PEMBENTUKAN TOPIK PADA ARTIKEL JURNAL ILMU KOMPUTER MENGGUNAKAN ALGORITMA *E-ROCK*

(Enhanched-RObust Clustering using linKs)

SKRIPSI

Diajukan untuk memenuhi persyaratan

memperoleh gelar Sarjana Komputer

Disusun oleh:

Novieta Putri Rachmawati

NIM.0710963042

PROGRAM STUDI TEKNIK INFORMATIKA

PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER

UNIVERSITAS BRAWIJAYA

MALANG

2012

LEMBAR PERSETUJUAN

PEMBENTUKAN TOPIK PADA ARTIKEL JURNAL ILMU KOMPUTER MENGGUNAKAN ALGORITMA *E-ROCK*

(Enhanched-RObust Clustering using linKs)

SKRIPSI

Disusun oleh:

Novieta Putri Rachmawati

NIM.0710963042

Telah diperiksa dan disetujui oleh :

Pembimbing I,	Pembimbing II,
Lailil Muflikhah S.Kom, M.Sc	**Drs.Achmad Ridok,M.Kom**
NIP. 197411132005012001	**NIP. 196808251994031002**

LEMBAR PENGESAHAN

PEMBENTUKAN TOPIK PADA ARTIKEL JURNAL ILMU KOMPUTER MENGGUNAKAN ALGORITMA *E-ROCK*

(Enhanched-RObust Clustering using linKs)

SKRIPSI

Diajukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh:

Novieta Putri Rachmawati
NIM.0710963042

Skripsi ini telah diuji dan dinyatakan lulus pada
tanggal 30 November 2012

Penguji I	Penguji II
Drs. Marji, M.T.	**Candra Dewi, S.Kom., MSc**
NIP. 19750819 199903 1 001	**NIP. 19771114 200312 2 001**

Penguji III

Dian Eka Ratnawati, S.Kom., M.Kom
NIP. 19730619 200212 2 001

Mengetahui
Ketua Program Studi Teknik Informatika

Drs. Marji, M.T.
NIP. 19670801 199203 1 001

PERNYATAAN
ORISINALITAS SKRIPSI

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 dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan 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, 30 November 2012
Mahasiswa,

Novieta Putri Rachmawati
NIM 0710963042

KATA PENGANTAR

Alhamdulillah, dengan mengucapkan puji syukur kehadirat Allah SWT yang telah memberikan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan skripsi ini. Skripsi yang berjudul **"Pembentukan Topik Pada Artikel Jurnal Ilmu Komputer Menggunakan Algoritma *E-ROCK (Enhanched -RObust Clustering using linKs)* "**merupakan salah satu syarat memperoleh gelar Sarjana Komputer Program Studi Ilmu Komputer Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya Malang. Tidak dapat dipungkiri bahwa tidak mungkin penulis dapat menyelesaikan skripsi ini tanpa bantuan dan dukungan dari banyak pihak. Untuk itu, dengan ketulusan dan kerendahan hati penulis menyampaikan ucapan terima kasih yang sebesar-besarnya kepada:

1. Lailil Muflikhah S.Kom, M.Sc, selaku dosen pembimbing utama yang telah meluangkan waktu untuk memberikan pengarahan dan masukan bagi penulis.
2. Drs. Achmad Ridok, M.Kom selaku dosen pembimbing kedua yang telah meluangkan waktu untuk memberikan pengarahan dan masukan bagi penulis.
3. Bapak Ir. Sutrisno, M.T, Bapak Ir. Heru Nurwasito, M.Kom, Bapak Himawat Aryadita, S.T, M.Sc, dan Bapak Eddy Santoso, S.Kom selaku Ketua, Wakil Ketua 1, Wakil Ketua 2 dan Wakil Ketua 3 Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya.
4. Bapak Drs. Marji, M.T dan Bapak Issa Arwani, S.Kom, M.Sc selaku Ketua dan Sekretaris Program Studi Teknik Informatika Universitas Brawijaya.
5. Reza Andria Siregar, ST selaku dosen penasehat akademik yang selalu memberikan nasehat kepada penulis selama menempuh masa studi.
6. Maulfi,S.Pd, M.Pd, selaku ahli bahasa Indonesia Universitas Brawijaya yang telah membantu penulis dalam melengkapi data yang digunakan sebagai bahan skripsi .
7. Seluruh Dosen Teknik Informatika & FMIPA Universitas Brawijaya atas kesediaan membagi ilmunya kepada penulis.

8. Seluruh Civitas Akademika Teknik Informatika Universitas Brawijaya yang telah banyak memberi bantuan dan dukungan selama penulis menempuh studi di Teknik Informatika Universitas Brawijaya dan selama penyelesaian skripsi ini.
9. Kedua orang tua, bapak ibu, kakak tercinta dan adik terima kasih atas semua doa, kasih sayang dan perhatian yang tulus serta dukungan yang telah diberikan.
10. R Yan Vendy Dwi Cahyo, Yuita Arum S, Sigit Adinugroho, Kadek Andriana Putra, Sony Novianto, Aprilliana Dharma Susanti, Maria Ulfah, Noralia Ayu Shinda, Marissa Hamnon dan sahabat-sahabat di Program Studi Ilmu Komputer Fakultas MIPA xii Universitas Brawijaya yang telah banyak memberikan dukungannya demi kelancaran pelaksanaan penyusunan skripsi ini.
11. Dan semua pihak yang telah terlibat baik secara langsung maupun tidak langsung yang tidak dapat penulis sebutkan satu per satu terima kasih atas semua bantuan yang telah diberikan.

