ANALISIS SENTIMEN ULASAN VIDEO ANIMASI MENGGUNAKAN METODE *LATENT SEMANTIC INDEXING*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Faraz Dhia Alkadri
NIM: 135150200111039

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENGESAHAN

ANALISIS SENTIMEN ULASAN VIDEO ANIMASI MENGGUNAKAN METODE *LATENT SEMANTIC INDEXING*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan

Memperoleh gelar sarjana komputer

Disusun Oleh :

Faraz Dhia Alkadri

NIM : 135150200111039

Skripsi telah diuji dan dinyatakan lulus pada

1 Agustus 2018

Telah diperiksa dan disetujui oleh :

Dosen Pembimbing I	Dosen Pembimbing II
Yuita Arum Sari, S.Kom., M.Kom	Ir. Sutrisno, M.T
NIK. 21060988 0715 2 001	NIP. 19570325 198701 1 001

Mengetahui

Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan , S.T, M.T, Ph.D

NIP. 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 5 Agustus 2018

Faraz Dhia Alkadri

NIM: 135150200111039

KATA PENGANTAR

Puji syukur saya panjatkan kehadirat Tuhan Yang Maha Esa, karena berkat rahmat dan hidayah-Nya, penulis dapa menyelesaikan laporan Skripsi yang berjudul "Analisi Sentimen Ulasan Video Animasi Menggunakan Metode *Latent Semantic Indexing*.

Skripsi ini merupakan salah satu mata kuliah wajib sebagai syarat kelulusan di Fakultas Ilmu Komputer Universitas Brawijaya. Laporan ini disusun sebagai syarat kelulusan dalam studi di Universitas Brawijaya.

Dalam penyusunan laporan ini penulis mendapatkan banyak bantuan dari berbagai pihak baik secara moril maupun matriil. Dalam kesempatan ini penulis ingin mengucapkan terimakasih yang sebesar-besarnya kepada :

1. Ibu Yuita Arum Sari, S.Kom., M.Kom, selaku Dosen Pembimbing 1 dan Bapak Sutrisno, Ir., M.T, selaku Dosen Pembimbing 2 yang telah membimbing, memberi saran serta motivasi kepada penulis selama penyusunan skripsi ini.
2. Seluruh dosen dan civitas akademika Fakultas Ilmu Komputer yang telah memberikan wawasan, bantuan dan dukungan yang sangat bermanfaat bagi penulis.
3. Kedua orang tua penulis, atas segala dukungan yang diberikan kepada penulis, terutama Ibu tercinta saya.
4. Teman-teman dari EROganization yang telah banyak membantu kelancaran pelaksanaan dan penyusunan laporan Skripsi ini.

Saya menyadari bahwa dalam penyusunan laporan Skripsi ini masih banyak kekurangan baik dari segi susunan maupun isi. Oleh karena itu, saya dengan senang hati menerima kritik dan saran yang membangun sebagai pedoman perbaikan kedepannya.

Malang, 5 Agustus 2018

Penulis

farazalk@gmail.com

ABSTRAK

Faraz Dhia Alkadri., Analisis Sentimen Ulasan Video Animasi Menggunakan Metode *Latent Semantic Indexing*

Pembimbing : Yuita Arum Sari, S.Kom., M.Kom. Ir. Sutrisno, M.T.

Video animasi berkembang dengan sangat pesat menghasilkan puluhan bahkan ratusan judul per-tahunnya. Tentunya tidak semuanya video animasi yang dihasilkan menarik untuk ditonton. Beberapa dari video tersebut mungkin terlihat tidak menarik untuk beberapa orang. Untuk mengatahui apakah video animasi tergolong menarik atau tidak, pengguna dapat melihat ulasan–ulasan yang diberikan pengguna lain tentang video animasi tersebut. Beberapa *website* yang memang bertujuan untuk memfasilitasi para penggunanya untuk dapat saling memberikan *feedback* berupa ulasan tentang video animasi yang telah mereka tonton. Dari ulasan tersebut dapat dilihat sentimennya apakah ulasan tersebut merupakan ulasan yang tergolong ke dalam sentimen positif atau sentimen negatif. Metode *Latent Semantic Indexing* (LSI) yang mengadopsi proses reduksi matriks *Singular Value Decomposition* (SVD) digunakan untuk mencari relevansi antar dokumen. Dengan metode LSI dapat mengetahui ulasan tersebut tergolong pada sentimen positif atau sentimen negatif. Metode TF IDF digunakan untuk mengolah data tekstual menjadi data numerik dan *Cosine similiarty* yang digunakan untuk menghitung kemiripan antar data yang selanjutnya digolongkan ke dalam sentimen kelas positif dan sentiment kelas negatif. pengujian dilakukan sebanyak 19 kali dengan menggunakan masukkan *k-rank* yang berbeda-beda. Berdasarkan hasil pengujian, sistem ini menghasilkan akurasi optimal di *k-rank = 10* yaitu sebesar 86% sehingga dapat disimpulkan bahwa penggunaan *metode latent semantic indexing* baik digunakan untuk mencari relevansi antar dokumen.

Kata Kunci : Sentimen analisis, *Latent Semantic Indexing*, *Singular Value Decomposition*.

ABSTRACT

Faraz Dhia Alkadri., ***Sentiment Analysis on Review of Animation Video Using Latent Semantic Indexing***

Supervisiors **: Yuita Arum Sari, S.Kom., M.Kom. Ir. Sutrisno, M.T.**

Animation videos are growing significantly producing tens even hundreds of titles per year. Certainly not everything were produced was interesting. Some of these videos may not be appealing to some people. To find out whether the animated videos is interesting or not, users can read the reviews given by other user about animation videos. Some sites that are intended to facilitate its users to be able give each other feed back about the animation video they have watched. From those reviews can be seen sentiment whether the review is a review that classified in to positive class sentiment or negative class sentiment. The Latent Semantic Indexing (LSI) method that adopts the Singular Value Decomposition (SVD) matrix reduction process is used to find the relevance between documents. With the LSI method helps us to be able to know the reviews are classified on positive sentiment or negative sentiment. The TF IDF method is used to process textual data into numerical data and cosine similiarity method is used to calculate the similiarity between data which is further classified into positive class sentiment and negative class sentiment. Testing done as much as 19 times by using different k-rank input. Based on the test result, this system produces an optimal accuracy on k-rank =10 that is equal to 86% so we can conclude that latent semantic indexing is good to use for searching relevance between documents.

Key Word : Sentiment analysis, *Latent Semantic Indexing*, *Singular Value Decomposition*.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR KODE PROGRAM

BAB 1
PENDAHULUAN

1.1 Latar belakang

Internet saat ini berkembang dengan sangat pesat. Pemanfaatan internet juga sudah mencakup berbagai bidang seperti media sosial, media informasi, media riset, media bisnis, media entertainment, dan media lainnya. Media *social* seperti *facebook*, *Twitter* dan beberapa media sosial lainnya digunakan banyak orang untuk mengekspresikan pendapat mereka pada berbagai topik. Jutaan tweet yang berisikan pikiran, pernyataan, komentar, ulasan dan kritik di *post* setiap harinya. Salah satu contohnya adalah ulasan, banyak situs yang menyediakan ulasan tentang suatu produk, dari ulasan-ulasan tersebut pengguna lain dapat mengetahui tentang kelebihan ataupun kekurangan produk tersebut. Sebanyak 81% pengguna internet melakukan *online research* setidaknya satu kali pada pada produk yang akan mereka beli. Hal ini menunjukan bahwa ulasan sangat berpengaruh pada pengambilan keputusan seseorang akan sebuah produk (Pang & Lee, 2008).

Analisis sentimen adalah proses mengekstrak pendapat, prilaku dan emosi dari sebuah dokumen teks untuk mendapatkan suatu informasi (Kristiyanti, 2015). Dokumen yang digunakan berupa ulasan yang selanjutnya akan dilakukan proses klasifikasi. Proses klasifikasi dilakukan untuk pemisahan apakan ulasan termasuk pada sentiment positif atau negative (Medhat, Hassan, & Korashy, 2014) . Dalam Kasus ini topik yang akan digunakan yaitu ulasan video animasi, sehingga pengguna lain dapat mengetahui apakah video animasi tersebut memiliki ulasan positif atau negatif.

Pada tahun 2017, penelitian tentang analisis sentiment *review* film dilakukan oleh Setyo Budi. Penelitian ini menggunakan algoritma *K-Means* dan *clustering* digunakan sebagai proses klasifikasi. Penelitian dilakukan dengan menguji data ke varian dataset yaitu 300 data positif dan 300 data negatif, 700 data positif dan 700 data negatif, 1000 data positif dan 1000 data negatif. Hasil akurasi dari pengujian di atas yaitu 57,83% untuk dataset dengan jumlah 300 data positif dan 300 data negatif, 56,71% untuk data et dengan jumlah 700 data positif dan 700 data negative, 50,40% untuk dataset dengan jumlah 1000 data positif dan 1000 data negatif. Dari hasil pengujian diatas dapat disimpulkan bahwa semakin besar dataset yang digunakan maka semakin rendah akurasi yang dihasilkan.

Pada tahun 2012 Sari, et al., melakukan penelitian tentang penentuan lirik lagu berdasarkan emosi. Metode yang digunakan adalah LSI (*Latent Semantic Indexing*). Hasil Penelitian didapat dari evaluasi MAP (*Mean Average Precision*) pada masing-masing k-rank 300,200,100,50 dan 10 dengan jumlah lirik lagu sebanyak 370 lirik menghasilkan nilai MAP mendekati 1. Dari peneltian tersebut

didapatkan kesimpulan bahwa semakin kecil input k-rank makan rata-rata akurasi sistem akan semakin baik.

Dari penelitian di atas *Latent Semantic Indexing* yang mengadposi proses matematis reduksi dimensi *Singular Value Decomposition* (SVD) memungkinkan suatu data untuk di uji kemiripannya dengan sekumpulan data. Lalu menghasilkan nilai kedekatan antara data yang uji ke sekumpulan data tersebut. metode LSI cocok diimplementasikan pada analisis sentiment ulasan video animasi yang akan dibagi menjadi 2 kategori kelas , yaitu kelas positif dan kelas negatif. Dan melihat hasil dari penggunaan metode LSI berdasarkan ulasan video animasi.

1.2 Rumusan masalah

Berdasarkan latar belakang masalah yang telah dikemukakan sebelumnya, maka masalah yang dirumuskan adalah sebagai berikut :

1. Bagaimana melakukan analisis sentimen kalimat yang di dapat dari ulasan yang disampaikan *reviewer*?
2. Bagaimana tingkat akurasi metode *Latent Semantic Indexing* untuk analisis sentimen pada rulasan video animasi?

1.3 Tujuan

1. Menerapkan algoritma *Latent Semantic Indexing* sehingga mampu mengklasifikasi kata-kata yang mewakili isi dokumen ulasan.
2. Menguji tingkat akurasi sistem analisis sentiment pada ulasan video animasi yang menggunkaan metode *Latent Semantic Indexing*.

1.4 Manfaat

Manfaat penelitian ini bagi penulis adalah untuk dapat menerapkan ilmu yang didapat selama mengikuti perkuliahan. Manfaat lain dari penelitian ini untuk penelitian selanjutnya yang memerlukan pakar dapat membuat sistem tanpa pakar dan juga agar dapat dilanjutkan untuk penelitian penelitian selanjutnya yang meneliti topik yang sama.

1.5 Batasan masalah

Agar pembahasan bisa lebih berfokus, maka sesuai dengan rumusan masalah batasan masalah yang perlu diberlakukan dalam skripsi ini adalah sebagai berikut:

1. Data yang digunakan yaitu data ulasan video animasi yang diambil dari web http://www.myanimelist.net
2. Data yang akan diolah adalah data berbahasa inggris.
3. Jenis data yang berupa teks diubah menjadi matriks melalui metode pemrosesan teks.

4. Data latih yang digunakan sebanyak 200 data latih dan 50 data uji yang akan di uji dalam berbagai variasi,
5. Kelas yang akan digolongkan dibagi menjadi dua jenis yaitu kelas positif dan kelas negatif.

1.6 Sistematika pembahasan

BAB I PENDAHULUAN

Bab Pendahuluan mencakup Latar Belakang penelitian, rumusan masalah yang disertai dengan tujuan penelitian, batasan masalah dan manfaat dari analisis sentimen ulasan video animasi menggunakan metode *Latent Semantic Indexing*.

BAB II LANDASAN KEPUSTAKAAN

Bab ini mendeskripsikan dasar teori dan penelitian-penelitian sebelumnya yang berhubungan dengan topic pada skripsi analisis sentimen ulasan video animasi menggunakan metode *Latent Semantic Indexing.*

BAB III METODOLOGI

Bab Metodologi beriti tahapan-tahapan yang harus di buat, berisi langkah kerja untuk membangun sistem analisis sentimen ulasan video animasi menggunakan metode *Latent Semantic Indexing*.

BAB IV ANALISIS DAN PERANCANGAN

Bab Perancangan berisi antarmuka dan segala sesuatu yang diperlukan dalam rancangan pada sistem analisis sentimen ulasan video animasi menggunakan metode *Latent Semantic Indexing*.

BAB V IMPLEMENTASI DAN PENGUJIAN

Bab ini membahas tentang implementasi dari sistem analisis sentimen ulasan video animasi menggunakan metode *Latent Semantic Indexing* secara keseluruhan, serta melakukan pengujian terhadap system untuk mengetahui system tersebut dapat menyelesaikan permasalahan yang dihadapi sesuai yang diharapkan.

BAB VI KESIMPULAN DAN PENUTUP

Bab ini memuat kesimpulan yang di dapat dari hasil sistem analisis sentimen ulasan video animasi menggunakan metode *Latent Semantic Indexing* dan saran saran untuk pengembangan selanjutnya.

BAB 2
LANDASAN KEPUSTAKAAN

2.1 Kajian Pustaka

Pada tahun 2015 Kristiyanti melakukan penelitian tentang analisis sentiment *review* produk kosmetik. Penelitian ini menggunakan metode *SVM* (*Support Vector Machine*) untuk proses klasifikasi dan PSO (*Particle Swarm Optimization*) sebagai metode seleksi fitur. Pengujian dilakukan dengan menggunakan metode *Support Vector Machine* dan *Support Vector Machine* berbasis *Particle Swarm Optimization* dengan menggunakan data *review* produk kosmetik sebanyak 200 data. Dari hasil pengujian didapatkan bahwa nilai akurasi untuk pengujian menggunakan *Support Vector Machine* adalah 89%. Sedangkan untuk pengujian dengan menggunakan *Support Vector Machine* berbasis *Particle Swarm Optimization* mendapatkan nilai akurasi 97%. Dapat disimpulkan bahwa penggunakan *Support Vector Machine* berbasis *Particle Swarm Optimization* lebih baik dari pada *Support Vector Machine*.

Pada tahun 2017, penelitian tentang analisis sentiment *review* film dilakukan oleh Setyo Budi. Penelitian ini menggunakan algoritma *K-Means* dan *clustering* digunakan sebagai proses klasifikasi. Penelitian dilakukan dengan menguji data ke varian dataset yaitu 300 data positif dan 300 data negatif, 700 data positif dan 700 data negatif, 1000 data positif dan 1000 data negatif. Hasil akurasi dari pengujian di atas yaitu 57,83% untuk dataset dengan jumlah 300 data positif dan 300 data negatif, 56,71% untuk data et dengan jumlah 700 data positif dan 700 data negative, 50,40% untuk dataset dengan jumlah 1000 data positif dan 1000 data negatif. Dari hasil pengujian diatas dapat disimpulkan bahwa semakin besar dataset yang digunakan maka semakin rendah akurasi yang dihasilkan.

Pada tahun 2015 Kristiyanti melakukan penelitian tentang analisis sentiment *review* produk kosmetik. Metode yang digunakan adalah *Support Vector Machine* dan menggunakan seleksi fitur dengan metode *Particle Swarm Optimization*. Penelitian dilakukan dengan membandingan hasil pengujian antara penggunaan metode *support vector machine* dengan penggunaan metode vector machine yang telah dilakukan seleksi fitur menggunakan metode *particle swarm optimization*. Dari hasil pengujian didapatkan akurasi 89,00% untuk penggunaan metode *support vector machine*. Untuk pengujian dengan menggunakan seleksi fitur didapatkan akurasi 97,00%, dapat disimpulkan bahwa terjadi peningkatan yang cukup signifikan pada pengujian mengugnakan seleksi fitur, dengan demikian penggunaan seleksi fitur *particle swarm optimization* memberikan pemecahan untuk permasalahakan klasifikasi *review* produk kosmetik menjadi lebih akurat.

