IMPLEMENTASI METODE *LATENT SEMANTIC ANALYSIS* UNTUK PREDIKSI KESAMAAN PADA PENGAJUAN JUDUL PROPOSAL SKRIPSI

Robi Wanda Kharisma
Teknik Informatika, Universitas Islam Negeri Sunan Gunung Djati Bandung, Jl. AH Nasution No. 105 Bandung
robikharisma96@gmail.com

***Abstrak*- Pengajuan judul skripsi adalah hal yang wajib dilakukan bagi mahasiswa Jurusan Teknik Informatika Universitas Negeri Islam Sunan Gunung Djati Bandung yang ingin membuat skripsi yang merupakan salah satu persyaratan kelulusan mahasiswa untuk memperoleh gelar wisuda. Namun sering kali penguji sidang proposal mengalami kesulitan dalam mencari apakah judul yang diajukan sudah diajukan atau belum. Maka dari itu penulis membuat tugas akhir ini dengan tujuan untuk membantu penguji dalam mencari data judul skripsi dan membandingkannya dengan judul yang diajukan mahasiswa. Dengan metode *Latent Semantic Analysis*, penguji dapat lebih mudah mencari data judul skripsi yang memiliki hubungan antar kata dengan judul yang masukan oleh penguji dalam *textbox* pencarian. Oleh karena itu metode *Latent Semantic Analysis* (LSA) yang mengadaptasi perhitungan *Singular Value Decomposition* (SVD) ini menjadi pilihan untuk membantu penguji dalam pencarian data judul skripsi yang relevan. Berdasarkan pengujian yang telah dilakukan dengan teori pengujian berupa *Recall* dan *Precision*, sistem yang dibuat dapat melakukan pencarian data judul skripsi dengan hasil berupa *list* judul skripsi yang memiliki keterkaitan semantik dengan rata-rata nilai *Recall* sebesar 83.88% dan rata-rata nilai *Precision* sebesar 39.34%.**

***Kata kunci*- Judul Skripsi, LSA, SVD, Relevan.**

I. PENDAHULUAN

A. LATAR BELAKANG

Skripsi atau tugas akhir adalah suatu karya ilmiah yang disusun oleh mahasiswa berdasarkan hasil penelitian yang dilakukan secara seksama dengan bimbingan dosen pembimbing juga merupakan salah satu persyaratan kelulusan mahasiswa [1]. Untuk itu suatu perguruan tinggi tentunya mengharuskan atau mewajibkan mahasiswanya untuk membuat suatu tugas akhir atau skripsi sebagai persyaratan untuk memperoleh gelar sarjana.

Salah satu masalah yang dihadapi oleh lembaga pendidikan perguruan tinggi khususnya di Jurusan Teknik Informatika Universitas Islam Negeri Sunan Gunung Djati Bandung adalah pada pengajuan judul skripsi, dimana parameter untuk memutuskan diterima atau ditolaknya pengajuan judul skripsi yang diajukan dalam sidang proposal oleh mahasiswa selama ini masih dilakukan secara manual, yaitu hanya menurut pertimbangan penguji saja tanpa adanya keterkaitan sistem untuk melakukan pencarian judul skripsi yang memiliki kesamaan dengan judul skripsi yang diajukan sebagai tolak ukur kelulusan sidang proposal tersebut.

Pada dasarnya penentuan diterima dan ditolaknya pengajuan judul proposal skripsi adalah mutlak dari pihak jurusan serta pertimbangan penguji sidang proposal dengan melalui beberapa tahapan masalah yang dibahas kemudian diambil keputusan akhirnya. Hal yang paling utama yang menjadi tolak ukur diterimanya judul proposal skripsi adalah keoriginalitasannya. Artinya bahwa judul yang diajukan adalah satu-satunya yang ada di Jurusan Teknik Informatika ini, dengan kata lain belum pernah ada judul proposal skripsi yang diajukan yang memiliki kesamaan dalam segi semantik dengan judul proposal skripsi di tahun-tahun sebelumnya.