Semoga skripsi ini bermanfaat bagi pembaca sekalian. Akhirnya, penulis menyadari bahwa skripsi ini masih jauh dari kesempurnaan dan mengandung banyak kekurangan, sehingga dengan segala kerendahan hati penulis mengharapkan kritik dan saran yang membangun dari pembaca.

Malang, 30 November 2012

Penulis

ABSTRAK

Novieta Putri Rachmawati. 2012. : Pembentukan Topik Pada Artikel Jurnal Ilmu Komputer Menggunakan Algoritma *E-ROCK (Enhanched -RObust Clustering using linKs)*.

Skripsi Program Ilmu Komputer, Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya.

Dosen Pembimbing : Lailil Muflikhah S.Kom, M.Sc dan Drs. Achmad Ridok, M.Kom

Teknologi yang semakin berkembang memudahkan akses apapun, dimanapun, dan kapanpun, salah satunya adalah memberikan kemudahan untuk menemukan materi pendidikan yang diinginkan. Akan tetapi dibalik kemudahan tersebut, terdapat kebingungan dalam pemilihan topik yang diinginkan dikarenakan banyaknya materi yang tersedia. Cara alternatif untuk mempermudah pencarian topik adalah dengan mengelompokkan artikel jurnal berdasarkan kemiripan isinya. Salah satu metode diantara banyak metode yang digunakan untuk mengelompokkan data adalah algoritma *E-ROCK (Enhanched - RObust Clustering using linKs)*. Dalam penelitian ini algoritma *E-ROCK (Enhanched -RObust Clustering using linKs)* merupakan perbaikan dari algoritma *ROCK (RObust Clustering using linKs)* dimana digunakan rumus *cosine similarity distance* yang diyakini menghasilkan performa yang lebih baik dibandingkan rumus *similarity distance* yang terdapat dalam algoritma *ROCK (RObust Clustering using linKs)* . Penelitian ini menggunakan 100 dokumen artikel jurnal ilmiah Ilmu Komputer yang terbagi dalam 10 kategori. Hasil penelitian menunjukkan Didapatkan hasil *error ratio* minimum yang sebesar 16% dengan tingkat kesalahan pembentukan topik minimum sebesar 0.1 atau 10% pada saat *threshold* sebesar 0.65 . Sedangkan *error ratio* maksimum yang dihasilkan sebesar 61% dengan tingkat kesalahan pembentukan topic maksimal sebesar 0.5 atau 50% pada saat *threshold* sebesar 0.15 . Nilai *average error ratio* yang dihasilkan oleh agoritma *E-ROCK (Enhanched - RObust Clustering using linKs)* adalah sebesar 32.778%.

Kata Kunci : artikel, ilmu komputer, data, pengelompokan, *E-ROCK*.

ABSTRACT

Novieta Putri Rachmawati. 2012. ***: Document Topic Generation in Computer Science Journal's using E-ROCK***

(Enhanched -RObust Clustering using linKs)Algorithm.

Advisor : **Lailil Muflikhah S.Kom, M.Sc and Drs. Achmad Ridok, M.Kom**

Growing technology provides convenience in every aspect of life, one of which is in the world of education. With this technology, all parties can easily get access to educational material desired. But the simplicity behind it, there is confusion in the selection of the desired topic because there are many available materials. An alternative way to facilitate the search topic is to classify journal articles based on similar content. One method among many methods used to classify the data is the algorithm E-ROCK (Enhanched-robust Clustering using links). In this study, the algorithm E-ROCK (Enhanched-robust Clustering using links) is a refinement of the algorithm ROCK (Robust Clustering using links) which are used cosine similarity distance formula is believed to produce a better performance than the formula contained in the similarity distance algorithm ROCK (Robust clustering using links). This study used 100 scientific journal articles document Computer Science is divided into 10 categories. Further preprocessing and weighting process. Then do the formation of clusters using the formula goodness measure based on cosine distance similarity calculation. Obtained results showed that yield minimum error ratio of 16% with an error rate of 0.1 minimum topic formation or 10% when the threshold of 0.65. While the resulting maximum error ratio of 61% with an error rate maximum of 0.5 topic creation or 50% when the threshold of 0.15. Average error ratio values generated by agoritma E-ROCK (Enhanched-Robust Clustering using links) amounted to 32.778%.

Keywords*: article, computer science, data, clustering, E-ROCK.*

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR KODE PROGRAM

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1. Latar Belakang

Topik dapat dikatakan inti dari sebuah permasalahan yang diangkat. Topik berasal dari bahasa Yunani "*topoi*" yang berarti tempat, dalam tulis menulis yang berarti pokok pembicaraan atau sesuatu yang menjadi landasan penulisan suatu artikel [ALG-02]. Topik biasa terdiri dari satu dua kata yang singkat, dan memiliki persamaan serta perbedaan dengan tema karangan.Persamaannya adalah baik topik maupun tema keduanya sama-sama dapat dijadikan sebagai judul karangan. Sedangkan perbedaannya adalah topik masih mengandung hal yang umum, sementara tema akan lebih spesifik dan lebih terarah dalam membahas suatu permasalahan[WID-08].

Sebuah artikel mempunyai satu topik utama yang dapat menjadi landasan pengembangan dari isi artikel tersebut.Kunci keberhasilan membuat artikel terletak pada topik, bagaimana penjabarannya dan menampilkan hal-hal menarik bagi yang membacanya [ELF-12].