Pada tahun 2012 Sari dan Ridok melakukan penelitian tentang penentuan lirik lagu berdasarkan emosi. Metode yang digunakan adalah *Latent Semantic Indexing*. Hasil Penelitian didapat dari evaluasi MAP (*Mean Average Precision*) pada masing-masing k-rank 300,200,100,50 dan 10 dengan jumlah lirik lagu sebanyak 370 lirik menghasilkan nilai MAP mendekati 1. Dari peneltian tersebut didapatkan kesimpulan bahwa semakin kecil input k-rank makan rata-rata akurasi sistem akan semakin baik.

Pada tahun 2013 Sari dan Puspaningrum melakukan penelitian mengenai pencarian informasi dengan mengurangi dimensi pencarian menggunakan reduksi fitur. Metode yang digunakan adalah *Essential Dimension of Latent Semantic Indexing* dan reduksi fitur menggunakan *Document Frequency thresholding* dan *Information Gain*. Penelitian ini dilakukan 2 kali pengujian dengan jumlah data yang berbeda yaitu 90 data dan 120 data. Dari hasil pengujian di dapatkan nilai MAP (*Mean Average Precision*) optimal ketika *threshold* 0,9, k-rank = 2, dan x = 0,7.

2.2 Text Mining

Text Mining dapat didefinisikan sebagai penerapan teknik data mining untuk penemuan otomatis pada informasi yang berguna atau menarik dari teks yang tidak terstruktur. Teks mining digunakan untuk mengekstrak informasi yang berguna dari data yang tak terstruktur tersebut. Beberapa pengembangan *teks mining* telah diakukan, termasuk struktur konseptual, pohon keputusan dan metode induki aturan. Selain itu *Information Retrieval* (IR) telah banyak digunakan sebagai model untuk tugas seperti pencocokan dokumen, rangking dan klasterisasi. (Ipmawati, 2017). Terdapat beberapa proses dalam melakukan pemrosesan teks yakni *case folding, tokenisasi, filtering,* dan *stemming*.

2.2.1 Case Folding

Proses *Casefolding* memisahkan setiap kata dari sekumpulan teks, dan mengubah semua huruf menjadi huruf kecil dan membuang karakter selain huruf, seperti angka dan tanda baca. Hasil dari proses *Case Folding* adalah kata-kata yang sudah terpisah dan tidak memiliki karakter selain huruf (Vijayarani., et al, 2015).

2.2.2 Tokenisasi

Tokenisasi merupakan proses pemecahan dari sekumpulan teks yang besar menjadi beberapa bagian. Beberapa bagian tersebut dapat disebut kalimat, kalimat yang dihasilkan kemudian di pisah menjadi kata-kata (Vijayarani., et al, 2015).

2.2.3 *Filtering / Stopword Removal*

Sebuah proses untung menghilang kata yang tidak penting dari hasil Tokenisasi. Untuk menghilangkan kata yang tidak penting digunakan stopword yang nantinya setiap kata dari hasil casefolding di periksa dengan stopword. Apabila kata dari hasil casefolding terdapat pada daftar stopword maka kata tersebut di hilangkan (Vijayarani., et al, 2015).

2.2.4 *Stemming*

Stemming merupakan suatu proses untuk merubah kata dari hasil filtering menjadi kata dasar dengan cara menghilangkan semua imbuhan, imbuhan yang harus di hilangkan adalah awalan, akhiran, sisipan. Kombinasi awalan dan akhiran pada kata turunan pun juga harus dihilangkan. *Stemming* digunakan untuk memangkas kata dari bentuk asli yang masih terdapat awalan, akhiran atau sisipan menjadi suatu kata dasar (Yamout, 2004).

2.3 Term Frequency & Inverse Document frequency (TF-IDF)

Metode TF-IDF merupakan metode *text mining* yang digunakan untuk mengekstraksi kata-kata yang bermakna dari sebuah dokumen. Untuk mengetahui makna sebuah dokumen ,dilakukan dengan memberikan bobot pada setiap kata *(term)*. Metode ini akan menghitung nilai *Term Frequency (TF)* dan *Inverse Document Frequency (IDF)* pada setiap *term* di setiap dokumen.

Dalam perhitungan bobot menggunakan TF-IDF, dihitung terlebih dahulu nilai *Term Frequency (TF)*. *TF* yaitu banyaknya kemunculan sebuah *term* dalam suatu dokumen. Selanjutnya menghitung nilai *Inverse Document Frequency (IDF)*. *IDF* adalah perhitungan yang digunakan untuk mengecek berapa banyak dokumen yang memiliki hubungan antara sebuah *term* dalam semua dokumen. Untuk menghitung nilai *IDF* digunakan Persamaan 2.1

$$IDF\ (word) = log\ \frac{td}{df} \tag{2.1}$$

IDF(word) adalah nilai IDF dari setiap kata yang akan dicari, *td* adalah jumlah keseluruhan dokumen yang ada, *df* adalah jumlah dokumen yang memiliki kata yang akan dicari. (Oh, 2013).

2.4 Analisis Sentimen

Analisis sentimen adalah bidang yang mengevaluasi emosi dan perasaan dalam sebuah teks. proses memahami, mengekstrak dan mengolah data tekstual secara otomatis untuk memperoleh informasi. Analisis sentiment dalam skripsi ini adalah proses klasifikasi dokumen ke dalam dua kelas, yaitu kelas positif dan negatif. Pada dasarnya anilisis sentimen merupakan klasifikasi, tetapi karena tidak adanya intonasi dalam sebuah teks menybabkan penggunaan kata menjadi ambigu sehingga membuat proses klasifikasi tidak semudah proses klasifikasi yang ada(Liu, 2012).

Manfaat analisis sentimen salah satunya yaitu untuk mengetahui survei mengenai produk, organisasi dan lain-lain. Proses analisis sentiment pada penelitian untuk mengetahui sentimen terhadap suatu video animasi yang di dapatkan dari ulasan ulasan yang diberikan oleh pengguna. Proses di atas dilakukan dengan metode *Latent Sematic Indexing* dan dikhusukan pada dokumen berbahasa inggris pada sebuah situs *web* (Liu, 2012).

2.5 Latent Semantic Indexing

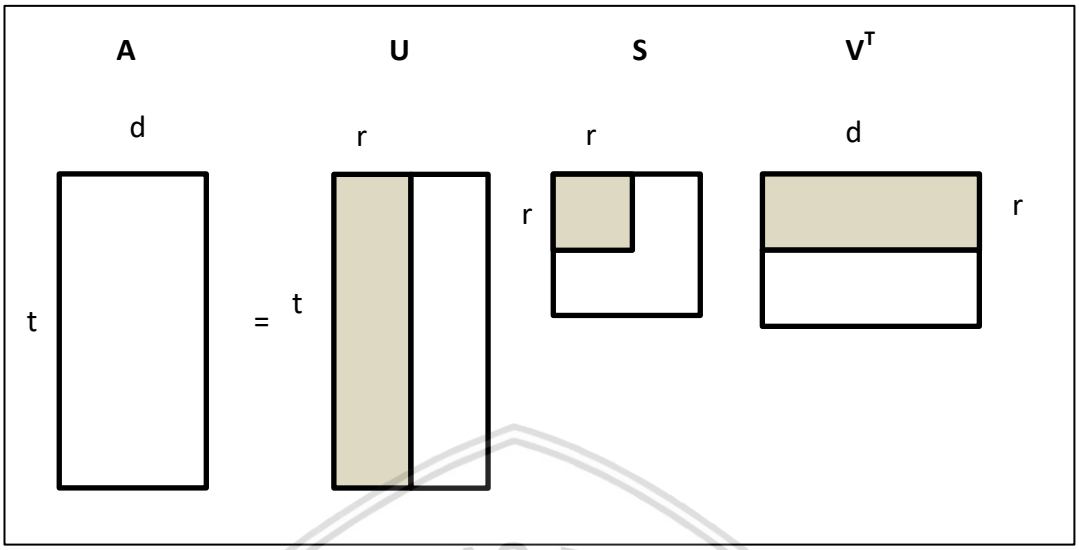
Latent Semantic Indexing merupakan salah satu metode klasifikasi dengan menggunakan metode *machine learning* (*supervised learning*) yang memprediksi kelas berdasarkan pola dari hasil proses training yang diciptakan oleh Vladimir Vapnik. Klasifikasi dilakukan dengan garis pembatas (*hyperlane*) yang memisahkan antara kelas opini positif dan opini negatif. Secara intuitif, suatu garis pembatas yang baik adalah memiliki jarak terbesar ke titik data pelatihan terdekat dari setiap kelas Karena pada umunya semakin bersar margin, semakin redah *error* generalisasi dari pemilah. Margin adalah jarak dari suatu titik vector di suatu kelas terhadap *hyperplane*. Representasi dari LSI adalah

$$q' = q^T . U_k . S_k^{-1} \quad (2.2)$$

Dimana variabel q' adalah *query vector* dari LSI, q^T adalah *transpose* dari pembobotan TF-IDF ternormalisasi, U_k adalah reduksi dimensi k dari matriks U, dan S_k^{-1} adalah *inverse* dari reduksi dimensi k matriks S (Sari, 2012).

2.6 *Singular Value Decomposition* (SVD)

Singular Value Decomposition adalah metode yang memecah matriks A yaitu sebuah matriks (*terms-dokumen*) yang memiliki dimensi (t x d) yang akan dikomposisi menjadi tiga buah matriks USV. Matriks *U* adalah matriks kata terms berukuran (t x r), Matriks *S* adalah matriks diagonal berisi nilai scalar (*Eigen Values*) berdimensi (r x r), dan V adalah matriks yang berukuran (r x d). Dekomposisi matriks A menjadi tiga buah matriks dinyatakan sebagai $A = USV^T$.



Gambar 2.1 Ilustrasi Matriks SVD

SVD digunakan untuk mereduksi dimensi matriks A dengan cara mengurangi panjang dimensi r dari matriks diagonal S. Pengurangan dimensi matriks S dilakukan dengan mengubah nilai diagonal matriks S menjadi nolmkecuali untuk nilai diagonal dari dimensi yang tersisa. Pengalian ketiga matriks USV^T akan membentuk matriks A awal dengan setiap elemenya mendekati nilai sebenarnya (Sari, 2012).

2.7 *Vector Space Model* (VSM)

Vector Space Model adalah cara konvensional yang biasa digunakan pada proses temu kembali informasi dengan menghitung kemiripan dua buah vector, yaitu vector dari data uji dan vector dari data latih. penghitungan kemiripan antara dua buah vektor tersebut menggunakan rumus *cosine similiarity* (Sari, 2012).

$$\boldsymbol{similiarity} = \mathbf{cos}(\theta) = \frac{\sum q.d}{||q||||d||} = \frac{\sum_{i=1}^{n} q_i x\, d_i}{\sqrt{\sum_{i=1}^{n} q_i^2}\, x \sqrt{\sum_{i=1}^{n} d_i^2}} \qquad (2.3)$$

Dimana nilai *q* adalah nilai dari hasil perhitungan *LSI* . *d* merupakan nilai matriks V dari dekomposisi SVD. dimana nilai dimensi dari matriks V merupakan hasil input *k* yang merupakan nilai reduksi dengan *k* ≤ *min (m x n)*, dimana *m* adalah banyaknya *term* dan *n* adalah banyaknya dokumen (Sari. 2012).

2.8 *Confusion Matrix*

Confusion Matrix adalah sebuah sistem yang mengandung informasi tentang klasifikasi secara akutal dan pridiktif . Kinerja suatu sistem dievaluasi

menggunakan data matriks (Santra, 2012). Table berikut memperlihatkan *Confusion Matrix* yang diklasifikasikan untuk 2 kelas.

Tabel 2.1 *Confusion Matrix*

	Predicted : No	Predicted : Yes
Actual : No	TN	FP
Actual : Yes	FN	TP

Berikut pengertian dari table di atas:

1. TN adalah nilai benar dari prediksi dengan hasil negatif.
2. FP adalah nilai salah dari prediksi dengan hasil positif.
3. FN adalah nilai salah dari prediksi dengan hasil negatif.
4. TP adalah nilai benar dari prediksi dengan hasil positif.

Beberapa term standar telah didefinisikan untuk menghitung akurasi, *True Positive Rate dan True Negative Rate*. Akurasi digunakan untuk menghitung nilai prediksi yang benar. *True Positive Rate* digunakan untuk menghitung nilai prediksi positif yang benar. *True Negative Rate* digunakan untuk menghitung nilai prediksi negative yang benar (Santra, 2012). Proses perhitungan menggunakan persamaan berikut.

1. Menghitung akurasi sesuai persamaan 2.1.

$$AC = \frac{\mathrm{TN+TP}}{\mathrm{TN+FP+FN+TP}} \quad (2.4)$$

2. Menghitung *True Positive Rate* sesuai persamaan 2.2.

$$TPR = \frac{\mathrm{TP}}{\mathrm{FN+TP}} \quad (2.5)$$

3. Menghitung *True Negative Rate* sesuai persamaan 2.3.

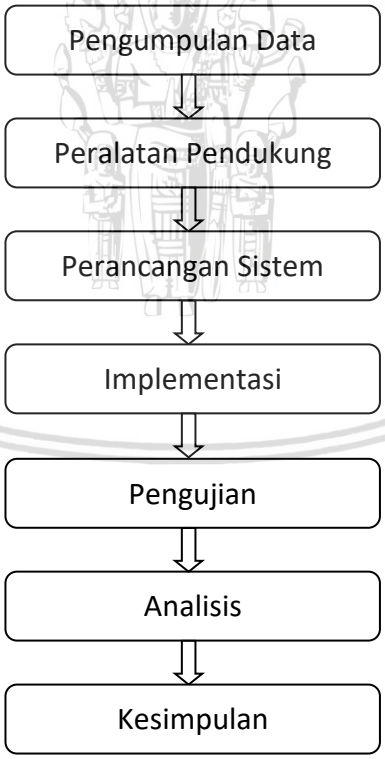
$$TNR = \frac{\mathrm{TN}}{\mathrm{TN+FP}} \quad (2.6)$$

BAB 3
METODOLOGI

Bab metodologi menguraikan langkah-langkah yang ditempuh dalam proses penerapan algoritme *Latent Semantic Indexing* (LSI) yang dipakai dalam penelitian. Metodelogi penelitian menjelaskan mengenai penerapan metode-metode yang akan digunakan serta rancangan dari proses penerapan algoritme yang digunakan.

3.1 Metodologi Penelitian

Subbab ini menguraikan proses yang akan dilakukan dalam penelitian. pengerjaan sistem analisis sentiment ulasan video animasi menggunakan metode *Latent Semantic Indexing* ini dlakukan dengan beberapa tahap, yakni Pengumpulan Data, Peralatan Pendukung, Perancangan Sistem, Implementasi, Pengujian, Analisis dan Kesimpulan.



Gambar 3.1 Diagram Metodologi Penelitian

3.2 Pengumpulan Data

Penelitian ini dilakukan menggunakan data yang diambil adalah data ulasan dari *website myanimelist.com* yang terdiri dari 250 dataset, 50 dataset sebagai data uji yang terdiri dari 25 ulasan positif dan 25 ulasan negatif, 200 dataset sebagai data latih yang terdiri dari 100 ulasan positif dan 100 ulasan negatif. dataset berisi kalimat ulasan seseorang dari *website myanimelist.com* sehingga diperlukan proses *preprocessing* terlebih dahulu menggunakan pemrosesan teks. Setelah proses preprocessing barulah bias dilanjutkan untuk di analisis sentiment dari tiap data yang diambil menggunakan metode *Latent Semantic Indexing*.