Untuk mengatasi masalah tersebut tentu saja diperlukan suatu metode pencarian yang diterapkan dalam sistem untuk menemukan hubungan semantik antara judul proposal skripsi yang diajukan dengan judul proposal skripsi yang sudah ada. Suatu metode yang mampu menghitung nilai kemiripan semantik antar dua kalimat, yaitu *Latent Semantic Analysis* (LSA).

B. RUMUSAN MASALAH

Berdasarkan latar belakang yang telah dijelaskan di atas, dapat dirumuskan beberapa permasalahan yaitu :

1. Bagaimana menerapkan metode *Latent Semantic Analysis* dalam sistem prediksi kesamaan pada judul proposal skripsi ?
2. Bagaimana kinerja sistem dari penerapan metode *Latent Semantic Analysis* dalam sistem prediksi kesamaan pada judul proposal skripsi ?

C. MANFAAT PENELITIAN

Manfaat yang diambil dari penelitian ini adalah :

1. Membantu penguji ujian proposal untuk menentukan diterima atau ditolaknya judul proposal yang diajukan mahasiswa dengan melihat apakah judul sudah ada atau belum.

2. Membantu pihak administrasi jurusan untuk memeriksa apakah judul yang diajukan mahasiswa sudah ada atau belum di arsip jurusan.
3. Membantu mahasiswa dalam menentukan judul proposal skripsi agar terhindar dari judul yang sama dengan judul skripsi yang sudah ada.

II. LANDASAN TEORI

A. *INFORMATION RETRIEVAL*

Sistem temu kembali informasi (*information retrieval system*) digunakan untuk menemukan kembali (*retrieve*) informasi-informasi yang relevan terhadap kebutuhan pengguna dari suatu kumpulan informasi secara otomatis [2].

Adapun bagian-bagian dari IR *system* meliputi [2]:

1. *Text Operations* (operasi terhadap teks) yang meliputi pemilihan kata-kata dalam *keyword* maupun dokumen (*term selection*) dalam transformasi dokumen atau *keyword* menjadi *term index* (indeks dari kata-kata).
2. *Query formulation* (formulasi terhadap *keyword*) yaitu memberi bobot pada indeks kata-kata *keyword*.
3. *Ranking* (perankingan), mencari dokumen-dokumen yang relevan terhadap *keyword* dan mengurutkan dokumen tersebut berdasarkan kesesuaiannya dengan *keyword*.
4. *Indexing* (pengindeksan), membangun basis data indeks dari koleksi dokumen. Dilakukan terlebih dahulu sebelum pencarian dokumen dilakukan.

IR *system* menerima *keyword* dari pengguna, kemudian melakukan perankingan terhadap dokumen pada koleksi berdasarkan kesesuaiannya dengan *keyword*. Hasil perankingan yang diberikan kepada pengguna merupakan dokumen yang menurut sistem relevan dengan *keyword*. Namun relevansi dokumen terhadap suatu *keyword* merupakan penilaian pengguna yang subjektif dan dipengaruhi banyak faktor seperti topik, pewaktuan, sumber informasi maupun tujuan pengguna [2].

B. STEMMING NAZIEF & ADRIANI

Stemming merupakan suatu proses yang terdapat dalam sistem *Information Retrieval* yang mentransformasi kata-kata yang terdapat dalam suatu dokumen ke kata-kata akarnya (*rootword*) dengan menggunakan aturan-aturan tertentu. Sebagai contoh, kata bersama, kebersamaan, menyamai, akan distem ke root wordnya yaitu "sama". Proses *stemming* pada teks Bahasa Indonesia berbeda dengan *stemming* pada teks berbahasa Inggris. Pada teks berbahasa Inggris, proses yang diperlukan hanya proses menghilangkan sufiks. Sedangkan pada teks berbahasa Indonesia, selain sufiks, prefiks, dan konfiks juga dihilangkan.