Jurnal merupakan salah satu jenis artikel.Jurnal adalah terbitan berkala yang berisi bahan yang sangat diminati orang saat diterbitkan. Bila dikaitkan dengan kata ilmiah di belakang kata jurnal dapat dikatakan merupakan sebuah terbitan berkala yang berbentuk pamphlet yang berisi bahan ilmiah yang diminati orang saat diterbitkan[RIF-95]. Berbagai macam topik yang terdapat dalam artikel jurnal ilmu komputer antara lain animasi, multimedia, jaringan, *fuzzy, clustering*, klasifikasi, *apriori*, dan lain sebagainya [SIL-11].

Kesulitan dalam pembentukan topik pada sejumlah artikel yang ada disebabkan karena kebingungan akan mengungkapkan gagasan yang terdapat pada sejumlah artikel tersebut. Jalan keluar yang didapatkan adalah menentukan pokok-pokok tujuan atau fokus yang terdapat dalam artikel yang telah ditetapkan.Proses pengelompokan artikel-artikel menjadi salah satu pemecahan yang ada [NUR-04].

Proses pengelompokan artikel jurnal dapat dikelompokkan ke dalam data mining, lebih khususnya adalah *text mining*.*Text mining* memiliki definisi menambang data yang berupa teks dimana sumber data biasanya didapatkan dari dokumen, dan tujuannya adalah mencari kata- kata yang dapat mewakili isi dari dokumen sehingga dapat dilakukan analisa keterhubungan antar dokumen [HAR-06]. Proses pengelompokan tersebut dinamakan dengan *clustering.*

Clustering dapat diterapkan dalam berbagai bidang, misalnya data mining, pengenalan pola, klasifikasi gambar, pemasaran, dan pengelolaan kota [BAR-06].

Proses *clustering* merupakan suatu proses pengelompokan data menjadi beberapa kelompok berdasarkan kemiripannya. Berbeda dengan klasifikasi, pengelompokan data menggunakan *clustering* dilakukan tanpa menggunakan proses pelatihan atau yang biasa disebut dengan proses *unsupervised learning* [MAN-09]. Sedangkan untuk masalah pencarian topik, digunakan metode dimana setelah didapatkan hasil *clustering* dilakukan pencarian *keywordoutput* berupa topik berdasarkan frekuensi kata terbesar yang terdapat pada tiap-tiap *cluster*.

Clustering dapat diterapkan dalam berbagai bidang, misalnya data mining, pengenalan pola, klasifikasi gambar, pemasaran, dan pengelolaan kota [BAR-06].

Proses *clustering* merupakan suatu proses pengelompokan data menjadi beberapa kelompok berdasarkan kemiripannya. Berbeda dengan klasifikasi, pengelompokan data menggunakan *clustering* dilakukan tanpa menggunakan proses pelatihan atau yang biasa disebut dengan proses *unsupervised learning* [MAN-09]. Sedangkan untuk masalah pencarian topik, digunakan metode dimana setelah didapatkan hasil *clustering* dilakukan pencarian *keywordoutput* berupa topik berdasarkan frekuensi kata terbesar yang terdapat pada tiap-tiap *cluster*. Metode *clustering* terdiri dari *sequensial clustering* dan *hierarchical clustering*. Dalam penelitian ini digunakan pengelompokan *hierarchical clustering* menggunakan algoritma *E-ROCK (Enhanched-RObust Clustering using linKs).*

Algoritma *ROCK (RObust Clustering using linKs*) merupakan salah satu dari algoritma *hierarchichal clustering* yang sering digunakan dalam proses

clustering. Dalam jurnalnya, Rizwan Ahmad [RIZ-10], mengatakan bahwa hasil pengelompokkan algoritma *ROCK (RObust Clustering using linKs*) lebih baik dibandingkan dengan algoritma yang lain, akan tetapi terdapat beberapa keterbatasan dalam penggunaan algoritma *ROCK (RObust Clustering using linKs*), yaitu dalam hal penggunakan *sparse* matriks dalam pembentukan *linkcluster* yang memiliki tingkat akurasi rendah dan waktu yang lama, selain itu digunakan koefisien *Jaccard* dalam perhitungan *similarity measure.* Menanggapi kelemahan tersebut dilakukan pendekatan algoritma *ROCK (RObust Clustering using linKs*) yang dinamakan algoritma *E-ROCK (Enhanched-RObust Clustering using linKs).* Pada algoritma *E-ROCK(Enhanched-RObust Clustering using linKs)* dilakukan perubahan yaitu menggunakan matriks *adjacency* dalam pembentukan *link* yang dinilai memiliki tingkat efektifitas lebih tinggi dibandingkan dengan *sparse* matriks. Algoritma *E-ROCK Enhanched-RObust Clustering using linKs)* menetapkan nilai *similarity value* berupa persamaan *cosine similarity* yang dinilai merupakan perhitungan yang lebih baik dibandingkan menggunakan koefisien *Jaccard*. Hasil pengelompokan menggunakan algoritma *E-ROCK Enhanched-RObust Clustering using linKs)* menunjukkan tingkat akurat yang lebih tinggi dalam hal pencarian topik dalam dokumen teks dibandingkan dengan algoritma *ROCK (RObust Clustering using linKs).*