3.3 Peralatan Pendukung

Peralatan Pendukung pada penelitian ini dilakukan untuk menentukan pendukung kebutuhan yang nantinya akan digunakan dalam pembuatan sistem analisis sentimen ulasan video animasi menggunakan metode *Latent Semantic Indexing*. Peralatan pendukung pada sistem ini adalah:

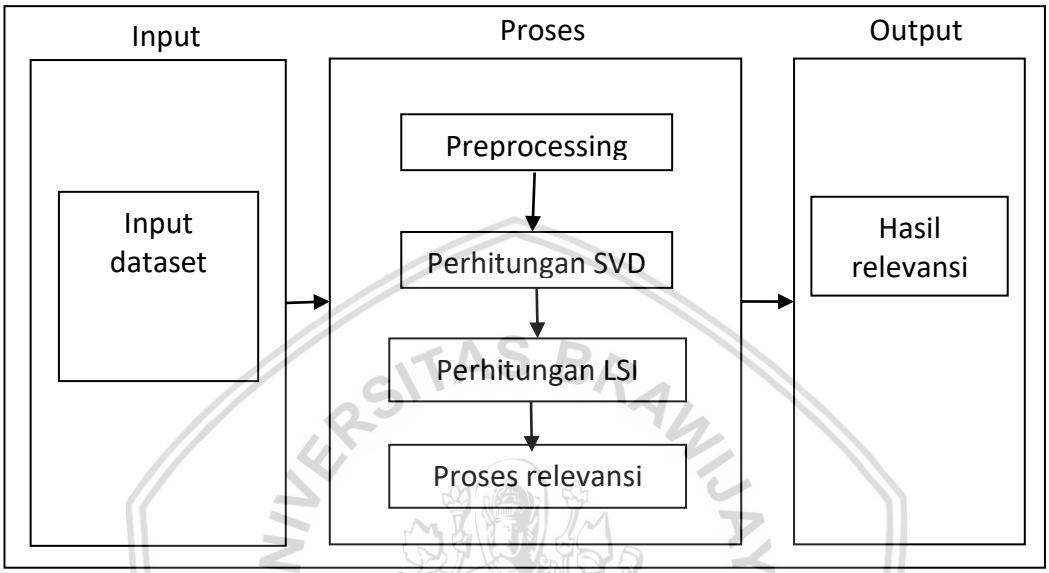
1. Kebutuhan perangkat keras dengan spesifikasi minimal:
   a. Intel Core i3 generasi 4 atau diatasnya
   b. RAM 2 GB
   c. Ruang pada harddisk yang kosong minimal 200 MB
   d. Monitor
   e. Keyboard
   f. Mouse
2. Kebutuhan perangkat lunak:
   a. Sistem operasi Windows 10
   b. Java Development Kit
   c. Netbeans IDE

3.4 Perancangan Sistem

Perancangan sistem bertujuan untuk memudahkan ketika dilakukan proses implementasi dan pengujian. Perancangan sistem menghasilkan rancangan tentang langkah kerja dari sistem yang dari proses awal sampai akhir sesuai dengan arsitektur sistem. Perancangan yang dihasilkan berupa model Perancangan dan arsitektur dari sistem *Latent Semantic Indexing*.

3.4.1 Model Perancangan

Model perancangan sistem yang menjelaskan cara kerja sistem secara terstruktur. Model ini memperlihatkan proses dari awal yang berupa masukan data sampai proses akhir yang berupa keluaran hasil relevansi. Dapat dilihat pada Gambar 3.2.



Gambar 3.2 Diagram Model Perancangan

User memasukkan data ulasan video animasi ke dalam sistem selanjutnya data yang dimasukkan akan diproses oleh sistem. Melalui proses *pre-processing*, *Perhitungan SVD*, *Perhitungan LSI,* dan *proses relevansi* menggunakan metode *cosine similiarity* lalu hasil relevansi yang dapat dicek untuk digolongkan ke kelas positif atau kelas negatif.

3.5 Implementasi

Implementasi adalah tahapan pembangunan sistem, pada tahap implementasi semua hal yang telah di rancang akan diimplementasikan sesuai dengan perancangan tersebut. Berikut tahapan-tahapan implementasi dari sistem yang akan dibuat.

1. Implementasi antarmuka pengguna.
2. Implementasi basis data untuk proses pengolahan dan penyimpanan data.
3. Implementasi program dengan metode *Latent Semantic Indexing* menggunakan bahasa pemrograman JAVA.
4. Hasil sistem ini akan menghasilkan analisis sentiment ulasan yang digolongkan ke ulasan positif atau ulasan negatif.

3.6 Pengujian

Pengujian dilakukan untuk melihat apakah sistem yang dibuat sudah berjalan sesuai dengan diinginkan. Pengujian dilakukan dalam 2 tahapan:

1. Pengujian pertama yaitu pengujian *true positive rate* , *true negative rate* dan akurasi.
2. Pengujian kedua yaitu pengujian nilai *k-rank* yang bertujuan untuk mengetahui nilai *k-rank* yang menghasilkan akurasi paling optimal.

Impelementasi sistem ini akan menghasilkan analisis sentimen ulasan yang digolongkan ke ulasan positif atau ulasan negatif.

3.7 Analisis

Tahapan analisis pada penelitian ini yaitu hasil dari keluaran sistem yang telah di uji hasil akurasi, *true positive rate* dan *true negative rate*. Dibandingkan antara hasil uji tersebut dengan beberapa percobaan nilai *k-rank*. Kemudian di ambil kesimpulan dari hasil analisis tersebut.

3.8 Kesimpulan

Kesimpulan didapatkan setelah semua tahapan selesai, tahapan yang harus dilalui dimulai dari tahapan perancangan, imeplementasi, dan pengujian. kemudian didapatkan hasil analisis dari sistem yang dibuat dan dari hasil analisis tersebut di buat kesimpulan. Dari kesimpulan yang didapat di berikan saran untuk pengembangan kedepannya. Saran bertujuan agar kesalahan yang terjadi pada penilitian ini dapat diperbaiki dan agar penelitian ini dapat dikembangkan menjadi lebih baik.

BAB 4
PERANCANGAN

4.1 Perancangan proses

Pada perancangan menjelaskan tahapan yang harus dilalui sesuai dengan rancangan sistem. Proses ini dimulai dari *preprocessing* yang terdiri dari *case folding, tokenisasi, filtering*, dan proses *stemming.* Setelah didapat hasil stemming kata-kata tersebut kemudia di proses untuk diubah menjadi suatu nilai numeric menggunakan metode TF-IDF. Dan dilanjutkan proses utama yaitu proses LSI untuk menghitung nilai *query* yang akan dihitung untuk di cari relevansi antar dokumennya menggunakan *cosine similiarity*. Untuk menghitung nilai *query* pada proses LSI dibutuh nilai dari Matriks *U* dan matriks *S* yang didapat dari proses SVD. setelah mendapat nilai *query*, *query* tersebut dihitung pada proses *cosine similiarity* yang akan menghasilkan relevansi dokumen, dokumen tersebut nantinya akan digolongkan ke dokumen kelas positif atau dokumen kelas negatif.

4.2 Proses Pelatihan

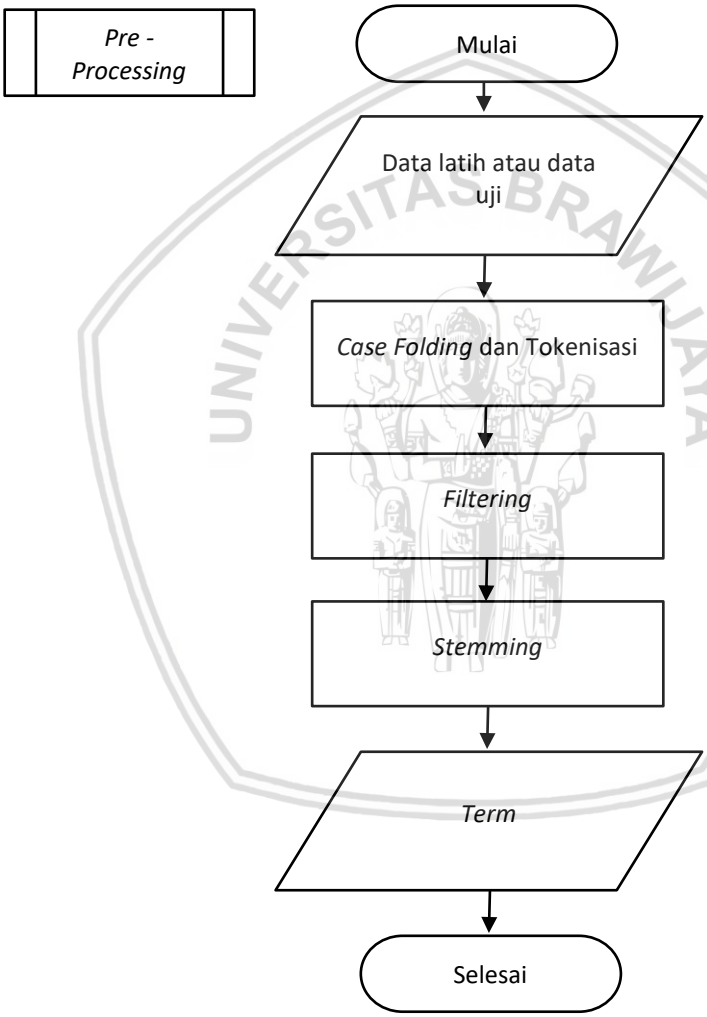
Pada proses pelatihan yaitu dilakukan pelatihan data yang nantinya digunakan untuk relevansi berdasarkan kecocokan antar dokumen. Proses pelatihan dilakukan menggunakan data latih berjumlah 100 data. Proses pelatihan akan digunakan sebagai perbandingan relevansi dokumen yang akan di bandingkan oleh data uji. Setelah dibandingkan dan dicocokkan antar dokumen lalu hasil tersebut di uji untuk mencari nilai akurasi dari sistem yang dibuat.

4.2.1 Proses *Pre-Processing*

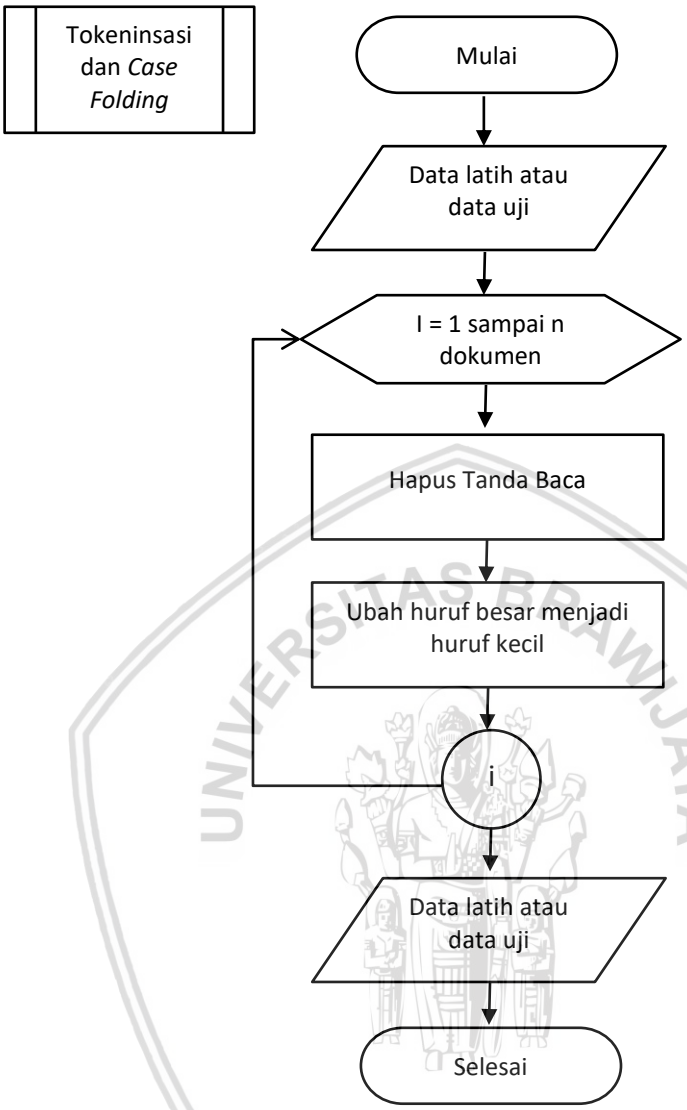
Pada proses ini data awal yang berupa teks diubah menjadi angka. *Pre-processing* diproses menggunakan metode *text mining* dimana dilakukan pengolahan teks yang membuat teks menjadi lebih terstruktur dimulai dengan proses *case folding* yang digunakan untuk mengubah huruf besar menjadi huruf kecil, untuk penjelasan lebih lengkap dapat dilihat pada Bab 2*,* lalu dilanjutkan dengan proses *tokenisasi* yang digunakan untuk menghilangkan tanda baca, untuk penjelasan lebih lengkap dapat dilihat pada Bab 2*,* selanjutnya *filtering* yaitu proses untuk membuang kata yang tidak relevan, penjelasan lebih lengkap data dilihat pada Bab 2 dan terakhir proses *stemming* yang merubah kata menjadi kata dasar dengan menghilangkan awalan kata, akhiran dan sisipan kata, untuk penjelasan lebih lengkap dapat di lihat pada Bab 2. Diagram alur *Pre-Processing* dapat dilihat pada Gambar 4.1.

4.2.1.1 Proses Tokenisasi dan Case Folding

Tokenisasi merupakan proses untuk memcah teks bebas yang besar menjadi bagian-bagian yang disebut kalimat. Dari proses tokeninsasi dilanjutkan dengan Proses *casefolding* yaitu memotong setiap kata dari bagian-bagian kalimat yang didapat dari proses sebelumnya yaitu pada proses pada *tokenisasi*. Selanjutnya mengubah semua huruf dalam dokumen menjadi huruf kecil dan menghilangkan karater selain huruf. Alur proses dari tokenisasi dan *case folding* dapat dilihat pada Gambar 4.2.



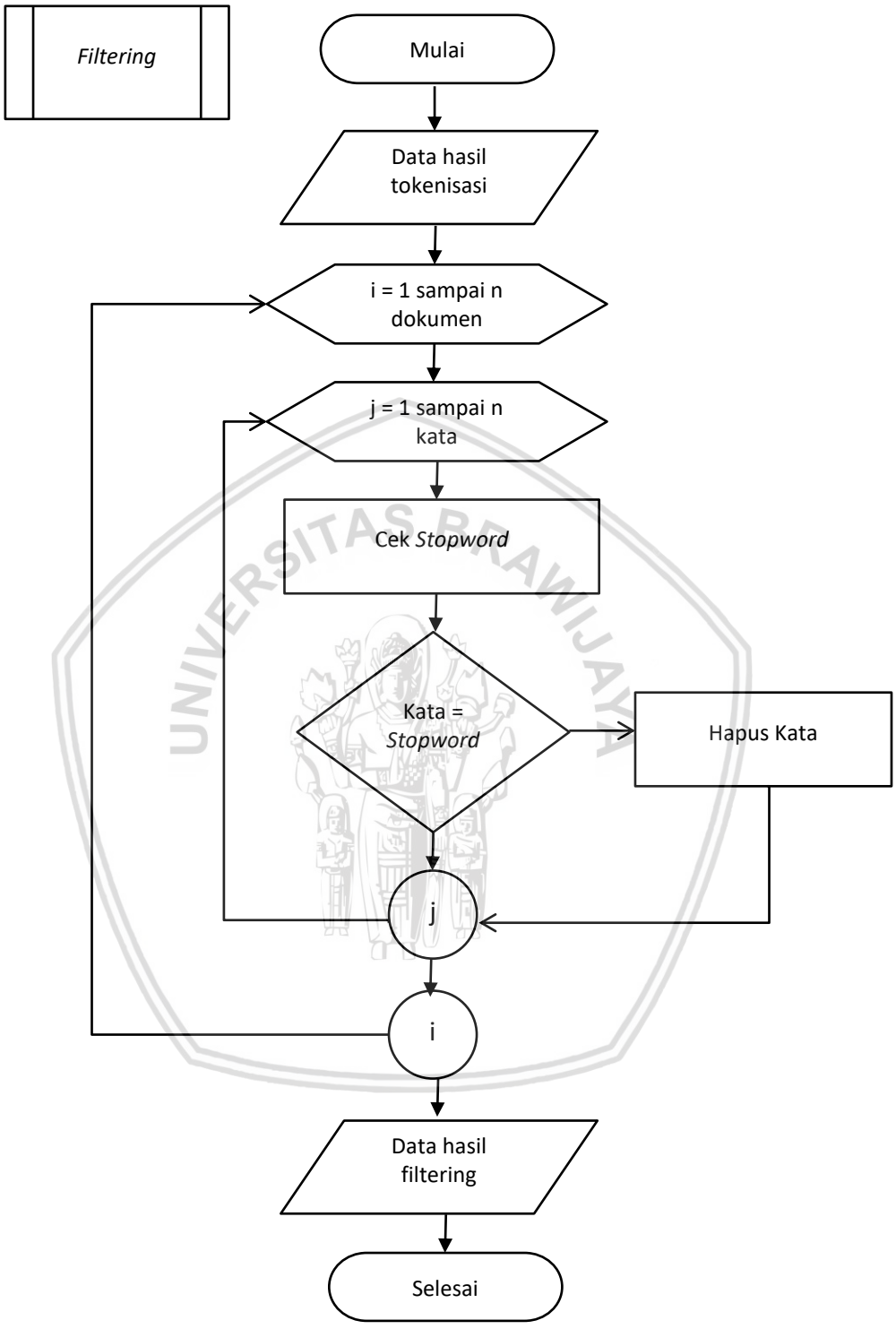
Gambar 4.1 Diagram Alur *Pre-Processing*