Algoritma *stemming* untuk bahasa Indonesia telah dikembangkan, seperti Algoritma Nazief & Adriani untuk teks berbahasa Indonesia. Algoritma yang dibuat oleh Bobby Nazief dan Mirna Adriani ini memiliki tahap-tahap sebagai berikut [3]:

1. Cari kata yang akan distem dalam kamus kata dasar. Jika ditemukan maka diasumsikan bahwa kata tesebut adalah *rootword*. Maka algoritma berhenti.
2. *Inflection Suffixes* ("-lah", "-kah", "-ku", "-mu", atau "-nya") dibuang. Jika berupa *particles* ("-lah", "-kah", "-tah" atau "-pun") maka langkah ini diulangi lagi untuk menghapus *Possesive Pronouns* ("-ku", "-mu", atau "-nya"), jika ada.
3. Hapus *Derivation Suffixes* ("-i", "-an" atau "-kan"). Jika kata ditemukan di kamus, maka algoritma berhenti.
4. Hilangkan *Derivation Prefixes* DP {"di-","ke-","se-","me-","be-","pe", "te-"} dengan iterasi maksimum adalah 3 kali.

Kelebihan dan Kelemahan Algoritma Nazief dan Adriani [3]:

A. Kelebihan :
   1) Memperhatikan kemungkinan adanya partikel-partikel yang mungkin mengikuti suatu kata berimbuhan.
   2) Proses *stemming* dokumen teks Bahasa Indonesia menggunakan Algoritma Nazief dan Adriani memiliki prosentase keakuratan (presisi) lebih besar dibandingkan dengan *stemming* menggunakan Algoritma Porter.

B. Kelemahan :
   1) Penyamarataan makna variasi kata.
   2) Jumlah *Database* kata dan kata dasarnya harus besar. Kesalahan terjadi bila kata tidak ditemukan di *Database* dan kemudian dianggap kata dasar, padahal bukan.
   3) Lamanya waktu yang diperlukan dalam proses pencarian kata di dalam kamus.

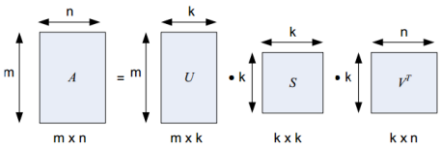
C. *SINGULAR VALUE DECOMPOSITION*

SVD adalah sebuah metode untuk mengidentifikasi dan mengurutkan dimensi yang menunjukan data mana yang menunjukan variasi yang paling banyak. Berkaitan dengan hal itu, SVD dapat mengidentifikasi dimana variasi muncul paling banyak, sehingga hal ini memungkinkan untuk mencari pendekatan yang terbaik pada data asli menggunakan dimensi yang lebih kecil. Oleh karena itu, SVD dapat dilihat sebagai metode pengurangan data [4].

Hal yang mendasari SVD adalah SVD mengambil data asli biasanya terdiri dari variasi matriks kata dan dokumen kemudian memecahnya menjadi komponen independen yang linear.

SVD akan menguraikan sebuah matriks menjadi tiga buah matriks baru yaitu matriks vektor singular kiri, matriks nilai singular, dan matriks vektor singular kanan. SVD dari sebuah matriks A yang berdimensi (m x n) adalah sebagai berikut [4]:

$$A = U\,S\,V^T \quad \text{.....(1)}$$



Gambar 1 Ilustrasi SVD dari nilai A [5].

Hasil SVD adalah matriks U adalah matriks berukuran *m x k* dan V matriks bujur sangkar *n x k*, keduanya mempunyai kolom-kolom ortogonal sedemikian sehingga

$$U^T U = V^T V = 1 \text{.....(2)}$$

dan S adalah matriks diagonal berukuran *k x k*. *Entry – entry* di diagonal utama matriks S adalah nilai singular dari mtriks A.