Pada penelitian sebelumnya mengenai pembentukan topik dilakukan oleh Widhy Hayuhardika Nugraha Putra[HAY-10] yang berjudul "*Sistem Peringkas Dokumen teks Otomatis Berbahasa Indonesia dengan Metode Lexrank: Graph – Based Summarization Algorithm*" dan Dedy Prastyawan [PRA-10] yang berjudul "*Klasifikasi Really Simple Syndication dan Penetuan Trend Topik Menggunakan Algoritma Key Nearest Neighbor*". Pada algoritma *Lexrank* dilakukan peringkasan teks dengan cara mengekstrak kalimat topik dengan menghitung rangking matematis untuk tiap kalimat berdasarkan nilai *similarity* antar kalimat. Pada penelitian ini masih lemahnya penggunaan metode *stemming* yang dilakukan dalam aplikasi menjadi hal yang sangat berpengaruh terhadap data hasil uji coba, dimana stemming yang dilakukan tidak menggunakan kamus kata dasar sebagai acuannya. Kekurangan dari metode *stemming* ini akan berpengaruh terhadap nilai

similarity antar kalimat. Karena dua kata yang sebenarnya satu kata dasar tadi tidak terhitung sebagi kata atau *term* yang sama, sehingga akan mempengaruhi nilai *similarity* yang dihasilkan [HAY-10]. Pada algoritma *Key Nearest Neighbor (KNN)* dilakukan perbandingan antara dokumen baru dengan dokumen contoh yang sudah terkategori dan mencari kemiripan yang paling dekat. Pada penelitian ini proses *stemming* yang masih diperlukan dan kata-kata yang kurang optimal. Sistem akurasi tergantung dari jumlah data, semakin banyak data latih (*training*) maka sistem akan mampu menghasilkan akurasi yang lebih bagus, namun akan berbanding lurus dengan lamanya proses komputasi karena *array* yang besar. Selain itu berpengaruh pula pada proses pengukuran *similarity* [PRA-10].

Terdapat beberapa algoritma *stemming* untuk dokumen berbahasa Indonesia diantaranya adalah algoritma Nazief dan Andriani,algoritma Arifin dan Setiono, algoritma Vega, serta algoritma Ahmad, Yussof, dan Sembok. Berdasarkan penelitian yag dilakukan Jelita Asian, Hugh E.Williams, dan S.M.M Tahaghoghi, diantara algoritma-algoritma tersebut yang paling efektif adalah algoritma Nazief-Andriani yang mempunyai kebenaran *stemming* sekitar 93% [ASI-05]. Sedangkan untuk proses pengukuran kemiripan (*similarity)* terdapat empat model perhitungan, yaitu *cosine distance similarity, inner similarity, dice similarity*, dan *jaccard similarity*. Salah satu proses perhitungan kemiripan teks yang paling populer adalah *cosine distance similarity*.Hal tersebut dikarenakan *Cosine distance similarity* menghasilkan nilai kesamaan yang lebih baik dibandingkan dengan model lain. *Cosine distance similarity* mengitung sudut antar dua vektor [ADI-10].

Pada penelitian sebelumnya dilakukan oleh Rizwan & Aasia (2010), pembentukan topik menggunakan algoritma *E-ROCK Enhanched-RObust Clustering using linKs)* pada artikel website (www) berbahasa Inggris .Atas latar belakang diatas, maka penulis menetapkan algoritma tersebut dalam dokumen teks terutama artikel jurnal ilmu komputer berbahasa Indonesia.Oleh karena itu judul yang ditetapkan dalam penelitian ini adalah "**Pembentukan Topik pada Artikel Jurnal Ilmu Komputer Menggunakan Algoritma *E-ROCK(Enhanched-RObust Clustering using linKs)***".

1.2. Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan sebelumnya, dapat diambil rumusan masalah sebagai berikut:

1. Bagaimana menerapkan proses pembentukan topik dan metode pada artikel jurnal ilmu komputer berbahasa Indonesia menggunakan algoritma *E-ROCK (Enhanched-RObust Clustering using linKs*.
2. Berapa besar akurasi dari proses pembentukan topik pada artikel jurnal ilmu komputer menggunakan algoritma *E-ROCK*.

1.3. Batasan Masalah

Berdasarkan pada permaslahan yang telah diuraikan sebelumnya, batasan masalah yang digunakan adalah:

1. Inputan data latih yang digunakan adalah artikel jurnal ilmu komputer dari url http://google.com.
2. Artikel jurnal ilmu komputer yang digunakan adalah berbahasa Indonesia.
3. Untuk pengelompokan kategori topik dilakukan berdasarkan riset yang dilakukan ahli bahasa Indonesia.
4. Data latih yang digunakan adalah berupa data teks dengan format *plain text (.txt).*
5. Sebuah kata dianggap berdiri sendiri.
6. *Stemming* yang digunakan adalah *stemmingNazief-Andriani*
7. Keakuratan sistem yang mengimplementasikan algoritma *E-ROCK*akan dihitung menggunakan *error ratio.*

1.4. Tujuan

Tujuan yang ingin dicapai dalam penyusunan penelitian skripsi ini adalah :

1. Menerapkan Algoritma *E-ROCK (Enhanched-RObust Clustering using linKs)* pada proses pembentukan topik pada artikel jurnal ilmu komputer berbahasa Indonesia.
2. Menghitung hasil akurasi dan melakukan analisa dari proses pembentukan topik menggunakan algoritma *E-ROCK (Enhanched-RObust Clustering using linKs)* pada artikel jurnal ilmu komputer berbahasa Indonesia.