Gambar 4.2 Diagram Alur Tokenisasi dan *Case Folding*

4.2.1.2 Proses Filtering

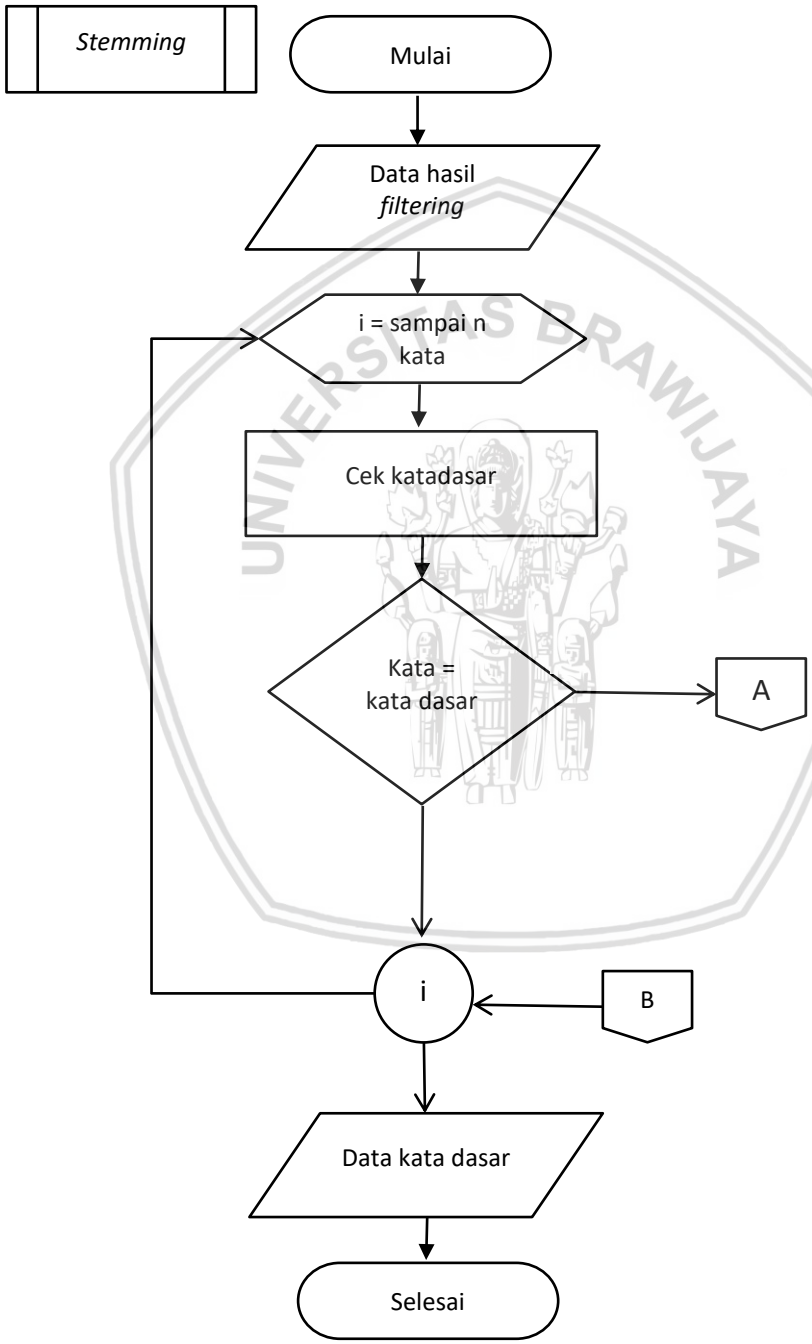
Filtering adalah proses untuk menghilangkan kata-kata yang dianggap tidak perlu atau tidak penting. Pada tahap ini kata yang dianggap tidak diperlukan akan dihilangkan dengan cara mengecek kata tersebut dengan sebuah *stopword list*.. Apabila pada dokumen yang diproses terdapat kata yang juga terdapat pada kata dalam *stopword list* maka kata tersebut akan di hilangkan. Alur proses *filtering* ditunjukkan oleh Gambar 4.4.

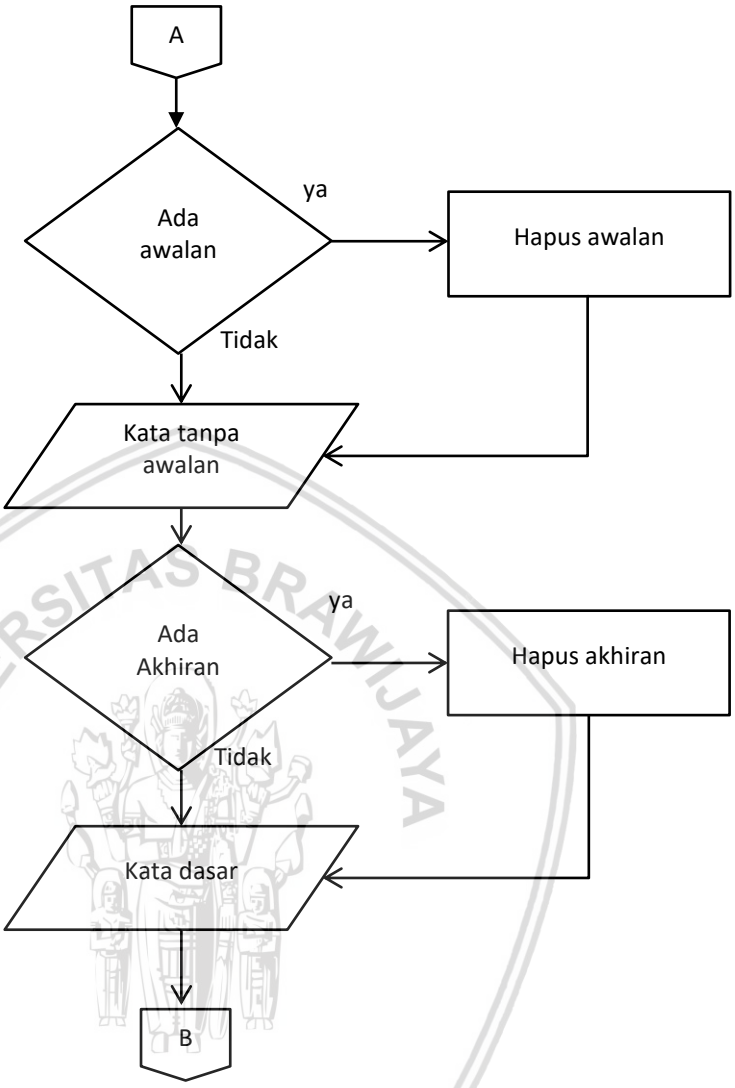


Gambar 4.3 Diagram Alur *Filtering*

4.2.1.3 Proses Stemming

Stemming adalah proses mengubah kata mejadi kata dasar. Pada proses ini setelah data diproses melalui *filtering* yang telah menjadi huruf kecil selanjutnya akan dilakukan proses *stemming* yaitu proses menghapus suatu imbuhan dari kata baik itu awalan maupun akhiran. Algoritma *stemming* yang digunakan adalah Porter *stemmer*. Alur proses dapat dilihat pada Gambar 4.4.



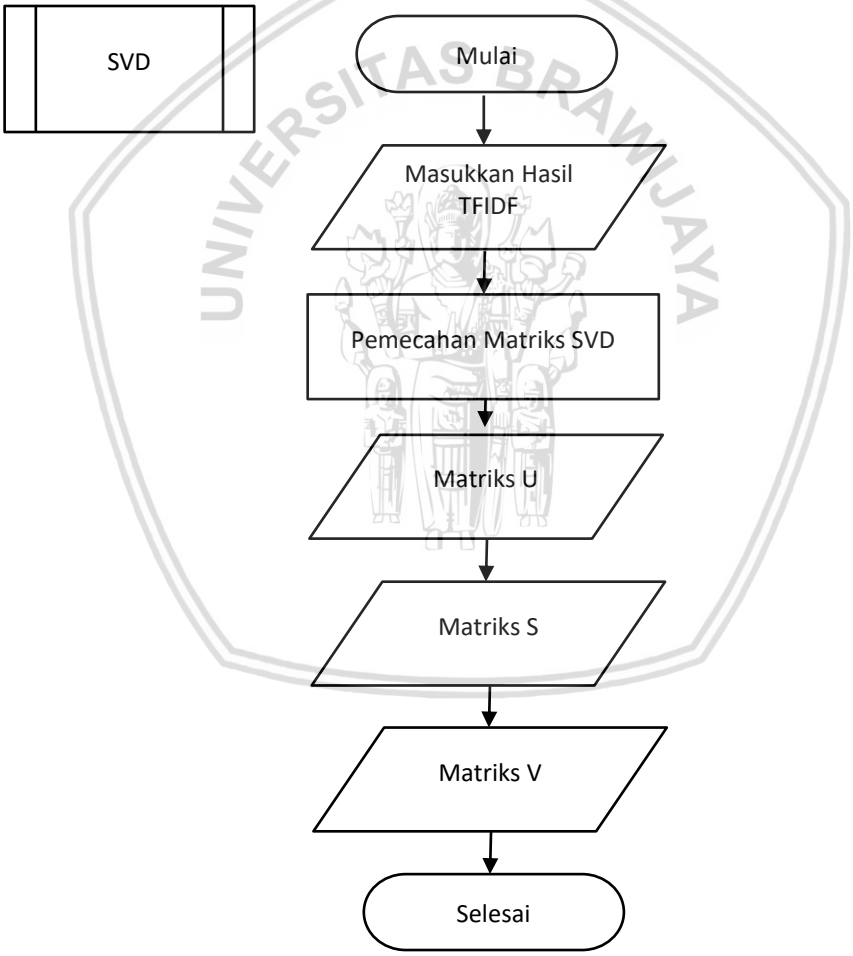


Gambar 4.4 Diagram Alur *Stemming*

Alur proses *stemming* dimulai dari data hasil *filtering* yang sudah dipisah tiap kata, dihilangkan tanda baca, di ubah menjadi huruf kecil dan telah di hilangkan kata kata yang tidak penting ini digunakan sebagai masukkan untuk selanjutnya diproses dengan tujuan untuk membuat kata tersebut menjadi kata dasar. Proses pertama yaitu memeriksa kata dasar apakah kata yang didapat dari masukkan sudah termasuk kata dasar, jika belum maka akan dilanjutkan proses untuk memeriksa awalan jika ada awalan maka awalan tersebut dihapus. Proses selanjutnya yaitu memeriksa akhiran jika ada akhiran maka akhiran tersebut dihapus. Setelah awal dan akhir dihapus jadilah kata dasar yang nantinya akan digunakan untuk pembobotan *TF-IDF*.

4.2.2 Proses SVD

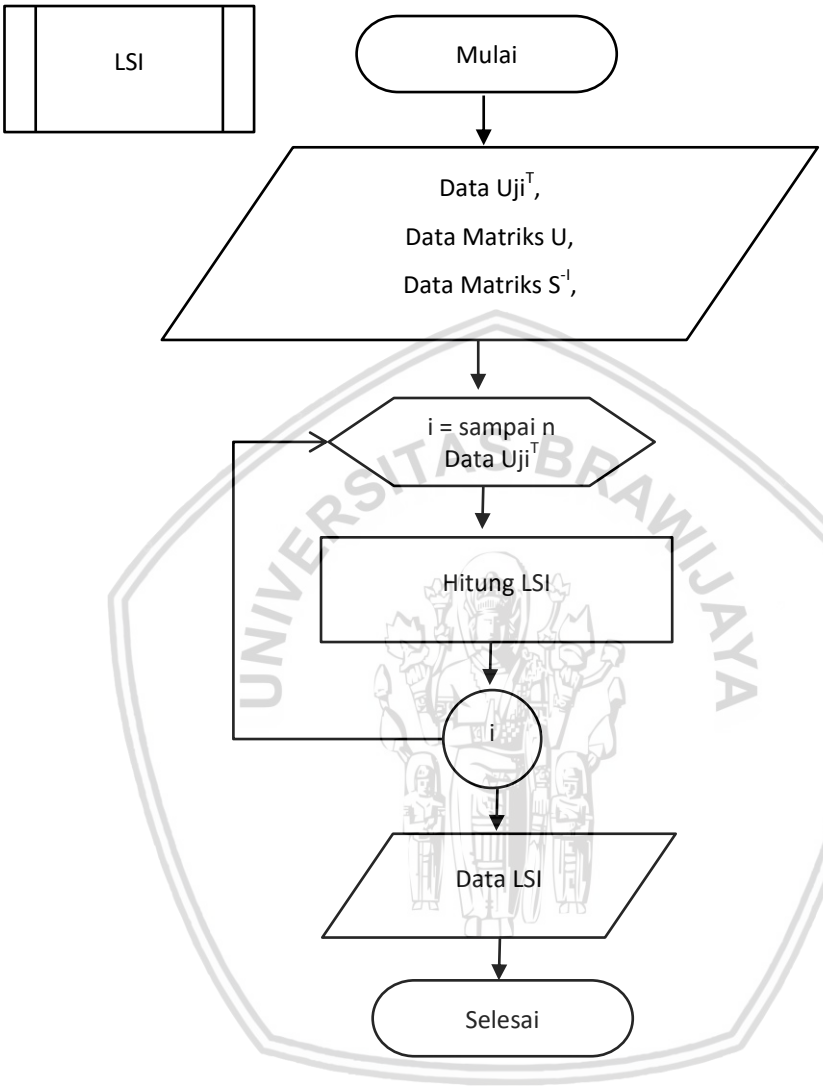
Proses SVD adalah proses pemecahan matriks hasil normalisasi menjadi 3 buah matriks yaitu matriks USV. 3 buah matriks terbut akan digunakan untuk proses selanjutnya yaitu pada proses *Latent Semantic Indexing* dan *Cosine similiarity*. Pada proses LSI matriks yang digunakan adalah matriks *U* dan matriks *S* yang sudah di*inverse*kan. Kedua matriks tersebut akan digunakan untuk menghitung *query* input yang berada pada proses LSI dengan cara *query* input dikalikan dengan nilai hasil kali dari matriks *U* dan matriks *S inverse*. Sedangkan untuk matriks *V* akan digunakan pada proses *Cosine similiarity* dengan cara menalikan nilai *query* hasil dari proses LSI ke matriks *V.* Diagram alur proses SVD dapat dilihat pada Gambar 4.5



Gambar 4.5 Diagram Alur Proses SVD

4.2.2.1 Proses LSI

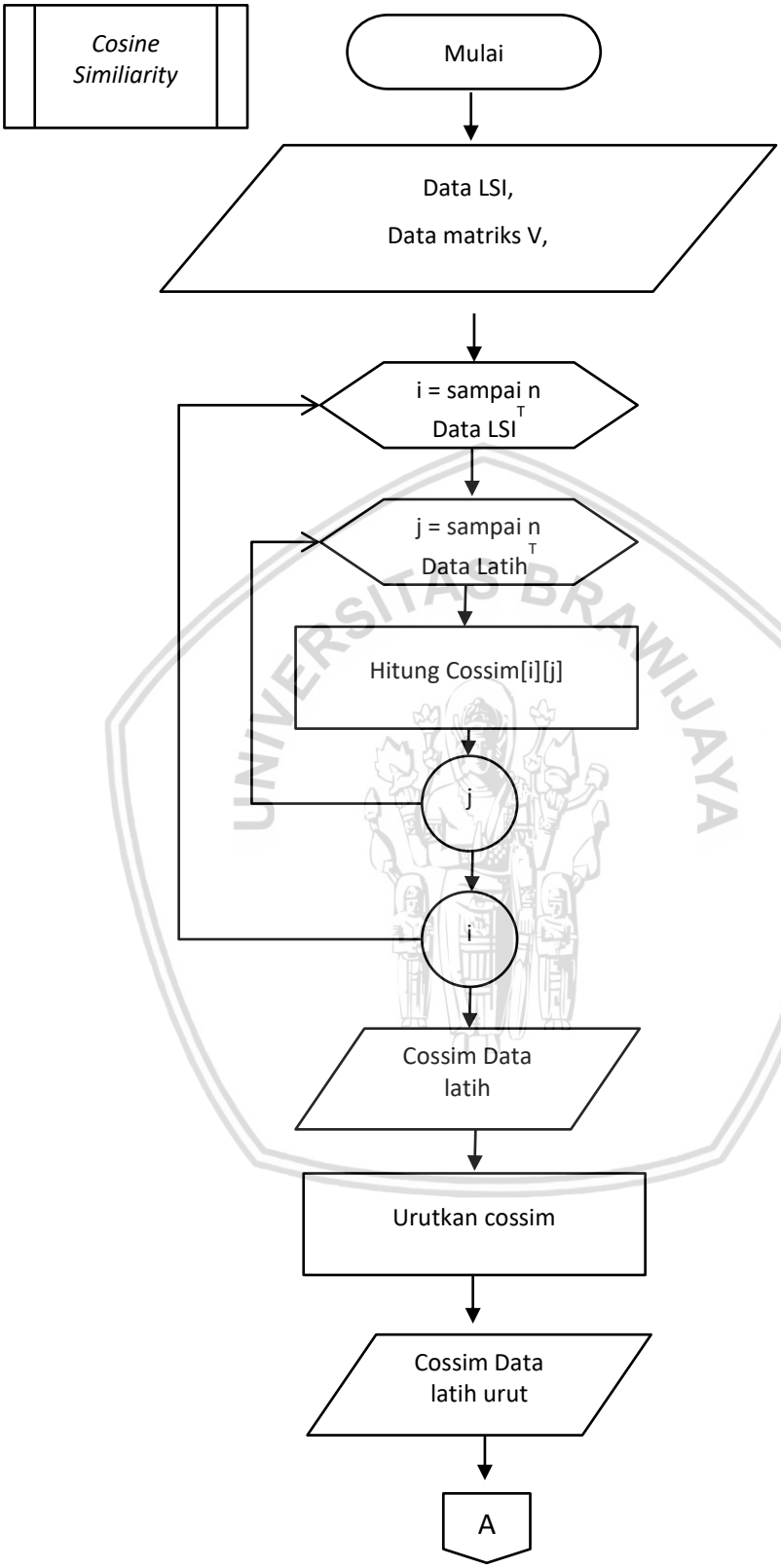
Proses LSI yaitu proses perhitungan data uji ke data matriks *U* dan matriks S^{-1} yang didapat dari hasil dekomposisi matriks pada proses SVD.

