/course/neural/ch4/index.htm. [Diakses 24 November 2015].

[7] E. W. Weisstein, "Normal Distribution," MathWorld--A Wolfram Web Resource, [Online]. Available: http://mathworld.wolfram.com/NormalDistribution.html. [Diakses 24 November 2015].

[8] B. Narasimhan, "The Normal Distribution," [Online]. Available: http://statweb.stanford.edu/~naras/jsm/NormalDensity/NormalDensity.html. [Diakses 24 November 2015].

[9] "Normal Distribution," Math is Fun Advanced, [Online]. Available: https://www.mathsisfun.com/data/standard-normal-distribution.html. [Diakses 2014 November 2015].

[10] "PHP," [Online]. Available: http://www.tutorialspoint.com/php/. [Diakses 24 November 2015].

[11] P. Taei, "10 Advantages of PHP Over Other Languages," 25 February 2013. [Online]. Available: http://www.webnethosting.net/10-advantages-of-php-over-other-languages/. [Diakses 24 November 2015].

[12] "MATLAB," Mathworks, [Online]. Available: http://www.mathworks.com/products/matlab/. [Diakses 24 November 2015].

[13] D. Eftaiha, "An Introduction to Apache," 12 July 2012. [Online]. Available: http://code.tutsplus.com/tutorials/an-introduction-to-apache--net-25786. [Diakses 24 November 2015].

[14] Sidik Soleman, Ayu Purwarianti, "Experiments on the Indonesian Plagiarism Detection using Latent Semantic Analysis," dalam *International Conference on Information and Communication Technology*, Bandung, 2014.

[15] M. Littman, S. Dumais, T. Landauer, "Automatic Cross-language Information Retrieval Using Latent Semantic Indexing," *Kluwer Academic Publishers*, vol. 2, pp. 51-62, 1998.

[16] "Intrinsic Plagiarism Detection Using Latent Semantic Indexing and Stylometry," dalam *Developments in eSystems Engineering (DeSE)*, Abu Dhabi, 2013.

[17] C.D. Manning, P. Raghavan, H. Schutze, “Dropping Common Terms: Stop Words,” dalam *Introduction to Information Rerieval*, Cambridge University Press, 2008, pp. 27-28.

[18] E. Lomempow, Analisis Efek Modifikasi Algoritma Latent Semantic Analysis Terhadap Akurasi Deteksi Plagiarisme Paper Bahasa Indonesia dan Inggris, Depok: Universitas Indonesia, 2015.