1.5. Manfaat

Manfaat yang diperoleh dari penyusunan penelitian skripsi ini adalah menyediakan sistem untuk menentukan topik secara otomatis pada artikel jurnal ilmu komputer berbahasa Indonesia menggunakan Algoritma *E-ROCK (Enhanched-RObust Clustering using linKs)* yang diharapkan memiliki akurasi yang tinggi.

1.6. Metodologi Penulisan

Sistematika makalah penelitian skripsi ini dibagi menjadi lima bab dengan masing-masing bab diuraikan sebagai berikut:

1. BAB I PENDAHULUAN

   Berisi latar belakang penelitian, rumusan masalah, batasan masalah, tujuan penelitian, manfaat, metode penelitian, dan sistematika penulisan.

2. BAB II TINJAUAN PUSTAKA

   Bab ini berisis teori-teori berbagai pustaka yang menunjang penelitian dalam penulisan skripsi. Adapun teori yang tercakup dalam bab ini adalah mengenai definisi dan konsep *text mining*, *clustering*, *agglomerative hierarcial clustering*, algoritma *ROCK (RObust Clustering using linKs)*, *EROCK (Enhanched-RObust Clustering using linKs)*, serta metode evaluasi.

3. BAB III METODOLOGI DAN PERANCANGAN

   Bab ini berisi tentang perancangan perangkat lunak yang dibangun, meliputi perancangan proses, perancangan tabel, dan perancanagan uji coba

4. BAB IV IMPLEMENTASI DAN PEMBAHASAN

   Bab ini berisi hasil dari implementasi perangkat lunak yang digunakan untuk mengukur hasil pembentukan topik dengan algoritma EROCK, hasil pengklasteran, serta pembahasan analisa hasil uji coba dan evaluasi hasil uji coba.

5. BAB V PENUTUP

   Bab ini berisi tentang kesimpulan dari hasil penelitian dan saran sebagai pengembangan penelitian selanjutnya.

BAB II
TINJAUAN PUSTAKA

2.1. Topik

Topik (bahasa Yunani: *topoi*) adalah inti utama dari seluruh isi tulisan yang hendak disampaikan atau lebih dikenal dengan topik pembicaraan. Topik adalah hal yang pertama kali ditentukan ketika penulis akan membuat tulisan. Topik yang masih awal tersebut, selanjutnya dikembangkan dengan membuat cakupan yang lebih sempit atau luas. Terdapat beberapa kriteria untuk sebuah topik yang dikatakan baik, diantaranya adalah topik tersebut harus mencakup keseluruhan isi tulisan, yaitu mampu menjawab pertanyaan akan masalah yang hendak ditulis. Ciri utama dari topik adalah cakupannya atas sesuatu permasalahan yang masih bersifat umum dan belum diuraikan secara mendetail [RES-07].

Topik biasa terdiri dari satu satu dua kata yang singkat dan memiliki persamaan serta perbedaan dengan tema karangan.Persamaannya adalah baik topik maupun tema keduanya sama-sama dapat dijadikan sebagai judul karangan. Sedangkan perbedaannya adalah topik masih mengandung hal yang umum, sementara tema akan lebih spesifik dan lebih terarah dalam membahas suatu permasalahan [WID-08].

2.2. Jurnal Ilmu Komputer

Jurnal merupakan salah satu dari jenis artikel (Elfitri, 2012). Jurnal Ilmu Komputer atau biasa sering disebut sebagai **jurnal informatika** merupakan media untuk mempublikasikan hasil penelitian dan pemikiran kalangan akademisi, peneliti dan praktisi bidang informatika maupun teknologi informasi komputer. Terdapat beberapa topik dalam ilmu komputer meliputi algoritma genetika, fuzzy, *artificial intelligence*, pemrosesan teks, pencitraan, dan lain sebagainya [SIL-11].

2.3. Text Mining

Text mining secara luas didefinisikan sebagai proses mencari tahu secara intensif dimana pengguna berinteraksi dengan kumpulan dokumen sepanjang waktu dengan menggunakan beberapa alat analisis. Pada cara yang sejalan dengan data *mining*, *text mining* berusaha mengutip informasi yang berguna dari sumber data melalui identifikasi dan eksplorasi pola yang menarik [FEL-07].

Text mining adalah salah satu bidang khusus dari data mining.Yang membedakan hanyalah sumber data yang digunakan.Pada data mining data yang digunakan adalah data terstruktur sedangkan dalam *text mining* data yang digunakan adalah data yang tidak terstruktur berupa teks [HAR-06].

2.4. Prepocessing

Pemrosesan suatu dokumen teks atau yang lebih dikenal dengan sebutan *preprocessing* dilakukan dalam beberapa tahapan yaitu *case folding, parsing /tokenizing , removing stopword / filtering ,* dan *stemming* [HAR-06].

a) Case Folding

Merupakan tahapan dimana dilakukan perubahan terhadap semua huruf dari a sampai z di dalam sebuah dokumen teks menjadi huruf kecil semua atau huruf besar semua. Sedangkan karakter selain huruf dihilangkan dan dianggap sebagai token, yaitu karakter dasar yang sudah tidak dapat diturunkan lagi [HAR-06].

b) Parsing / Tokenizing

Merupakan proses penguraian kalimat menjadi unit kata dengan menggunakan sintaks atau tata bahasa yang sesuai dengan ketentuan bahasa yang ada [HAR-06].