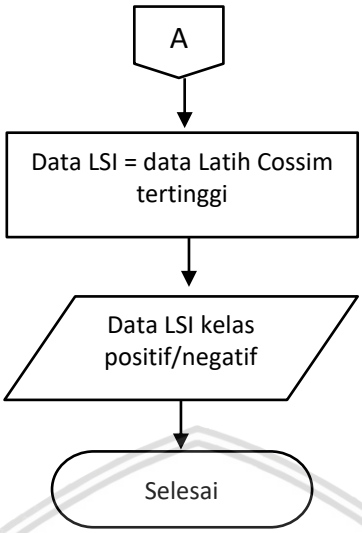


Gambar 4.6 Diagram Alur LSI

4.2.2.2 Proses Cosine Similiarity

Proses menghitung kemiripan dari dua buah vektor yaitu vektor data latih yang telah diproses pada LSI dan nilai Matriks *V* dari proses SVD. Lalu mengurutkan hasil perhitungan. Setelah itu dicek hasil urutan teratas termasuk dokumen kelas positif atau kelas negatif.


Cosine Similiarity
Mulai
Data LSI,
Data matriks V,
i = sampai n
Data LSI^T
j = sampai n
Data Latih^T
Hitung Cossim[i][j]
j
i
Cossim Data latih
Urutkan cossim
Cossim Data latih urut
A



Gambar 4.7 Diagram Alur *Cosine Similiarity*

4.3 Perhitungan manual

4.3.1 Perhitungan TF-IDF

Berikut data awal sebelum diproses. Data awal sebelum diproses dapat dilihat pada Tabel 4.1 berikut:

Tabel 4.1 data awal sebelum diproses

Data		Sentimen
data 1	good good well	Positif
data 2	well well good good enjoy enjoy	Positif
data 3	enjoy amazing amazing amazing	Positif
data 4	bad worst worst worse worse worse	Negatif
data 5	dissatisfying dissatisfying dissatisfying dissatisfying worst	Negatif
Query	good good well enjoy enjoy enjoy	?

Proses selanjutnya adalah proses *stemming* dengan menggunakan metode porter *stemmer*. Proses *stemming* disini mencakup proses *case folding* , dilanjutkan dengan proses tokenisasi, proses *filtering* dan terakhir *stemming*. Hasil *stemming* dapat dilihat pada Table 4.2.

Tabel 4.2 Data ulasan yang telah melalui proses *stemming*

data		Sentimen
data 1	good good well	Positif
data 2	well well good good enjoi enjoi	Positif
data 3	enjoi amaz amaz amaz	Positif
data 4	bad worst worst wors wors wors	Negatif
data 5	dissatisfying dissatisfi dissatisfi dissatisfi worst	Negatif
Query	good good well enjoi enjoi enjoi	?

Setelah didapat hasil *stemming* maka proses selanjutnya adalah melakukan pembobotan frekuensi tiap term dan dari hasil pembobotan tersebut di cari nilai *DF* dengan menghitung frekuensi kata pada dokumen. Hasil pembobotan dan nilai DF dapat dilihat pada Tabel 4.3

Tabel 4.3 Frekuensi Kemunculan Term Tiap Dokumen

term	D1	D2	D3	D4	D5	Query	DF
Enjoi	0	2	1	0	0	3	3
Good	2	2	0	0	0	2	3
Bad	0	0	0	1	0	0	1
Amaz	0	0	3	0	0	0	1
Well	1	2	0	0	0	1	3
Wors	0	0	0	2	1	0	2
Dissatisfy	0	0	0	0	4	0	1
wors	0	0	0	3	0	0	1

Dilanjutkan dengan mencari nilai *WTF* dan *IDF*. Nilai *Wtf* yang didapat dari perhitungan dari LOG 10 nilai *Tf*, sedangkan untuk IDF didapatkan dari LOG 10 (jumlah term/*Df*). Hasil dari Wtf dan Idf dapat dlihat pada Tabel 4.4

Tabel 4.4 perhitungan nilai WTF dan IDF

term	D1	D2	D3	D4	D5	Query	IDF
Enjoi	0	1,30	1	0	0	1,48	0,30
Good	1,30	1,30	0	0	0	1,30	0,30
Bad	0	0	0	1	0	0	0,78
Amaz	0	0	1,48	0	0	0	0,78
Well	1	1,30	0	0	0	1	0,30
Wors	0	0	0	1,30	1	0	0,48

term	D1	D2	D3	D4	D5	Query	IDF
Dissatisfy	0	0	0	0	1,60	0	0,78
wors	0	0	0	1,48	0	0	0,78

Dilanjutkan dengan perhitungan *Wtd* dengan cara mengalikan nilai *Wtf * Idf*. Hasil Wtd dapat dilihat pada Tabel 4.5

Tabel 4.5 perhitugan nilai WTD sebelum dinormalisasi

term	D1	D2	D3	D4	D5	Query
Enjoi	0	0,39	0,30	0	0	0,44
Good	0,39	0,39	0	0	0	0,39
Bad	0	0	0	0,78	0	0
Amaz	0	0	1,15	0	0	0
Well	0,30	0,39	0	0	0	0,30
Wors	0	0	0	0.62	0.48	0
Dissatisfy	0	0	0	0	1.25	0
wors	0	0	0	1.15	0	0

Nilai *Wtd* yang didapat masih harus diteruskan dengan proses normalisasi agar nilai yang didapat terstruktur dengan baik. Hasil normalisasi dari *Wtd* dapat dilihat pada Tablel 4.6.

Tabel 4.6 Perhitungan WTD setelah dinormalisasi

term	D1	D2	D3	D4	D5	Query
Enjoi	0	0,58	0,25	0	0	0,67
Good	0,79	0,58	0	0	0	0,59
Bad	0	0	0	0,51	0	0
Amaz	0	0	0,97	0	0	0
Well	0,61	0,58	0	0	0	0,45
Wors	0	0	0	0,41	0,36	0
Dissatisfy	0	0	0	0	0,93	0
wors	0	0	0	0,76	0	0

Proses selanjutnya mentranspose Matriks hasil normalisasi disini didefinisikan dengan Huruf i yang di transpose menjadi A^T yang ditunjukkan pada Tabel 4.7.

4.3.2 Perhitungan pemecahan matriks SVD

Tabel 4.7 Matriks A^T

Doc	T1	T2	T3	T4	T5	T6	T7	T8
D1	0	0,79	0	0	0,61	0	0	0
D2	0,58	0,58	0	0	0,58	0	0	0
D3	0,25	0	0	0,97	0	0	0	0
D4	0	0	0,51	0	0	0,41	0	0,76
D5	0	0	0	0	0	0,36	0,93	0

Matriks A^T ditujunjukkan dengan kolom menyatakan *term* dan baris menyatakan dokumen. Selanjutnya dilakukan perhitungan $A^T.A$ yang ditunjukan pada Table 4.8.

Tabel 4.8 Matriks $A^T.A$

0,996	0,812	0	0	0
0,812	1,009	0,145	0	0
0	0,145	1,003	0	0
0	0	0	1,006	0,148
0	0	0	0,148	0,995

Mencari nilai *eigenvalue* dari perkalian matriks $A^T.A$.

$$\begin{bmatrix} 0{,}996 & 0{,}812 & 0 & 0 & 0 \\ 0{,}812 & 1{,}009 & 0{,}145 & 0 & 0 \\ 0 & 0{,}145 & 1{,}003 & 0 & 0 \\ 0 & 0 & 0 & 1{,}006 & 0{,}148 \\ 0 & 0 & 0 & 0{,}148 & 0{,}995 \end{bmatrix} - c \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 0{,}996 & 0{,}812 & 0 & 0 & 0 \\ 0{,}812 & 1{,}009 & 0{,}145 & 0 & 0 \\ 0 & 0{,}145 & 1{,}003 & 0 & 0 \\ 0 & 0 & 0 & 1{,}006 & 0{,}148 \\ 0 & 0 & 0 & 0{,}148 & 0{,}995 \end{bmatrix} - \begin{bmatrix} c & 0 & 0 & 0 & 0 \\ 0 & c & 0 & 0 & 0 \\ 0 & 0 & c & 0 & 0 \\ 0 & 0 & 0 & c & 0 \\ 0 & 0 & 0 & 0 & c \end{bmatrix}$$

$$\begin{bmatrix} 0{,}996 - c & 0{,}812 & 0 & 0 & 0 \\ 0{,}812 & 1{,}009 - c & 0{,}145 & 0 & 0 \\ 0 & 0{,}145 & 1{,}003 - c & 0 & 0 \\ 0 & 0 & 0 & 1{,}006 - c & 0{,}148 \\ 0 & 0 & 0 & 0{,}148 & 0{,}995 - c \end{bmatrix} = 0$$

Perhitungan nilai *eigenvalue* dan nilai *eigenvector* dihitung dengan bantuan kalkulator matriks (http://www.bluebit.gr/matrix-calculator/). Didapatkan nilai *eigen* dan *singular* adalah sebagai berikut:

Tabel 4.9 Nilai *eigen* dan nilai *singular*

	Eigenvalue	Singular value
c1	1,827	1,352
c2	0,178	0,422
c3	1,003	1,002
c4	1,149	1,071
c5	0,852	0,923

Nilai *singular* didapatkan dari hasil akar nilai *Eigen*. Lalu matriks *S* dibentuk dari nilai *singular* yang telah diurutkan dari nilai yang terbesar ke yang terkecil membentuk matriks diagonal. Matriks *S* terdapat pada Tabel 4.10.

Tabel 4.10 Matriks *S*

1,352	0	0	0	0
0	1,071	0	0	0
0	0	1,002	0	0
0	0	0	0,923	0
0	0	0	0	0,422

Matriks S^{-1} digunakan untuk perhitungan pada saat proses *LSI*.

Tabel 4.11 Matriks S^{-i}

0,740	0	0	0	0
0	0,934	0	0	0
0	0	0,998	0	0
0	0	0	1,083	0
0	0	0	0	2,370

Selanjutnya untuk membangun matriks *V* digunakan dari *Eigenvector* pada masing-masing nilai *Eigen* yang telah didapatkan. Terdapat lima nilai *Eigen* sesuai dengan jumlah dokumen.

Untuk c1 = 1,827

$$\begin{bmatrix} 0{,}996-1{,}827 & 0{,}812 & 0 & 0 & 0 \\ 0{,}812 & 1{,}009-1{,}827 & 0{,}145 & 0 & 0 \\ 0 & 0{,}145 & 1{,}003-1{,}827 & 0 & 0 \\ 0 & 0 & 0 & 1{,}006-1{,}827 & 0{,}148 \\ 0 & 0 & 0 & 0{,}148 & 0{,}995-1{,}827 \end{bmatrix}$$

$$\begin{bmatrix} -0{,}831 & 0{,}812 & 0 & 0 & 0 \\ 0{,}812 & -0{,}818 & 0{,}145 & 0 & 0 \\ 0 & 0{,}145 & -0{,}824 & 0 & 0 \\ 0 & 0 & 0 & -0{,}821 & 0{,}148 \\ 0 & 0 & 0 & 0{,}148 & -0{,}832 \end{bmatrix}$$

$$\begin{bmatrix} -0{,}831 & 0{,}812 & 0 & 0 & 0 \\ 0{,}812 & -0{,}818 & 0{,}145 & 0 & 0 \\ 0 & 0{,}145 & -0{,}824 & 0 & 0 \\ 0 & 0 & 0 & -0{,}821 & 0{,}148 \\ 0 & 0 & 0 & 0{,}148 & -0{,}832 \end{bmatrix} \begin{bmatrix} x1 \\ x2 \\ x3 \\ x4 \\ x5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Sehingga diperoleh,

-0,831 $x1$ + 0,812 $x2$ + 0 $x3$ + 0 $x4$ + 0 $x5$ = 0

0,812 $x1$ + -0,818 $x2$ + 0,145 $x3$ + 0 $x4$ + 0 $x5$ = 0

0 $x1$ + 0,145 $x2$ + -0,824 $x3$ + 0 $x4$ + 0 $x5$ = 0

0 $x1$ + 0 $x2$ + 0 $x3$ + -0,821 $x4$ + 0,148 $x5$ = 0

0 $x1$ + 0 $x2$ + 0 $x3$ + 0,148 $x4$ + -0,832 $x5$ = 0

Jadi matriks untuk nilai eigen vector dengan nilai Eigen c1 = 1,827 adalah

$$\begin{bmatrix} x1 \\ x2 \\ x3 \\ x4 \\ x5 \end{bmatrix} = \begin{bmatrix} -0{,}693 \\ -0{,}710 \\ -0{,}125 \\ 0 \\ 0 \end{bmatrix}$$

Perhitungan *eigenvecto*r berlanjut sampai dengan c5 = 0,852, sehingga akan didapatkan matriks yang berisi semua nilai *Eigenvector* yang telah digabungkan menjadi matriks *V*. Tabel 4.12 menunjukan hasil perhitungan *Eigenvector* mulai dari c1 = 1,827 sampai dengan c5 = 0,852.

Tabel 4.12 Matriks *V*

c1 = 1,827	c2 = 0,178	c3 = 1,003	c4 = 1.149	c5 = 0.852
-0,693	0	-0,176	0	-0,699
-0,710	0	-0,002	0	0,704
-0,125	0	0,984	0	-0,124
0	0,721	0	-0,693	0
0	0,693	0	0,721	0

Selanjutnya membuat matriks *U*, hasil kali matriks *U* terdapat pada Tabel 4.13. Matriks *U* ini merupakan hasil nilai *Eigenvector* dari nilai *Eigenvalue*.

4.13 Matriks *U*

-0,328	0	0,245	0	0,895
-0,710	0	-0,140	0	-0,341

0	0,343	0	-0,383	0
-0,090	0	0,953	0	-0,285
-0,617	0	-0,108	0	-0,042
0	0,509	0	-0,027	0
0	0,602	0	0,726	0
0	0,511	0	-0,571	0

Untuk menyesuaikan hasil dekomposisi matriks *A*, maka dilakukan pengencekan melalui perkalian 3 buah matriks USV^T . Tabel 4.14 menunjukkan hasil transpose dari matriks *V*.