Hasil SVD dapat lebih dipahami apabila matriks *A* ditulis dengan interpretasi yang berbeda. Bila $u_{1,}u_{2,\ldots,}u_k$ adalah vektor – vektor kolom dari matriks *U* , $\sigma_{1,}\sigma_{2,\ldots}\sigma_k$ adalah *entry – entry* di diagonal utama dari matriks *S* , dan $v_{1,}v_{2,\ldots}v_k$ adalah vektor – vektor kolom dari matriks *V*. Maka matriks *A* dapat ditulis sebagai

$$A = \sum_{i=1}^{k} \sigma_i \, u_i \, v_i^T \text{ ……..(3)}$$

Nilai – nilai σ_1 , untuk *i = 1, 2, … , k* , pada persamaan (3) diurutkan menurun dari yang terbesar sampai terkecil. Apabila beberapa nilai σ_1 yang besar diambil dan nilai σ_1 yang kecil (mendekati nol) dibuang, kita memperoleh suatu aproksimasi dari *A* yang bail. Jadi, dengan SVD, suatu matriks dapat ditulis sebagai penjumlahan dari komponen – komponen ($v_1 \, v_i^T$ untuk *i = 1, 2, …, k*) dengan bobotanya adalah nilai singular (v_1 , untuk *i = 1, 2, … k*) [5].

D. LATENT SEMANTIC ANALYSIS

Latent Semantic Analysis (LSA) adalah suatu metode aljabar yang mengekstrak struktur semantik yang tersembunyi dari kata dan kalimat [6]. Algoritma LSA merupakan salah satu algoritma pengembangan dalam bidang ilmu *Information Retrival* yang mampu menghimpun sejumlah besar dokumen dalam basis data dan mengghubungkan relasi antar dokumen dengan mencocokan masukan yang diberikan. Fungsi utama dari LSA ini adalah untuk menghitung kemiripan (*similarity*) dokumen dengan membandingkan representasi vektor dari dokumen yang berbeda. Penilaian dengan metode LSA lebih kepada kata-kata yang ada dalam tulisan tanpa memperhatikan urutan kata dan tata bahasa dalam tulisan tersebut, sehingga suatu kalimat yang dinilai adalah berdasarkan kata-kata kunci yang ada pada kalimat tersebut [7].

Pada dasarnya LSA ini menggunakan konteks yaitu memasukan dokumen dan mengekstrak informasi dari kata yang digunakan bersama dan kata-kata umum yang sering dilihat pada kalimat yang berbeda. Jika jumlah dari kata – kata umum pada kalimat dalam jumlah banyak, itu berarti kalimat tersebut lebih banyak bersifat semantik [6]. Secara umum, langkah-langkah LSA adalah sebagai berikut [8] :

1. *Text preprocessing*
   Preprocessing adalah proses normalisasi teks sehingga informasi yang dimuat merupakan bagian yang padat dan ringkas namun tetap merepresentasikan informasi yang *term*uat didalamnya. Dalam tahap ini, terdapat beberapa proses diantaranya :
   a. *Stopwords Removal*
      Pada *stopwords removal*, kata-kata yang tergolong sebagai kata depan, kata penghubung, dan kata-kata lain yang tidak mewakili makna dari kalimat akan dieliminasi. Contohnya adalah sebagai berikut [8]:
      1) Menghapus kata-kata sambung seperti “yang, di, ke, dari, pada, dalam, dan”.
      2) Mengubah kata yang diawali dengan huruf besar menjadi huruf kecil.
   b. *Stemming*
      Langkah berikutnya adalah *stemming*. Pada proses ini, kata akan dinormalkan menjadi kata dasar pembentuk kata tersebut. Caranya adalah dengan menghilangkan imbuhan yang melekat pada kata, sehingga hasilnya adalah kata dasarnya.
2. *Term-document matrix*
   Setelah melalui *stopwords removal* dan *stemming*, matriks *term*-document dibangun dengan menempatkan kata hasil proses stemming (*term*) ke dalam baris. Matriks ini disebut *term-document matrix*. Setiap baris mewakili sebuah kata yang unik, sedangkan setiap kolom mewakili konteks dari mana kata-kata tersebut diambil. Konteks yang dimaksud bisa berupa kalimat, paragraf, atau seluruh bagian dari teks. Contoh *term-document matrix* dapat dilihat pada gambar 2.

	Document 1	Document 2	Document 3	Document n
Term 1	1	2	0	N
Term 2	1	0	3	N
Term 3	1	1	0	N
Term 4	1	0	0	N
Term 5	0	0	4	N
Term 6	1	1	0	N
Term 7	1	0	0	N
Term 8	0	2	1	N
Term 9	1	1	0	N
Term n	n	n	n	N

Gambar 2 Contoh *Term-document matrix* [8].