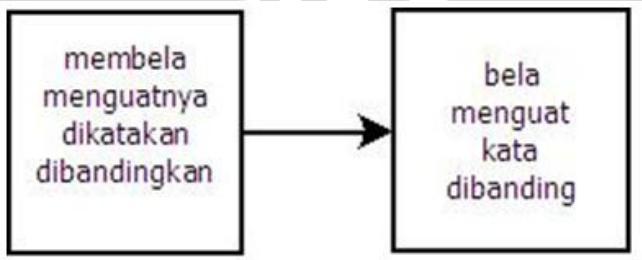
c) Removing Stopword/ Filtering

Stopword merupakan kata- kata yang sering muncul dalam dokumen namun artinya tidak deskriptif dan tidak memiliki keterkaitan dengan tema tertentu [TAL-03]. Removing stopword merupakan penghilangan *stopword* atau penghilangan kata-kata yang sering muncul yang tidak perlu (kata sandang , kata sambung , kata ganti). Daftar *stopword* dapat dilihat pada lampiran.

d) Stemming

Stemming merupakan proses yang terdapat dalam sistem information retrieval yang mentransformasi kata-kata yang terdapat pada suatu dokumen ke kata-kata akarnya *(root word)* dengan menggunakan aturan-aturan tertentu [FRA-92].

Pada *stemming* dengan menggunakan bahasa inggris yang digunakan adalah proses penghilangan sufiks sedangkan untuk stemming berbahasa Indonesia yang digunakan meliputi penghilangan sufiks, prefiks, infiks, dan konfiks [ASI-05]. Contoh untuk stemming dalam bahasa Indonesia tergambar seperti gambar 2.1.



Gambar 2. 1 Stemming [TRI-09]

Terdapat beberapa algoritma *stemming* untuk dokumen berbahasa Indonesia diantaranya adalah algoritma Nazief dan Andriani,algoritma Arifin dan Setiono, algoritma Vega, serta algoritma Ahmad, Yussof, dan Sembok. Berdasarkan penelitian yag dilakukan Jelita Asian, Hugh E.Williams, dan S.M.M Tahaghoghi, diantara algoritma-algoritma tersebut yang paling efektif adalah algoritma Nazief-Andriani yang mempunyai kebenaran *stemming* sekitar 93% [ASI-05].

2.5. Algoritma Stemming Nazief- Andriani

Tahap-tahap yang dilakukan pada algoritma Bobby Nazief dan Mirna Andriani adalah sebagai berikut:

1. Mencari kata yang akan di-*stemming* ke dalam kamus. Jika ditemukan dalam kamus dapat dikatakan bahwa kata tersebut merupakan kata dasar, dan algoritma berhenti.
2. *Inflection suffixes (-lah, -kah, -ku, -mu,* atau - *nya*) dihapus. Jika berhasil dan akhirnya adalah partikel (*-lah atau –kah*), langkah ini diulangi untuk menghapus *inflectional possessive pronoun suffixes(-ku, -mu, -nya*).

3. *Derivation suffix* (*-i* atau *–an*) dihapus. Jika berhasil, ke langkah 4. Jika langkah 3 tidak berhasil, maka :
   a. Jika *–an*telah dihapus, dan huruf terakhir adalah *–k*, kemudian *–k* juga dihapus dan dilanjutkan ke langkah 4.
   b. Jika gagal, dilanjutkan ke langkah 3b.
   c. Akhiran yang dihapus (*-i, -an,* atau *–kan*) dikembalikan, lanjut ke langkah 4.
4. *Derivational prefix* dihapus. Tahap ini mempunyai beberapa langkah, diantaranya :
   a. Jika akhiran telah dihapus pada langkah 3, kemudian kombinasi awalan-akhiran yang tidak diijinkan telah dicek menggunakan daftar pada tabel 2.1. Jika dalam kamus ditemukan maka algoritma berhenti. Jika tidak pergi ke langkah selanjutnya.

Tabel 2. 1 Kombinasi awalan dan akhiran yang tidak diijinkan

Awalan	Akhiran yang tidak diijinkan
be-	-i
di-	-an
ke-	-i, -kan
me-	-an
se-	-i, -kan
te-	-an

   b. Jika awalan saat ini sama dengan awalan yang sebelumnya, maka algoritma ini berhenti.
   c. Jika tiga awalan sebelumnya telah dihapus, maka algoritma berhenti.
   d. Tipe awalan dijelaskan pada langkah di bawah ini :
      i. Jika awalan *di, ke-*, atau *se-,* maka tipe awalan berturut-turut adalah *di, ke,* atau *se.*
      ii. Jika awalan adalah *te-* seperti yang terdapat pada tabel 2.2, awalan *be-, me-*, atau *pe-*, maka diperlukan proses tambahan untuk menentukan tipe awalannya.

Tabel 2. 2 Menentukan tipe awalan kata berawalan te-

Karakter yang mengikuti				Tipe awalan
Set1	Set2	Set3	Set4	
“-r”-	“-r-“	-	-	tidak ada
“-r-“	Huruf vokal	-	-	ter- luluh
“-r”	Bukan (huruf vokal atau “-r”)	“-er-“	huruf vokal	ter-
“-r”	Bukan (huruf vokal atau “-r”)	“-er-“	Bukan huruf vokal	ter-
“-r”	Bukan (huruf vokal atau “-r”)	Bukan “-er-“		ter-
Bukan (huruf vokal atau “-r”)	“-er-“	-		tidak ada
Bukan (huruf vokal atau “-r”)	“-er-“	Bukan huruf vokal		te-

iii. Jika dua karakter pertama bukan *di-*, *ke-*, *se-*, *te-*, *be-*, *me-* atau *pe-* maka algoritma berhenti.

e. Jika tipe awalan adalah “*none*”, maka algoritma berhenti. Jika tipe awalan tidak “*none*”, maka tipe awalan ada di tabel 2.3 awalan yang ditemukan dihapus.