Tabel 4.14 Matriks V^T

-0,693	-0,710	-0,125	0	0
0	0	0	0,721	0,693
-0,176	-0,002	0,984	0	0,000
0	0	0	-0,693	0,721
-0,699	0,704	-0,124	0	0

4.3.3 Proses LSI

Penggunaan SVD digunakan dalam LSI. Reduksi dimensi matriks ditentukan dengan nilai *k* = 3. Untuk nilai *k* = 3, maka dari matriks *U* akan diambil 4 dimensi untuk dimensi kolom yang ditunjukkan pada Table 4.15

Tabel 4.15 Matriks *U* dengan *k* = 3

-0,328	0	0,245
-0,710	0	-0,140
0	0,343	0
-0,090	0	0,953
-0,617	0	-0,108
0	0,509	0
0	0,602	0
0	0,511	0

Selanjutnya perhitungan *query vector* dari SVD, sehingga akan didapatkan masing-masing dari q^T, *U* dan S^{-1} kemudian mengalikan ketiga matriks tersebut. Untuk matriks S^{-1} dengan *k* = 3 akan dimabil 3 dimensi untuk dimensi baris dan kolom yang ditujukkan pada Table 4.16

Tabel 4.16 Matriks S^{-i} dengan $k = 3$

0,740	0	0
0	0,934	0
0	0	0,998

Nilai q^T didapatkan dari hasil nilai normalisasi untuk dokumen *query* yang telah di *transpose*. Table 4.17 menunjukan nilai q^T.

Tabel 4.17 Matriks q^T

0,670	0,590	0	0	0,450	0	0	0

Nilai q^i dengan $k = 4$

Tabel 4.17 Matriks $q' = q^T U_k.S_k^{-i}$

-0,678	0	0,033

Setelah mendapat nilai *q'*, maka mencari relevanasi dari *query* tersebut dengan menggunakan *cosine similiarty*. Nilai $q = q'$ dan $d = V$ dari dekomposisi matriks SVD.

Tabel 4.18 Hasil Pembilang dari *cosine similiarity*

q			Dok	D			q.d			∑(q.d)
-0,678	0	0,033	D1	-0,693	0	-0,176	0,469	0	-0,005	0,464
			D2	-0,710	0	-0,002	0,481	0	-6,6E-05	0,481
			D3	-0,125	0	0,984	0,084	0	0,0324	0,117
			D4	0	0,721	0	0	0	0	0
			D5	0	0,693	0	0	0	0	0

Tabel 4.19 Hasil Penyebut dari *cosine similiarity*

q^2			Dok	d^2			∑ (q^2)	∑ d^2	∑ (q^2)* ∑ (d^2)	|q||d|
-0,678	0	0,033	D1	0,480	0	0,030	0,460	0,511	0,235	0,485
			D2	0,504	0	4E-06		0,504	0,232	0,481
			D3	0,015	0	0,968		0,983	0,453	0,673
			D4	0	0,519	0		0,519	0,239	0,489
			D5	0	0,480	0		0,480	0,221	0,470

Sehingga didapatkan hasil *similiarity* atau relevansi *query* dengan menggunakan perhitungan *cosine similiarity* dapat dilihat pada Tabel 4.20

Tabel 4.20 Hasil similiaritas

Hasil Sebelum diurutkan		Hasil setelah diurutkan	
E1	0,956	E2	0,998
E2	0,998	E1	0,956
E3	0,174	E3	0,174
E4	0,000	E4	0,000
E5	0,000	E5	0,000

Dari hasil pada Tabel 4.20 didapat hasil *similiarity query* yang tertinggi yaitu dengan dokumen E2 yang memiliki sentimen positif, maka *query* termasuk ke dalam katagori positif.

4.4 Pengujian manual

Proses Pengujian dilakukan menggunakan *confusion matrix*. Pertama mencari nilai *TP*, *FN*, *TN*, dan *FP*. Proses pengujian manual dilakukan menggunakan sampel data sebanyak 50 yang terdiri dari 25 data positif dan 25 data negatif. Sampel data dapat dilihat pada Tabel 4.21.

Tabel 4.21 Sampel data pengujian

n=50 Doc Uji	Doc Uji	Hasil	Check
+	**1**	-	FN
+	**2**	+	TP
+	**3**	+	TP
+	**4**	+	TP
+	**5**	+	TP
+	**6**	+	TP
+	**7**	+	TP
+	**8**	+	TP
+	**9**	+	TP
+	**10**	+	TP
+	**11**	+	TP
+	**12**	+	TP
+	**13**	+	TP
+	**14**	+	TP
+	**15**	+	TP
+	**16**	+	TP
+	**17**	+	TP
+	**18**	-	FN
+	**19**	+	TP

+	**20**	+	TP
+	**21**	-	FN
+	**22**	+	TP
+	**23**	+	TP
+	**24**	+	TP
+	**25**	+	TP
-	**26**	+	FP
-	**27**	-	TN
-	**28**	-	TN
-	**29**	-	TN
-	**30**	-	TN
-	**31**	-	TN
-	**32**	+	FP
-	**33**	-	TN
-	**34**	-	TN
-	**35**	-	TN
-	**36**	+	FP
-	**37**	+	FP
-	**38**	+	FP
-	**39**	+	FP
-	**40**	-	TN
-	**41**	-	TN
-	**42**	-	TN
-	**43**	-	TN
-	**44**	-	TN
-	**45**	-	TN
-	**46**	-	TN
-	**47**	-	TN
-	**48**	+	FP
-	**49**	+	FP
-	**50**	+	FP

Dari Table 4.21 didapatkan nilai *TP* sebanyak 22, nilai *TP* didapat jika dok uji positif dan hasil uji juga positif. Nilai *FN* didapat jika data uji positif namun menghasilkan hasil uji negatif, nilai *FN* didapat sebanyak 3. Nilai *TN* didapat jika data uji negatif dan hasil uji juga negatif, nilai *TN* didapat sebanyak 16. Nilai *FP* didapat jika data uji negatif dan hasil uji positif, nilai *FP* didapat sebanyak 9. Hasil dari nilai *TP, FN, TN,* dan *FP* dapat dilihat pada table 4.22

Tabel 4.22 Table nilai *TP*, *FN*, *TN* dan *FP*

TP	22
FN	3

TN	16
FP	9

Setelah mendapatkan nilai *TP*,*FN*,*TN* dan *FP* dilanjutkan untuk mencari nilai *true positif rate* dengan Persamaan 2.2 yang dijelaskan pada Bab 2, *true negative rate* dengan Persamaan 2.3 yang dijelaskan pada Bab 2 dan *akurasi* dengan persamaan 2.1 yang dijelaskan pada Bab 2. Proses perhitungan dapat di lihat pada Tabel 4.23.

Tabel 4.23 Tabel *Confusion Matrix*

n=50 k= 45	Predicted : No	Predicted : Yes	
Actual : No	TN = 16	FP = 9	25
Actual : Yes	FN = 3	TP = 22	25
	19	31	

Pengujian yang dilakukan disini adalah mencari nilai *true positive rate*, *true negative rate*, dan akurasi. Hasil pengujian dapat dilihat pada Tabel 4.24.

Tabel 4.24 Tabel Hasil Pengujian manual

True Positive Rate	True Negative Rate	Akurasi
0,88	0,64	78%

Hasil pengujian menunjukkan nilai *true positive rate* yang didapat adalah 0.88, nilai *true negative rate* yang didapat adalah 0.64 dan untuk hasil akurasi dari pengujian di atas adalah 78%.

4.5 Perancangan Pengujian

Pada sistem ini terdapat dilakukan beberapa scenario pengujian untuk melihat apakah kinerja sistem sesuai dengan tujuan awal penelitian. Pengujian yang akan digunakan yaitu pengujian *k-rank*, pengujian *true positive rate,* dan pengujian *true negative rate*.

4.6 Penarikan Simpulan

Penarikan simpulan dilakukan setelah semua tahapan mulai dari perancangan, implementasi dan pengujian sistem selesai di jalankan dan menghasilkan suatu hasil. Setelah penarikan simpulan selanjutnya akan diberikan saran. Saran ini ditulis sebagai pertimbangan untuk penelitian kedepannya yang dilakukan untuk pengembangan.

BAB 5 IMPLEMENTASI DAN PENGUJIAN

5.1 Batasan Implementasi

Batasan implementasi merupakan batasan yang dipenuhi oleh sistem berdasarkan bab-bab sebelumnya. Dengan tujuan agar sistem dapat fokus dalam proses menyelesaikan masalah yang terjadi dan tidak keluar dari tujuan utama penelitian. Beberapa batasan dalam proses implementasi pada penelitian ini adalah sebagai berikut:

1. Analisis sentiment ulasan video animasi pada penelitian ini diimplementasikan dengan bahasa pemrograman *Java*.
2. Metode yang dipergunakan untuk penyelesaian masalah pada penlitian ini yang digunakan adalah *Latent Semantic Indexing*.
3. Data yang digunakan berupa kalimat ulasan positif dan negatif yang diambil dari *website* myanimelist yang kemudian disimpan dalam fail txt.
4. Hasil keluaran adalah data sentiment data yang diuji yakni positif atau negatif
5. Pengujian akurasi, *true positive rate* dan *true negative rate* menggunakan metode *confusion matrix*.

5.2 implementasi Sistem

Pada Implementasi sistem, penulis menjelaskan tentang sistem yang dibuat melalui kode program. Tahapan implementasi sistem diuraikan berdasarkan tahap perancangan sistem. Pembahasan dilakukan per-kelas dari program yang di buat. Daftar seluruh proses dan fungsi dalam sistem di tunjukkan oleh Tabel 5.1.

Tabel 5.1 Daftar *Method*

No	Proses	Fungsi (Method)	Keterangan
1	*Pre-Processing*	`CaseFolding()`	Mengubah kata dari huruf besar menjadi huruf kecil dan menghapus tanda baca
		`Filtering()`	Menghapus kata yang yang tidak dibutuhkan dengan cara mengecek terhadap stopword yang di buat
		`PorterStemming()`	Menghapus awalan dan akhiran agar menjadi kata dasar

2	*TF-IDF*	`Tf()`	Menghitung nilai *term dari frequency* kata yang keluar
		`Df()`	Menghitung nilai frekuensi dokumen
		`Idf()`	Menghitung idf dengan cara mengkalikan Tf dan *Df*.
		`Wtd()`	Menghitung *TF-IDF* dokumen yang di *input.*
		`Normalisasi()`	Melakukan normalisasi terhadap hasil dari *Wtd*
3	*SVD*	`getMatrixS()`	Mendapatkan nilai matriks *S*
		`getMatriksU()`	Mendapatkan nilai matriks *U*
		`getMatriksV()`	Mendapatkan nilai matriks *V*
		`getMatrixSi()`	Mendapatkan nilai matriks *Si* dengan cara men inverse matrix *S*
		`getLSI()`	Mendapatkan nilai *q'* dengan menggunakan perhitungan LSI
		`getCosSin()`	Menghitung nilai cossin sampai pada mendapatkan nilai smiliaritas dokumen
		`getHasil()`	Mendapatkan hasil dari query yang telah di proses melalui getCosSin() untuk di cocokkan apakah query tersebut termasuk ke sentiment positif atau sentiment negatif
		`getPengujian()`	Menghitung hasil pengujian dari nilai yang didapat dari getHasil() dan di cocokkan pada index positif dan negatif data uji