3. *Singular value decomposition* (SVD)
   SVD merupakan teorema aljabar linier yang menyebutkan bahwa persegi panjang dari *term-document matrix* dapat dipecah atau didekomposisikan menjadi tiga buah matriks, yaitu : Muatris Ortogonal U, Matriks Diagonal S, Transpose dari matriks ortogonal V yang dirumuskan dengan :

$$A = U \, S \, V^T \text{ ……..(4)}$$

$$A = [u_{1,}u_{2,\ldots,}u_k] \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_k \end{bmatrix} \begin{bmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_k^T \end{bmatrix} \text{.....(5)}$$

   Hasil dari proses SVD adalah vektor yang akan digunakan untuk menghitung similaritasnya dengan pendekatan cosine *similarity*.
4. *Cosine Similarity Measurement*
   Cosine similarity digunakan untuk menghitung nilai kosinus sudut antara vektor dokumen dengan vektor kueri. Semakin kecil sudut yang dihasilkan, maka tingkat kemiripan esai semakin tinggi. Formula dari *cosine similarity* adalah sebagai berikut :

$$\text{Cos } \alpha = \frac{\text{A.B}}{|\text{A}|.|\text{B}|} \frac{\sum_{i=1}^{n} \text{Ai x Bi}}{\sqrt{\sum_{i=1}^{n} (\text{Ai})^2 \text{ x } \sum_{i=1}^{n} (\text{Bi})^2}} \text{ ……(6)}$$

   Dengan keterangan bahwa A adalah vektor dokumen, B adalah vektor masukan, A **.** B adalah perkalian *dot* vektor A dengan vektor B, |A| adalah panjang vektor A, |B| adalah panjang vektor B, |A| **.** |B| adalah *cross product* antara |A| dengan |B| dan α adalah sudut yang terbentuk antara vektor A dengan vektor B.

E. PENGUJIAN *BLACK-BOX*

Black-box Testing berfokus pada persyaratan fungsional perangkat lunak. Artinya *black-box testing* memungkinkan untuk membuat kumpulan kondisi masukkan yang sepenuhnya akan melakukan semua kebutuhan fungsional untuk program. *Black-box testing* berupaya untuk menemukan kesalahan dalam kategori berikut [9] :

1. Fungsi yang salah atau hilang.
2. Kesalahan antar muka.
3. Kesalahan dalam struktur data atau akses basis data eksternal.
4. Kesalahan prilaku atau kinerja.
5. Kesalahan inisialisasi dan penghentian.

Pengujian *black-box* cenderung diterapkan selama tahap – tahap pengujian selanjutnya.

F. PENGUJIAN *RECALL & PRECISION*

Tujuan uji *Recall* dan *Precision* adalah untuk menilai hasil dari pencarian. *Precision* dapat dianggap sebagai ukuran ketepatan atau ketelitian, sedangkan *Recall* adalah kesempurnaan. Nilai *Precision* adalah tingkat ketepatan antara informasi yang diminta oleh pengguna dengan jawaban yang diberikan oleh sistem. Sedangkan nilai *Recall* adalah tingkat keberhasilan sistem dalam menemukan kembali sebuah informasi [10].

Sistem temu kembali informasi diharapkan untuk dapat memberikan nilai *Recall* dan *Precision* mendekati nilai tertinggi. Pengguna rata-rata ingin mencapai nilai *Recall* tinggi dan *Precision* tinggi, pada kenyataannya hal itu harus dikompromikan karena sulit dicapai [10].

$$R = \frac{Number\ of\ relevan\ items\ retrieved}{Total\ number\ of\ relevan\ items\ in\ collection} \text{ .....(7)}$$