Tabel 2. 3 Menentukan awalan dari tipe awalan

Awalan	Akhiran yang tidak diijinkan
di	di-
ke	ke-
se	se-
te	te-
ter	ter-
Ter-luluh	ter-

f. Jika kata dasar tidak ditemukan, langkah 4 dilakukan untuk menghapus awalan berikutnya secara berulang.

g. Melakukan *recoding*. Langkah ini tergantung pada tipe awalan, dan dapat mengakibatkan awalan yang berbeda. Jika semua langkah sudah selesai tetapi tidak berhasil, maka kata awal dianggap kata dasar [ASI-05].

2.6. Pembobotan

Pembobotan merupakan tahap memberikan bobot pada tiap-tiap *term* di dalam kamus. Metode yang paling banyak digunakan untuk melakukan pembobotan adalah *Term Frequency - Inverse Document Frequency* [SOU-03].

Term *Frequency* adalah pembobotan *term* yang didasarkan pada perhitungan jumlah term yang muncul pada suatu dokumen .Semakin tinggi nilai TF (*Term Frequency*) suatu *term* pada suatu dokumen maka semakin besar pula pengaruh kepentingan *term* terhadap dokumen tersebut.

Rumus TF ternormalisasi dihitung sebagai berikut: [GAR-06].

$$fij = \frac{tfij}{\max tfij} \quad (2.1)$$

Dengan

f_{ij} =Frekuensi ternormalisasi

tf_{ij} =Frekuensi kata *i* pada dokumen *j*

$max\ tf_{ij}$ =Frekuensi maksimum kata *i* pada dokumen *j*

Inverse *Document Frequency* (IDF) mengukur *term* yang jarang muncul pada data latih. Penilaiannya menggunakan seluruh dokumen uji yang digunakan.Jika sebuah *term* sering muncul pada data latih, maka *term* tersebut tidak bisa dianggap mewakili dokumen tersebut. Sebaliknya jika *term* jarang muncul pada data latih maka*term* tersebut dianggap memiliki hubungan yang relevan dengan dokumen.

Bobot suatu term dalam sebuah dokumen diperoleh dengan cara mengalikan nilai TF dan IDF-nya. Metode TF-IDF dapat dirumuskan melalui rumus sebagai berikut [SOU-03] :

$$W_{ij} = tf_i.idf_i \quad (2.2)$$

Dimana

W_{ij} = bobot term i pada dokumen j

tf_i = frekuensi *term* i pada dokumen j

Dengan rumus idf_i sebagai berikut [SOU-03]:

$$idf_i = Log_2 \left(\frac{N}{dfi}\right) \quad (2.3)$$

Dengan

N = banyaknya dokumen

df_i = banyaknya dokumen yang mempunyai *term* i

Sehingga rumus bobot TF-IDF dapat dituliskan sebagai berikut [SOU-03].

$$Wij = \frac{tfij}{\max tfij} = Log_2 \left(\frac{N}{dfi}\right) \quad (2.4)$$

2.7. *Term Document Matrix*

Term Document Matrix (TDM) merupakan sebuah matriks dengan isi baris dari matriks adalah kata dan kolom matriks adalah dokumen, dan isi dari matriks tersebut adalah frekuensi/pembobotan dari masing-masing kata dalam setiap dokumen. Susunan matriks tersebut bisa juga sebaliknya baris merupakan representasi dari dokumen, dan kolom matriks adalah kata [MAN-09].

	T_1	T_2	....	T_t
D_1	w_{11}	w_{21}	...	w_{t1}
D_2	w_{12}	w_{22}	...	w_{t2}
:	:	:		:
:	:	:		:
D_n	w_{1n}	w_{2n}	...	w_{tn}

Gambar 2. 2 Ilustrasi Term Document Matrix bobot [MAN-09].

Terms	Documents													
	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	M13	M14
abnormalities	0	0	0	0	0	0	0	1	0	1	0	0	0	0
age	1	0	0	0	0	0	0	0	0	0	0	1	0	0
behavior	0	0	0	0	1	1	0	0	0	0	0	0	0	0
blood	0	0	0	0	0	0	0	1	0	0	1	0	0	0
close	0	0	0	0	0	0	1	0	0	0	1	0	0	0
culture	1	1	0	0	0	0	0	1	1	0	0	0	0	0
depressed	1	0	1	1	1	0	0	0	0	0	0	0	0	0
discharge	1	1	0	0	0	1	0	0	0	0	0	0	0	0
disease	0	0	0	0	0	0	0	0	1	0	1	0	0	0
fast	0	0	0	0	0	0	0	0	0	1	0	1	1	1
generation	0	0	0	0	0	0	0	0	1	0	0	0	1	0
oestrogen	0	0	1	1	0	0	0	0	0	0	0	0	0	0
patients	1	1	0	1	0	0	0	1	0	0	0	0	0	0
pressure	0	0	0	0	0	0	0	0	0	0	1	0	0	1
rats	0	0	0	0	0	0	0	0	0	0	0	0	1	1
respect	0	0	0	0	0	0	0	1	0	0	0	1	0	0
rise	0	0	0	1	0	0	0	0	0	0	0	0	0	1
study	1	0	1	0	0	0	0	0	1	0	0	0	0	0

Gambar 2. 3 Ilustrasi Term Document Matrix pada tabel [MAN-09]

2.8. Data Kategori.

Data Kategori adalah data yang variabelnya berjenis kategori (bukan merupakan angka) atau data yang bukan merupakan hasil pengukuran (non metrik). Contoh dari variabel kategoris yaitu ras, jenis kelamin, kelompok usia, warna, tingkat pendidikan, dan lain sebagainya [HAS-08].