5.2.1 Kelas SVD

Kelas *SVD* adalah kelas utama yang mengimplementasikan semua proses perhitungan sampai dengan proses pengujian. *Source code* dan pembahasan dari tiap *method* di kelas SVD dapat dilihat pada Kode Program 5.1.

```
1  public class SVD {
2
3      Matrix A;
4      double U[][];
5      double Vt[][];
6      double [][] normalisasi;
7      double [][] normalisasiLatih;
8      double [][] normalisasiUji;
9      int [] Y;
10     int [] YUji;
11     int [] maxIndex;
12     int [] hasilUji;
13     int nilaiK ;
14     TFIDFDoc Uji;
15     int dataNormalisasiLatih = 100;
16     int dataNormalisasiUji = 50;
17     double[][] normalisasiLatihT;
18     double[][] normalisasiUjiT;
19     //TFIDFDoc T;
20
21    public SVD(List<String> terms, List<List<String>> Documents,  int
nilaiKinput){
22         TFIDFDoc Latih = new TFIDFDoc(terms,Documents);
23         normalisasi = Latih.getNormalisasi();
24         nilaiK = nilaiKinput;
25         getNormalisasiLatih();
26         getNormalisasiTranspose();
27         setMatrixU();
28         setMatrixS();
29         setMatrixV();
30         setMatrixSi();
31
32          int dl = 50;
33          int Y1[] = new int[dl];
34          int Y2[] = new int[dl];
35
```

```
36          Y = new int[dataNormalisasiLatih];
37
38           for (int i = 0; i < dl; i++) {
39              Y1[i] = 1;
40              Y2[i] = 0;
41           }
42
43            for (int i = 0; i < dataNormalisasiLatih; i++) {
44              if (i < dl) {
45                  Y[i] = Y1[i];
46              } else  {
47                  Y[i] = Y2[i - dl];
48              }
49           }
50
51              int dlUji = 25;
52              int YUji1[] = new int [dlUji];
53              int YUji2[] = new int [dlUji];
54
55              YUji = new int[dataNormalisasiUji];
56
57              for (int i = 0; i < dlUji; i++) {
58                  YUji1[i] = 1;
59                  YUji2[i] = 0;
60          }
61
62              for (int i = 0; i < dataNormalisasiUji; i++) {
63                  if (i < dlUji){
64                      YUji[i] = YUji1[i];
65                  }else {
66                      YUji[i] = YUji2[i - dlUji];
67                  }
68              }
69
70              System.out.println("Y");
71              for (int i = 0; i < dataNormalisasiLatih; i++) {
72               System.out.println(Y[i]);
73              }
74      }
75
76      public void setUji(){
77
```

```
78          getNormalisasiUji();
79          setMatrixLSI();
80          getCosSin();
81          getHasil();
82          getPengujian();
83
84
85      }
86
87      public void getNormalisasiLatih(){
88          double [][] normalisasiTotal = normalisasi;
89          normalisasiLatih = new double [dataNormalisasiLatih][normalisasi[0].length];
90          System.out.println("normalisasi latih");
91          for (int i = 0; i < dataNormalisasiLatih; i++) {
92              for (int j = 0; j < normalisasi[i].length; j++) {
93                  normalisasiLatih[i][j] = normalisasiTotal[i][j];
94                  System.out.printf("%.2f",normalisasiLatih[i][j]);
95                  System.out.print("||");
96              }
97              System.out.println();
98          }
99          //return normalisasiLatih;
100     }
101
102     public void getNormalisasiUji(){
103         double [][] normalisasiTotal = normalisasi;
104         int l = 0;
105
106         normalisasiUji = new double[dataNormalisasiUji][normalisasi[0].length];
107                 System.out.println("normalisasi Uji");
108         for (int i = dataNormalisasiLatih ; i <  normalisasi.length; i++) {
109             for (int j = 0; j < normalisasi[i].length; j++) {
110                 normalisasiUji[l][j] = normalisasiTotal[i][j];
111
112             }
113             l++;
114         }
115        }
116
117     public void getNormalisasiTranspose(){
```

```
118         Matrix A = new Matrix(normalisasiLatih);
119         Matrix At = A.transpose();
120         normalisasiLatihT = At.getArray();
121         System.out.println("normalisasi Transpose");
122         for (int i = 0; i < normalisasiLatihT.length; i++) {
123             for (int j = 0; j < normalisasiLatihT[0].length; j++)  {
124
125             }
126
127         }
128       }
129
130
131     public double[][] getMatrixS(){
132       Matrix A = new Matrix(normalisasiLatihT);
133       Matrix AtA = A.transpose().times(A);
134
135       SingularValueDecomposition svd = A.svd();
136       Matrix U = svd.getU();
137       double [][] matrixU = U.getArray();
138
139       EigenvalueDecomposition e = AtA.eig();
140       Matrix D = e.getD();
141       Matrix S = e.getD();
142
143       int k = 0;
144       int l = 0;
145             for (int i = MatrixD.length -1; i >= 0; i--) {
146             l=0;
147             for (int j = MatrixD[i].length -1; j >= 0; j--) {
148                 MatrixS[k][l] = MatrixD[i][j];
149                 l++;
150             }
151             k++;
152
153       }
154
155        int a =0;
156        int b = MatrixS.length;
157        for (int i = 0; i < MatrixS.length; i++) {
158            for (int j = 0; j < MatrixS[i].length; j++) {
```

```
                if (MatrixS[i][i] == MatrixS[j][j] &&  MatrixS[i][j]
159 <= 0) {
160                     a++;
161                     break;
162                 }else{
163
164                 }
165             }
166         }
167
168
            double [][] hasil = new
169 double[matrixU[0].length][matrixU[0].length];
170         for (int i = 0; i < matrixU[0].length; i++) {
171             for (int j = 0; j < matrixU[0].length; j++) {
172                 hasil [i][j] = MatrixS[i][j];
173
174             }
175                     }
176
177         return hasil;
178
179     }
180
181
182     public double [][] getMatrixU(){
183         Matrix A = new Matrix(normalisasiLatihT);
184         SingularValueDecomposition svd = A.svd();
185
186         Matrix U = svd.getU();
187         double [][] MatrixU = U.getArray();
188         double [][] MatrixUk = new double  [MatrixU.length][nilaiK];
189         for (int i = 0; i < MatrixU.length; i++) {
190             for (int j = 0; j < nilaiK; j++) {
191                 MatrixUk[i][j] = MatrixU[i][j];
192             }
193
194         }
195         return MatrixUk;
196     }
197
198
199     public double[][] getMatrixV(){
```

```
200         Matrix A = new Matrix(normalisasiLatihT);
201         SingularValueDecomposition svd = A.svd();
202
203         Matrix V = svd.getV();
204         double [][] matrixV = V.getArray();
205         double [][] matrixVk = new  double[matrixV.length][nilaiK];
206         for (int i = 0; i < matrixV.length; i++) {
207             for (int j = 0; j < nilaiK; j++) {
208                 matrixVk[i][j] = matrixV[i][j];
209
210             }
211             //System.out.println();
212         }
213         return matrixVk;
214     }
215
216     public double[][] getMatrixSi(){
217
218          Matrix B = new Matrix (getMatrixS());
219          Matrix Si = B.inverse();
220          double matrixSi [][] = Si.getArray();
221          double matrixSik [][] = new double [nilaiK][nilaiK];
222          for (int i = 0; i < nilaiK; i++) {
223              for (int j = 0; j < nilaiK; j++) {
224                  matrixSik[i][j] = matrixSi[i][j];
225              }
226
227         }
228
229
230          return matrixSik;
231     }
232
233     public void getNormalisasiUjiTranspose(){
234         Matrix A = new Matrix(normalisasiUji);
235         Matrix At = A.transpose();
236         normalisasiUjiT = At.getArray();
237         System.out.println("normalisasi Uji Transpose");
238         for (int i = 0; i < normalisasiUjiT.length; i++) {
239             for (int j = 0; j < normalisasiUjiT[0].length; j++) {
240                 System.out.printf("%.2f",normalisasiUjiT[i][j]);
241                 System.out.print("||");
```

```
242             }
243             System.out.println();
244         }
245     }
246
247
248     public double[][] getLSI(){
249
250
251         Matrix uji = new Matrix(normalisasiUji);
252         Matrix U = new Matrix(getMatrixU());
253         Matrix Si = new Matrix(getMatrixSi());
254         Matrix LSI = U.times(Si);
255         LSI = uji.times(LSI);
256
257         double [][] MatrixLSI = LSI.getArray();
258
259         return MatrixLSI;
260     }
261
262
263
264     public void getCosSin(){
265         double [][] q = getLSI();
266         double [][] d = getMatrixV();
267
268         int count = d.length*q.length;
269         double [][] qd  = new double [count][q[0].length];
270         int b = 0;
271         int c = 0;
272         for (int i = 0; i < q.length; i++) {
273             for (int j = 0; j <d.length; j++){
274                 for (int k = 0; k< d[0].length; k++){
275                     qd[b][k] = q[i][k]*d[j][k];
276                 }
277                  b++;
278             }
279
280         }
281
282         //double [][] Pembilang = Pem.getArray();
283         System.out.println("Q * D");
```

```
284         for (int i = 0; i < count; i++) {
285
286             for (int j = 0; j < q[0].length; j++) {
287
288                 System.out.printf("%.2f",qd[i][j]);
289                 System.out.print("||");
290             }
291             System.out.println();
292
293
294         double [] totalqd = new double[count];
295         System.out.println("Matrix Pembilang");
296         for (int i = 0; i < count; i++) {
297             for (int j=0; j<q[0].length;j++){
298                 totalqd[i] += qd[i][j];
299             }
300             System.out.printf("%.4f",totalqd[i]);
301             System.out.println();
302         }
303
304
305         //PENYEBUT
306         double [][] q2 = new double[q.length][q[0].length];
307         System.out.println("q2");
308         for (int i=0;i<q2.length;i++){
309              for(int j=0;j<q2[i].length;j++){
310                  q2[i][j] = Math.pow(q[i][j], 2);
311                  System.out.printf("%.2f",q2[i][j]);
312                  System.out.print("||");
313              }
314              System.out.println("");
315         }
316
317         double [][] d2 = new double [d.length][d[0].length];
318         System.out.println("d2");
319         for (int i=0;i<d2.length;i++){
320              for(int j=0;j<d2[i].length;j++){
321                  d2[i][j] = Math.pow(d[i][j], 2);
322                  System.out.printf("%.2f",d2[i][j]);
323                  System.out.print("||");
324              }
```

```
325             System.out.println("");
326         }
327
328         double [] totalq2 = new double[q2.length];
329         System.out.println("totalq2");
330         for (int i = 0; i < q2.length; i++) {
331             for (int j=0; j<q2[i].length;j++){
332                 totalq2[i] += q2[i][j];
333             }
334             System.out.printf("%.2f",totalq2[i]);
335             System.out.println();
336         }
337
338         double [] totald2 = new double[d2.length];
339         System.out.println("totald2");
340         for (int i = 0; i < d2.length; i++) {
341             for (int j=0; j<d2[i].length;j++){
342                 totald2[i] += d2[i][j];
343             }
344             System.out.printf("%.2f",totald2[i]);
345             System.out.println();
346         }
347
348         double [] totalq2d2  = new double [count];
349         int e = totald2.length;
350         for (int i = 0; i < totalq2.length; i++) {
351             for (int j = 0; j <totald2.length; j++){
352                     c++;
353                 }
354             }
355
356         for (int i = 0; i < count; i++) {
357             if (totalq2d2[i]==0){
358                 totalq2d2[i]=1;
359             }
360         }
361
```

```
362
363             //Pembilang / Penyebut
364             double[] PemPen = new double[count];
365
366             for (int i = 0; i < count; i++) {
367                 PemPen[i] = totalqd[i]/totalq2d2[i];
368         }
369
370
371         List<double[]> splitted = new ArrayList<double[]>();  //This
    list will contain all the splitted arrays.
372         int lengthToSplit = d.length;
373
374         int arrayLength = PemPen.length;
375
376         for (int i = 0; i < arrayLength; i = i + lengthToSplit) {
377             double[] val = new double[lengthToSplit];
378
379             if (arrayLength < i + lengthToSplit) {
380                 lengthToSplit = arrayLength - i;
381             }
382             System.arraycopy(PemPen, i, val, 0, lengthToSplit);
383             splitted.add(val);
384             }
385
386
387             System.out.println("splitted");
388             for (int i = 0; i < q.length; i++) {
389                 for (int j = 0; j<d.length; j++){
390                     System.out.println(splitted.get(i)[j]);
391                                     }
392                 System.out.println();
393         }
394
395
396             //Sorting
397
398              System.out.println("splitted");
399             for (int i = 0; i < q.length; i++) {
400                 for (int j = 0; j<d.length; j++){
401                     System.out.println(splitted.get(i)[j]);
402                 }
```

```
403                 System.out.println();
404             }
405
406              //CheckMAX
407              maxIndex = new int[splitted.size()];
408              double [] maxValue = new double [splitted.size()];
409              for (int i = 0; i < splitted.size(); i++) {
410                  maxValue[i] =0;
411                  maxIndex[i] =0;
412                 for (int j = 0; j<splitted.get(i).length; j++){
413                  if (splitted.get(i)[j] > maxValue[i]) {
414                      maxValue[i] = splitted.get(i)[j];
415                      maxIndex[i] = j;
416                  }
417             }
418         }
419          }
420
421 }
422
423
424 public void getHasil(){
425     hasilUji  = new int [maxIndex.length];
426     System.out.println("Hasil Uji");
427     for (int i = 0; i < maxIndex.length ; i++) {
428        hasilUji[i] = Y[maxIndex[i]];
429        }
430              }
431 }
432
433 public void getPengujian(){
434     int tp =0;
435     int fn =0;
436     int tn =0;
437     int fp =0;
438     for (int i = 0; i < hasilUji.length; i++) {
439         if(YUji[i]==1 && hasilUji[i]==1){
440             tp +=1;
441         }else if(YUji[i]==1 && hasilUji[i]==0){
442             fn +=1;
443         }else if(YUji[i]==0 && hasilUji[i]==0){
444             tn +=1;
```

```
445             }else if(YUji[i]==0 && hasilUji[i]==1){
446                 fp +=1;
447             }
448         }
449
450
451         //Perhitungan accuracy
452         double accuracy = (0.1 * (tp+tn)) / (0.1 *  (tp+fn+tn+fp));
453         double TPRate = (0.1 * (tp)) / (0.1 *(fn+tp));
454         double specificity = (0.1 * (tn)) / (0.1* (tn+fp));
455     }
456
```

Kode Program 5.1 Impelementasi kode program utama perhitungan *SVD* dan *LSI*

Keterangan Kode Program 5.1.

- Baris 3-18 : melakukan inisialisasi atribut
- Baris 21-30 : konstruktor untuk menjalankan *method* data latih
- Baris 32-49 : membuat indeks data latih
- Baris 51-68 : membuat indeks data uji
- Baris 76 -85 : *method* untuk menjalankan *method* data uji
- Baris 87-100 : mendapatkan nilai tf-idf ternormalisasi untuk data latih
- Baris 102-115 : mendapatkan nilai tf-idf ternormalisasi untuk data uji
- Baris 117-128 : mendapatkan nilai *transpose* dari tf-idf ternormalisasi untuk data latih
- Baris 131-179 : menghitung matriks *S*
- Baris 182-196 : menghitung matriks *U*
- Baris 199-213 : menghitung matriks *V*
- Baris 216-230 : menghitung matriks S yang akan di *Inverse*
- Baris 233-245 : mendapatkan nilai transpose dari *tf-idf* ternormalisasi untuk data uji
- Baris 248-259 : menghitung Matriks LSI yang telah didapatkan nilai nya dari matriks *S*, matriks *U* dan matriks *S inverse*
- Baris 264-368 : menghitung nilai *cosine similiarity* untuk mencari kedekatan dokumen latih dengan dokumen uji
- Baris 398-403 : mengurutkan hasil kedekatan dokumen uji di tiap dokumen latih dari yang terbesar ke terkecil

- Baris 407-421 : mendapatkan dokumen dengan kedekatan terbesar dari data uji terhadap data latih
- Baris 424-428 : menunjukan hasil pengujian kedekatan dokumen tiap data uji terhadap data latih apakah dokumen data uji tersebut masuk ke dalam kelas positif atau kelas negatif
- Baris 433-455 : menghitung pengujian akurasi program serta, *true positif rate* dan *true negative rate*

5.3 Pengujian Nilai *k-rank*

Pengujian dilakukan untuk mengetahui hasil dari program yang dibuat dengan perbedaan dari nilai *k-rank* yang di *inputkan* dan perbedaan jumlah data latih positif dan negatif. Dari banyak perbedaan nilai *k-rank* dan perbedaan jumlah data latih positif dan negatif yang di uji, dilakukan analisis terhadap pengujian yang telah dilakukan.

Pengujian ini dilakukan dengan menguji 50 data uji yang terdiri dari 25 data uji positif dan 25 data uji negatif data ke 3 varian data latih. Pengujian pertama menggunakan data latih sebanyak 100 dengan rasio 50 data positif dan 50 data negatif. pengujian kedua menggunakan data latih sebanyak 150 dengan rasio 100 data positif dan 50 data negatif. Pengujian ketiga menggunakan data latih sebanyak 150 dengan rasio 50 data positif dan 100 data negatif. untuk mengentahu nilai *k-rank* terbaik dari variasi nilai *k –rank* yang diuji, pengujian dilakukan sebanyak 19 kali dengan 19 nilai *k-rank* yang berbeda. Hasil pengujian data dilihat apda Tabel 5.2.

Tabel 5.2 Hasil Pengujian akurasi pada beberapa nilai *k-rank* dari 3 data latih yang berbeda

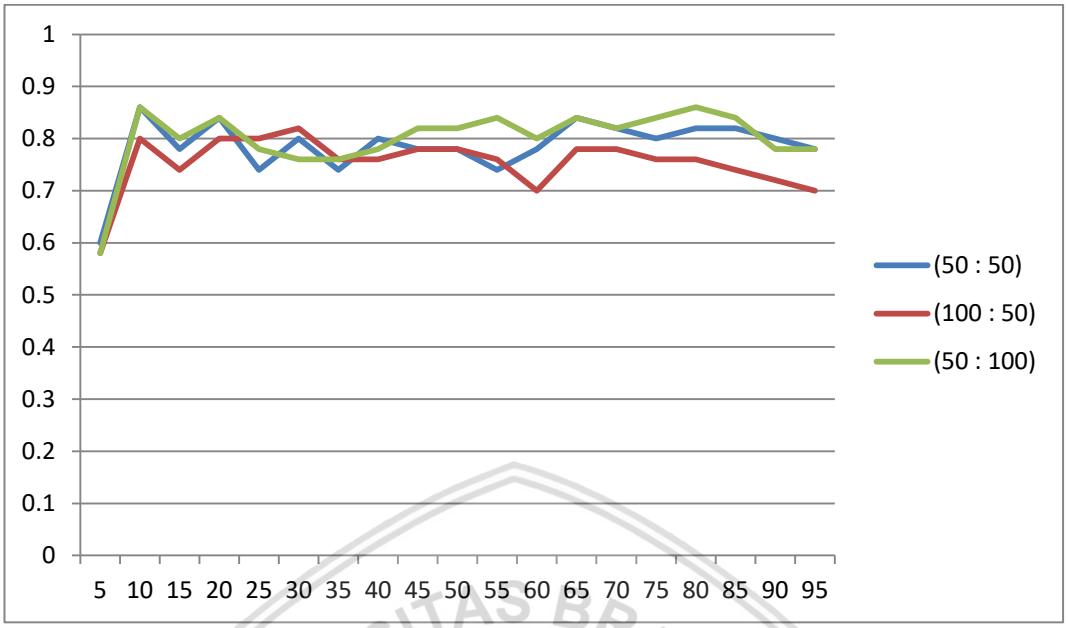
Pengujian Ke -	Nilai *k-rank*	Akurasi		
		Data latih (50 positif : 50 negatif)	Data latih (100 positif : 50 negatif)	Data latih (50 positif : 100 negatif)
1	5	0,60	0,58	0,58
2	10	**0,86**	0,80	**0,86**
3	15	0,78	0,74	0,80
4	20	0,84	0,80	0,84
5	25	0,74	0,80	0,78
6	30	0,80	**0,82**	0,76
7	35	0,74	0,76	0,76

8	40	0,80	0,76	0,78
9	45	0,78	0,78	0,82
10	50	0,78	0,78	0,82
11	55	0,74	0,76	0,84
12	60	0,78	0,70	0,80
13	65	0,84	0,78	0,84
14	70	0,82	0,78	0,82
15	75	0,80	0,76	0,84
16	80	0,82	0,76	**0,86**
17	85	0,82	0,74	0,84
18	90	0,80	0,72	0,78
19	95	0,78	0,70	0,78

5.3.1 Hasil dan analisis *k-rank*

Pada tiga kali pengujian nilai *k-rank* terhadap akurasi yang didapatkan program dengan jumlah data uji sebanyak 50. Terlihat pada pengujian pertama dengan jumlah data latih sebanyak 50 data positif dan 50 data negatif menghasilkan akurasi optimal pada nilai *k-rank* = 10 sebesar 86%. Pada pengujian kedua dengan jumlah data latih sebanyak 100 data positif dan 50 data negatif menghasilkan akurasi optimal pada nilai *k-rank* = 30 sebesar 82% dan pada pengujian ketiga dengan jumlah data latih sebanyak 50 data positif dan 100 data negatif menghasilkan akurasi optimal pada nilai *k-rank* = 10 dan *k-rank* = 80 sebesar 86%.