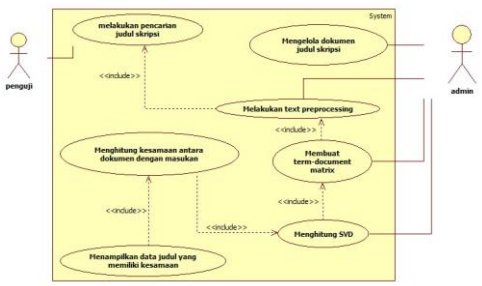
Dengan R adalah *Recall*, maka nilai R didapatkan dengan membandingkan *Number of relevant items retrieved* dengan *Total number of relevant items in collection*. *Recall* adalah dokumen yang terpanggil dari sistem sesuai dengan permintaan *user* yang mengikuti pola dari sistem. Nilai *Recall* makin besar belum bisa dikatakan suatu sistem baik atau tidak.

$$P = \frac{Number\ of\ relevan\ items\ retrieved}{Total\ number\ of\ items\ retrieved} \text{ .....(8)}$$

Dengan P adalah *Precision*. maka nilai P didapatkan dengan membandingkan *Number of relevant items retrieved* dengan *Total number of items retrieved*. *Precision* adalah jumlah dokumen yang terpanggil dari *Database* relevan setelah dinilai user dengan informasi yang dibutuhkan. Semakin besar nilai *Precision* suatu sistem, maka sistem bisa dikatakan baik.

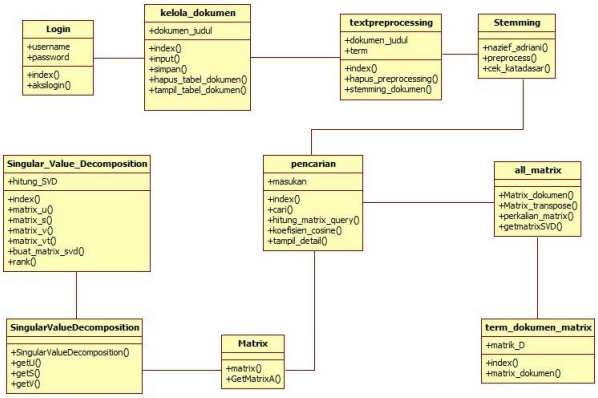
III. PERANCANGAN

A. USECASE DIAGRAM



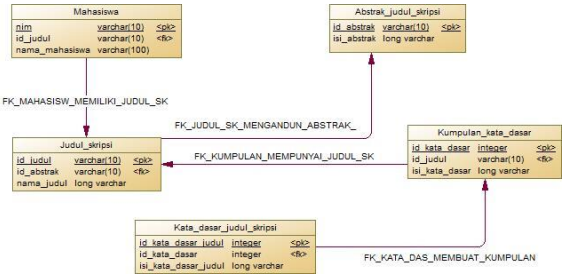
Gambar 3 *UseCase Diagram* Sistem

B. CLASS DIAGRAM



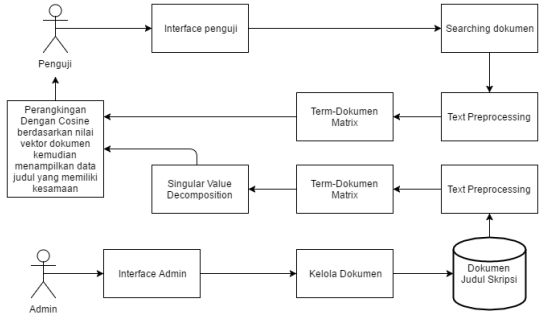
Gambar 4 *Class Diagram* Sistem

C. PERANCANGAN BASIS DATA



Gambar 5 *Class Diagram* Sistem

D. ARSITEKTUR DIAGRAM



Gambar 6 *Class Diagram* Sistem

E. PSEUDO-CODE

Tabel 1 Pseudo-Code *Metode* Latent Semantic Analysis

	Metode *Latent Semantic Analysis*
Deklarasi	Char dok_n , dok_masukan; Float A X, Y, U, S, V, Hasil;
Deskripsi	Write (“masukan dok_n :”); read dok_n; A = dok_n A = U S V^T S = eigenvalue A S^{-1} = Invers S V = eigenvektor A V^T = Tranpose V U = AVS^{-1} Write(“masukan dok_masukan :”); read dok_masukan; X = dok_n * U * S^{-1} Y = dok_masukan * U * S^{-1} Hasil = (X * Y) ÷(|X|*|Y|)

IV. IMPLEMENTASI & PENGUJIAN

A. IMPLEMENTASI ANTARMUKA

1) Halaman Pencarian

Tampilan halaman pencarian judul merupakan tampilan pertama yang akan dilihat oleh penguji. Halaman pencarian judul berisikan fungsi pencarian judul skripsi oleh penguji. Berikut ini adalah tampilan dari halaman pencarian judul yang akan terlihat pada Gambar 7.