Pada Tabel 5.7 dapat dilihat bahwa nilai *k-rank* yang dimasukkan sangat berpengaruh terhadap hasil akurasi yang didapatkan. Hal ini terlihat ketika nilai *k-rank* yang dimasukkan sangat kecil, maka hasil akurasi kurang optimal dikarenakan fitur yang terlalu kecil akan membuat data menjadi bias. Hasil kurang optimal juga di dapatkan saat nilai *k-rank* yang dimasukkan terlalu besar, dikarenakan terlalu banyak fitur sehingga *range* datanya terlalu luas. Dari tiga kali pengujian yang dilakukan dengan jumlah data latih yang berbeda-beda dapat disimpulkan bahwa besarnya nilai *k-rank* yang dimasukkan sangat berpengaruh terhadap akurasi yang dihasilkan program karena *k-rank* merupakan banyak fitur pada data latih. Hasil akurasi dari pengujian nilai *k-rank* dapat dilihat pada Gambar 5.1.



Gambar 5.1 Grafik Akurasi pada beberapa nilai *k-rank* dari 3 data latih yang berbeda

5.4 Pengujian *True Positive Rate*

Pengujian *True Positive Rate* dilakukan untuk melihat berapa persentase data uji positif yang menghasilkan hasil positif. Pengujian dilakukan dengan jumlah data uji positif sebanyak 25. Untuk melihat persentase *true positive rate* dilkaukan pengujian dengan menggunakan Persamaan 2.2. Hasil dari pengujian *true positive rate* dapat dilihat pada Tabel 5.3.

Tabel 5.3 Hasil Pengujian *True Positive Rate* pada beberapa nilai *k-rank* dari 3 data latih yang berbeda

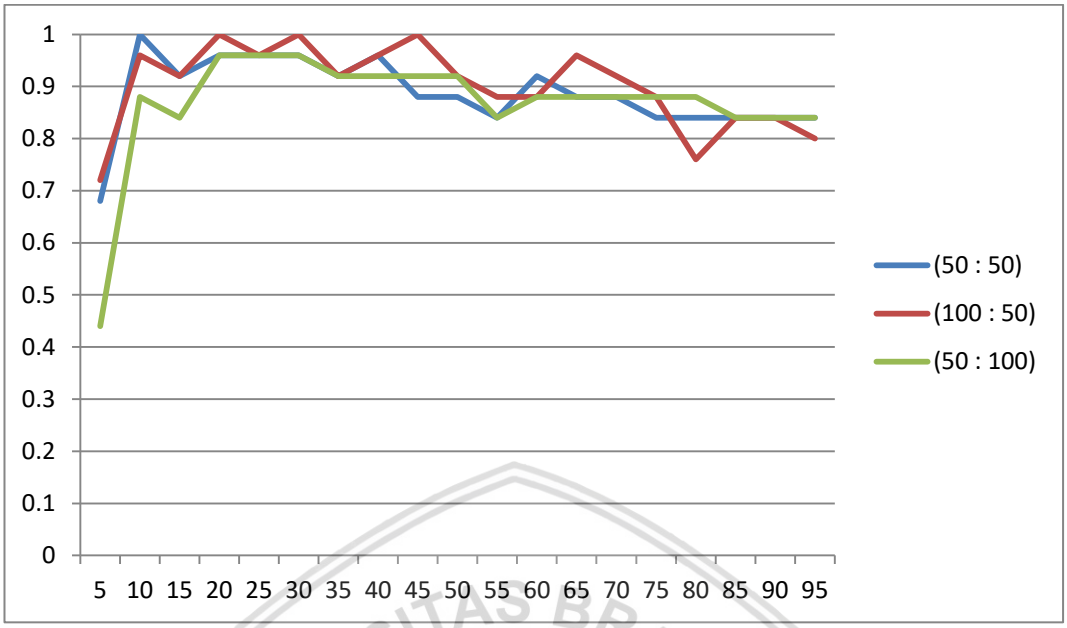
Pengujian Ke -	Nilai *k-rank*	True Positive Rate		
		Data latih (50 positif : 50 negatif)	Data latih (100 positif : 50 negatif)	Data latih (50 positif : 100 negatif)
1	5	0,68	0,72	0,44
2	10	**1,00**	0.96	0.88
3	15	0,92	0,92	0,84
4	20	0,96	**1,00**	**0,96**
5	25	0,96	0,96	**0,96**

6	30	0,96	**1,00**	**0,96**
7	35	0,92	0,92	0,92
8	40	0,96	0,96	0,92
9	45	0,88	**1,00**	0,92
10	50	0,88	0,92	0,92
11	55	0,84	0,88	0,84
12	60	0,92	0,88	0,88
13	65	0,88	0,96	0,88
14	70	0,88	0,92	0,88
15	75	0,84	0,88	0,88
16	80	0,84	0,76	0,88
17	85	0,84	0,84	0,84
18	90	0,84	0,84	0,84
19	95	0,84	0,80	0,84

5.4.1 Hasil dan Analisis *True Positive Rate*

Pada tiga kali pengujian *true positive rate* dengan jumlah data uji sebanyak 25. Terlihat pada pengujian pertama dengan jumlah data latih sebanyak 50 data positif dan 50 data negatif menghasilkan persentase optimal pada nilai *k-rank* = 10 sebesar 100%. Pada pengujian kedua dengan jumlah data latih sebanyak 100 data positif dan 50 data negatif menghasilkan persentase optimal pada nilai *k-rank* = 20, 30 dan 45 sebesar 100% dan pada pengujian ketiga dengan jumlah data latih sebanyak 50 data positif dan 100 data negatif menghasilkan persentase optimal pada nilai *k-rank* = 20, 25 dan 30 sebesar 96%.

Pada Tabel 5.2 dapat dilihat besar persetanse pengujian *true positive rate* dengan nilai *k-rank* dan jumlah data latih yang berbeda, terlihat bahwa hasil persentase *true positive rate* dapat dikatakan baik dikarenakan data uji positif aktual dengan hasil dari pengujian *true positive rate* memiliki nilai yang hampir sama bahkan sampai dengan 100% sama. Pada pengujian pertama nilai *true positive rate* pada *k-rank* = 10 menghasilkan persentase 100% yang berarti jumlah data uji positif yang menghasilkan hasil positif sebanyak 25 dan hasil tersebut sama dengan jumlah data uji positif aktual. Hasil Pengujian *true positive rate* dapat dilihat pada Gambar 5.2.



Gambar 5.2 Grafik *true positive rate* pada beberapa nilai *k-rank* dari 3 data latih yang berbeda

5.5 Pengujian *True Negative Rate*

Pengujian *True Negative Rate* dilakukan untuk melihat berapa persentase data uji negatif yang menghasilkan hasil negatif. Pengujian dilakukan dengan jumlah data uji negatif sebanyak 25. Untuk melihat persentase *true negative rate* dilakukan pengujian dengan menggunakan Persamaan 2.3. Hasil dari pengujian *true positive rate* dapat dilihat pada Tabel 5.4.

Tabel 5.4 Hasil Pengujian *True Negative Rate* pada beberapa nilai *k-rank* dari 3 data latih yang berbeda

Pengujian Ke -	Nilai *k-rank*	True Negative Rate		
		Data latih (50 positif : 50 negatif)	Data latih (100 positif : 50 negatif)	Data latih (50 positif : 100 negatif)
1	5	0,52	0,44	0,72
2	10	0,72	0,64	**0,84**
3	15	0,64	0,56	0,76
4	20	0,72	0,60	0,72
5	25	0,52	**0,68**	0,60
6	30	0,64	0,64	0,56

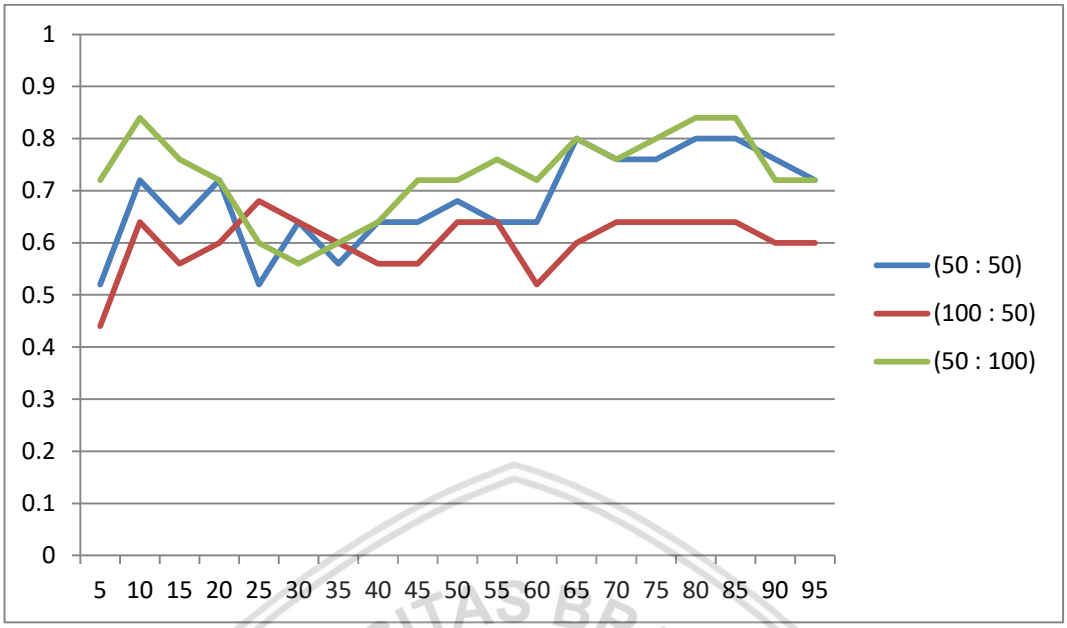
7	35	0,56	0,60	0,60
8	40	0,64	0,56	0,64
9	45	0,64	0,56	0,72
10	50	0,68	0,64	0,72
11	55	0,64	0,64	0,76
12	60	0,64	0,52	0,72
13	65	**0,80**	0,60	0,80
14	70	0,76	0,64	0,76
15	75	0,76	0,64	0,80
16	80	**0,80**	0,64	**0,84**
17	85	**0,80**	0,64	**0,84**
18	90	0,76	0,60	0,72
19	95	0,72	0,60	0,72

5.5.1 Hasil dan Analisis Pengujian *True Negative Rate*

Pada tiga kali pengujian *true negative rate* dengan jumlah data uji sebanyak 25. Terlihat pada pengujian pertama dengan jumlah data latih sebanyak 50 data positif dan 50 data negatif menghasilkan persentase optimal pada nilai *k-rank* = 65, 80 85 sebesar 80%. Pada pengujian kedua dengan jumlah data latih sebanyak 100 data positif dan 50 data negatif menghasilkan persentase optimal pada nilai *k-rank* = 25, sebesar 68% dan pada pengujian ketiga dengan jumlah data latih sebanyak 50 data positif dan 100 data negatif menghasilkan persentase optimal pada nilai *k-rank* = 10, 80 dan 85 sebesar 84%.

Pada Tabel 5.2 dapat dilihat besar persetanse pengujian *true negative rate* dengan nilai *k-rank* dan jumlah data latih yang berbeda, terlihat bahwa hasil persentase *true negative rate* dapat dikatakan cukup baik dikarenakan hasil persentase masih termasuk tinggi. Namun, hasil pengujian *true negative rate* menghasilkan rata – rata di bawah 65%. Hal ini dikarenakan, ketika suatu data memiliki kata–kata positif namun mengadung makna spoiler/membeberkan jalan cerita dari sebuah cerita yang menyebabkan data tersebut mendapat penilaian negatif meskipun berisi kata kata positif. Hasil Pengujian *true negative rate* dapat dilihat pada Gambar 5.3.

.



Gambar 5.3 Grafik *true negative rate* pada beberapa nilai *k-rank* dari 3 data latih yang berbeda

BAB 6
PENUTUP

Dalam bab ini akan dibahas simpulan dari perancangan, implementasi, serta pengujian sistem dari analisis sentimen ulasan video animasi menggunakan metode *Latent Semantic Indexing,* serta saran untuk digunakan dalam pengembangan yang lebih baik lagi ke depannya.

6.1 Kesimpulan

Dari hasil Perancangan, implementasi, pengujian dan analisis pada penilitian analisis sentimen ulasa video animasi menggunakan metode *Latent Semantic Indexing* maka dapat diambil kesimpulan sebagai berikut:

1. Metode *Latent Semantic Indexing* (LSI) dapat diimplementasikan pada analisis sentimen ulasan video animasi. Data ulasan diambil dari website myanimelist.net. *Pre-Processing* yang digunakan yaitu *Case Folding,* Tokenisasi, *Filtering* dan *Stemming*. Lalu di proses dengan metode TF IDF untuk mengubah kata menjadi sebuah nilai.
2. Metode LSI untuk analisis sintimen pada ulasa video animasi menghasilkan akurasi optimal sebesar 86% dari percobaan dengan 19 nilai *k* yang berbeda. Dari hasil pengujian dianalisis bahwa semakin besar nilai *k-rank* yang di uji maka data yang dihasilkan akan stabil namun semakin besar nilai *k-rank* yang dimasukkan hasil akurasi yang diapat akan semakin menurun dikarenakan terlalu banyak fitur sehingga range datanya terlalu luas. Sedangkan untuk nilai *k*-rank yang terlalu kecil memberikan hasil yang kurang bagus dikarenakan fitur yang terlalu kecil akan membuat data menjadi bias.

6.2 Saran

Untuk dapat meningkatkan hasil dari yang didapat pada penelitian ini, diharapkan untuk penelitian selanjutnya bisa melakukan beberapa perbaikan pada hal-hal sebagai berikut:

1. Diharapkan pada peneliti selanjutnya untuk melakukan optimasi fitur atau agar hasil akurasi yang didapat lebih maksimal.
2. Diharapkan pada peneliti selanjutnya untuk menggunakan metode yang dapat mereduksi fitur agar hasil yang didapatkan lebih baik.

DAFTAR PUSTAKA

Budi, S. 2017 Text Mining Untuk Analisis Sentimen Review Film Menggunakan Algoritma K-Means. Techno.COM, Vol. 16, No. 1.

Ehkharghani, R., Mercan, H., Javeed, A., & Saygin, Y., 2014. Dehkharghani, R., Mercan, H., Javeed, A., & Saygin, Y. Sentimental causal rule discovery from Twitter. Expert Systems with Applications, 41(10), 4950–4958. doi:10.1016/j.eswa.2014.02.024.

Ipmawati, J., Kusrini, & Lithfi. E.T., 2017. Komparasi Teknik Klasifikasi Teks Mining Pada Analisis Sentimen, Indonesian Journal on Networking and Security - Volume 6 No 1.

Kristiyanti, D. A., 2015. Analisis Sentimen Review Produk Kosmetik Menggunakan Algoritma Support Vector Machine Dan Particle Swarm Optimization Sebagai Metode Seleksi Fitur, Seminar Nasional Inovasi dan Tren (SNIT).

Liu, B., & Zhang, Z., 2012. A Survey of Opinion Mining and Sentiment Analysis. Springer, US.

Medhat, W., Hassan, A., & Korashy, H., 2014. Sentiment analysis algorithms and applications : Asurvey. Ain Shams EngineeringJournal, 5(4), 1093–1113.

Santra, B. K., & Christy, C. J., 2012. Genetic Algorithm and Confusion Matrix for Document Clustering, International Journal of Computer Science Issues, Vol. 9, Issue 1, No 2.

Sari, Y., Ridok, A., 2012, Penentuan lirik lagu berdasarkan emosi menggunakan sistem temu kembali informasi dengan metode latent semantic indexing. Universitas Brawijaya, Malang.

Sari, Y., & Puspaningrum, E.Y., 2013, Pencarian Semantik Dokumen Berita Menggunakan Essential Dimension of Latent Semantic Indexing dengan Memakai Reduksi Fitur Document Frequency dan Information Gain Thresholding. Universitas Brawijaya, Malang.

Vijayarani, S., Ilamathi, J., & Nithya., 2015. Preprocessing Techniques for Text Mining - An Overview. International Journal of Computer Science & Communication Networks,Vol 5(1),7-16.

Yamout, F., Demachkieh, R., Hamdan, G., Sabra, R., 2004. Further Enhancement to the Porter's Stemming Algorithm. In: Machine Learning and Interaction for Text based Information Retrieval, Germany, pp. 7– 23.

Zafikri, Atika., 2008, Implementasi Metode Term Frequency Inverse Document Frequency (TF-IDF) Pada Sistem Temu Kembali Informasi. Universitas Sumatera Utara, Medan.

Zhang, Z., Ye, Q., Zhang, Z., & Li, Y. 2011. Sentiment classification of Internet restaurant reviews written in Cantonese. Expert Systems with Applications, 38(6), 7674– 7682. doi:10.1016/j.eswa.2010.12.147.