Gambar 7 Halaman Pencarian

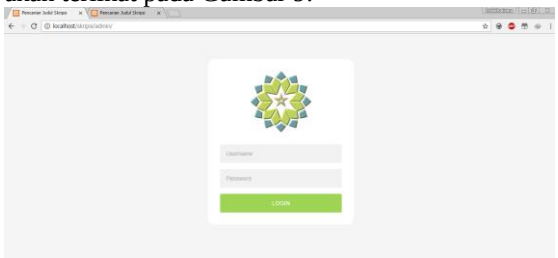
2) Halaman Hasil Pencarian

Tampilan halaman hasil pencarian adalah tampilan hasil pencarian dari halaman pencarian judul dan merupakan halaman kedua yang akan dilihat oleh penguji. Halaman hasil pencarian berisikan fungsi hasil pencarian judul skripsi yang dilakukan oleh penguji. Berikut ini adalah tampilan dari halaman hasil pencarian yang akan terlihat pada Gambar 8.

Gambar 8 Halaman Pencarian

3) Halaman Login Admin

Tampilan halaman *Login* admin merupakan tampilan dimana admin melakukan *Login* agar bisa masuk ke dalam halaman hak akses admin. Berikut ini adalah tampilan dari halaman *Login* admin yang akan terlihat pada Gambar 9.



Gambar 9 Halaman Pencarian

4) Halaman Admin

Tampilan halaman admin merupakan tampilan dimana admin bisa mengelola data dokumen sekaligus mengolahnya untuk kebutuhan kinerja sistem.. Berikut ini adalah tampilan dari halaman admin kelola dokumen yang akan terlihat pada Gambar 10.

Gambar 10 Halaman Pencarian

B. PENGUJIAN RECALL & PRECISION

Pengujian dilakukan dengan mencoba semua *query* judul pada sistem. Nilai *Recall* dan nilai *Precision* dicari dengan menggunakan rumus (7) untuk *Recall* dan (8) untuk *Precision*. Hasil pengujian *Recall* dan *Precision* terhadap *query* judul yang diujikan dapat dilihat pada tabel .

Query Judul	Data relevan muncul	Data irrelevan muncul	Total data relevan	*Recall* (%)	*Precision* (%)
Sistem Pendukung Keputusan Untuk Seleksi Beasiswa Siswa Sma	1	2	1	100	33.33
Sistem Pendukung Keputusan Pemberian Kredit	1	5	2	50	16.67
Aplikasi Pengolahan Data Nikah Dan Cerai Berbasis Web	2	4	2	100	33.33
Aplikasi Pengolahan Data Gaji	1	6	1	100	14.29
Sistem Informasi Pinjaman Uang Nasabah Koperasi	2	3	2	100	40
Sistem Informasi Kelola Data Di Perpustakaan	2	2	2	100	50
Aplikasi Pencarian Rute Terpendek	4	3	7	57.14	57.14
Aplikasi Keamanan Citra Digital	2	0	5	40	100
Pembuatan Sistem Informasi Akademik	1	4	1	100	20
Sistem Informasi Pengelolaan Keuangan	2	4	2	100	33.33
Sistem Pakar Pendeteksi Kerusakan Hardware Pada Komputer	2	5	2	100	28.57
Sistem Pengolahan Data	1	6	1	100	14.29

Administrasi Pasien					
Sistem Pengolahan Data Rental Bus	1	3	1	100	25
Rancang Bangun Sistem Informasi Penerimaan Mahasiswa Baru	2	4	2	100	33.33
Sistem Informasi Geografis Untuk Pencarian Tempat Ibadah Di Bandung	1	4	1	100	20
Rancang Bangun Sistem Informasi Pemesanan Tiket	3	3	3	100	50
Perancangan Sistem Informasi Penjualan	3	5	8	37.5	37.5
Perancangan Aplikasi Game Pengenalan Flora Dan Fauna	3	2	3	100	60
Perancangan Aplikasi Game Pengenalan Alat Musik	2	5	3	66.67	28.57
Perancangan Sistem Informasi Jalur Angkutan Kota	3	4	3	100	42.86
Aplikasi Pencarian Terjemahan Al-Quran	3	6	3	100	33.33
Sistem Pendukung Keputusan Rekrutmen Pemain Sepakbola	1	0	1	100	100
Perancangan Sistem Perhitungan Zakat Menurut Islam	3	6	3	100	33.33
Metode A*	0	2	2	0	0
Rata – Rata (%)				83.88	39.34

V. KESIMPULAN

Berdasarkan pengujian yang dilakukan sistem temu kembali informasi dengan menggunakan metode *Latent Semantic Analysis*, terbukti bahwa metode *Latent Semantic Analysis* dapat diimplementasikan dalam sistem prediksi kesamaan pada judul proposal skripsi. Setelah melakukan pengujian sebanyak 40 kali dengan *query* yang berbeda-beda pada setiap pengujiannya, Metode LSA terbukti dapat melakukan pencarian data dokumen judul skripsi berdasarkan nilai kemiripan antara dokumen judul skripsi dengan judul skripsi yang diinputkan penguji dengan hasil rata-rata nilai *Recall* sebesar 83.88% dan rata-rata nilai *Precision* sebesar 39.34%. Namun metode ini tidak dapat mengeksekusi judul yang berisi simbol-simbol karena tidak terdapat di *Database* kata dasar.

VI. REFERENSI

[1] T. W. UNY, "Tugas Akhir," 2012. [Online]. Available: https://uny.ac.id/akademik/tugas-akhir. [Accessed: 20-Sep-2016].

[2] T. Jaya, "Sistem Temu Kembali Informasi Terjemah Tafsir Al-Quran berdasarkan Teks Hadits dengan Metode Latent Semantic Indexing," pp. 1–8, 2016.

[3] D. Keke, R. Chikita, and A. D. Prayoga, "Sistem Temu Balik Informasi," 2012.

[4] R. B. Aji, Z. A. Baisal, and Y. Firdaus, "Automatic Essay Grading System Menggunakan Metode Latent Semantic Analysis," pp. E78–E86, 2011.

[5] H. Bunyamin, "Information Retrieval System dengan Metode Latent Semantic Indexing," Intitut Teknologi Bandung, 2005.

[6] M. Jamhari, E. Noersasongko, and H. Subagyo, "Pengaruh Peringkas Dokumen Otomatis Dengan Penggabungan Metode Fitur dan Latent Semantic Analysis (LSA) Pada Proses Clustering Dokumen Teks Berbahasa Indonesia," vol. 1, no. 2, pp. 72–82, 2014.

[7] M. Sofyan, "Pengembangan Koreksi Esai Otomatis Pada E-Learning Di SMK PLUS AN-NABA SUKABUMI Dengan Menggunakan Metode Latent Semantic Analysis (LSA)," pp. 45–52, 2015.

[8] S. Derwin, "Penggunaan Latent Semantic Analysis (LSA) Dalam Pemrosesan Teks," 2015. [Online]. Available: http://socs.binus.ac.id/2015/08/03/penggunaan-latent-semantic-analysis-lsa-dalam-pemrosesan-teks/. [Accessed: 01-Jan-2016].

[9] R. S. Pressman, *Rekayasa Perangkat Lunak*, Edisi 7. Yogyakarta: Andi, 2012.

[10] F. Amin, "Sistem Temu Kembali Informasi dengan Pemeringkatan Metode Vector Space Model," *J. Teknol. Inf. Din.*, vol. 18, no. 2, pp. 122–129, 2012.