Dalam kutipannya Agresti [AGR-90], dijelaskan bahwa variabel adalah variabel yang skala pengukurannya terdiri dari sekumpulan kategori.Variabel ini biasanya digunakan dalm ilmu perilaku, ekologi, pendidikan, dan marketing.

Data kategori termasuk data kualitatif sehingga dapat dianalisa menggunakan rumus matematika atau statistika dan perlu diberi kode berupa angka. Analisa matematika / statistika yang digunakan adalah berdasarkan hasil perhitungan dari setiap kategori [HAS-08].

2.9. Clustering

Clustering dapat dianggap sebagai persoalan paling penting mengenai *unsupervised learning* (pembelajaran tak terawasi). Persoalan ini berhadapan dengan penemuan sebuah struktur atau pola dalam sebuah koleksi data tanpa label.

Dengan demikian *clustering* dapat dikatakan sebagai sebuah metode dimana data-data(*record*) yang serupa dikelompokkan menjadi satu.Kadang *clustering* memilik arti seperti segmentasi atau pemisahan. Data-data yang ada dipisahkan

menurut kemipripan satu sama lain. Contoh sederhana dari *clustering* adalah ketika seseorang mencuci pakaian. Pakaian–pakaian dipisahkan menurut warna-nya agar tidak terjadi pelunturan antara warna terang dan warna gelap. Oleh karena itu *cluster* adalah sebuah koleksi obyek yang anggotanya saling serupa satu sama lain dan tidak serupa dengan obyek lain di luar *cluster* tersebut [BAR-06].

a) Prinsip Dasar *Clustering*

Prinsip dasar *clustering* adalah mengklasifikasikan data-data yang dikelom-pokkan pada beberapa golongan menggunakan ukuran-ukuran tertentu, sehingga pada suatu kelompok memiliki ukuran yang berdekatan, dan untuk kelompok lainnya juga memiliki ukuran yang berbeda [KAN-03].

b) Kategori *Clustering*

Kategori pengelompokan algoritma *clustering* yang utama adalah sebagai berikut : [KAN-03].

1) *Sequensial Clustering*

Merupakan suatu teknik *clustering* yang menghasilkan *clustering* tunggal. Hasil akhir dari *clustering* ini biasanya ditentukan oleh permintaan vektor yang terdapat dalam algoritma.

2) *Hierarcial Clustering*

Merupakan suatu teknik *clustering* yang digunakan apabila belum ada informasi mengenai jumlah kelompok.Pada metode ini, data tidak langsung dikelompokkan ke dalam beberapa cluster dalam 1 tahap, tetapi dimulai dari satu *cluster* yang mempunyai kesamaan dan berjalan seterusnya.selama beberapa iterasi, sehingga terbentuk beberapa *cluster* tertentu.

- *Divisive*
  - ✓ Dari satu *cluster* ke *kcluster*.
  - ✓ Pembagian dari atas ke bawah (*top to down division*).
- *Agglomerative*
  - ✓ Dari *n cluster* ke *k cluster*.
  - ✓ Penggabungan dari bawah ke atas (*down to top division*).

2.10. Agglomerative Hierarchical Clustering

Secara umum, algoritma *Agglomerative Hierarcial Clustering* adalah sebagai berikut [RAH-04]:

1. Membuat matriks similaritas antar dokumen dengan ukuran N*N(N=jumlah dokumen).

2. Tiap dokumen dimulai dengan *cluster* berukuran satu.

3. Lakukan secara iterative sehingga hanya tersisa 1 *cluster*.

- ✓ Gabungkan dua *cluster* dengan similaritas terbesar.
- ✓ Update matriks similaritas.

4. Mengukur similaritas antar *cluster* dengan menggunakan pendekatan *single link, complete link,* atau *average link.*

2.11. Algoritma ROCK (RObust Clustering using linKs)

Algoritma *ROCK (RObust Clustering using LinKs)* merupakan salah satu jenis *agglomerative hierarcial clustering* yang mendasar pada gagasan pengelompokan pada link dan *neighbors.* Sepasang kelompok dikatakan *neighbors* jika mempunyai kemiripan sekurang-kurangnya berada dalam ambang batas Θ (*threshold*) tertentu. Banyaknya link antar sepasang adalah banyaknya *commonneighbors* atau *neighbors* bersama untuk sepasang kelompok tersebut [GUH-00].

Algoritma *ROCK (RObust Clustering using linKs)* merupakan yang yang lebih baik untuk fitur nominal dan termasuk *categorial* data daripada algoritma lain. Kualitas kelompok data yang terbentuk menggunakan Algoritma *ROCK (RObust Clustering using linKs)* jauh lebih baik dibandingkan dengan kelompok data hasil bentukan *hierarcial algorithm* berdasarkan *centroid* tradisional [GUH-00].

Langkah- langkah dalam penggunaan algoritma *ROCK (RObust Clustering using linKs)*adalah : [GUH-00].

1) Mendapatkan sampel acak poin dari sebuah set data ‘S’.
2) Menghitung nilai *link* untuk setiap pasangan titik point menggunakan rumus koefisien *Jaccard* : [GUH-00].

$$Sim\ (V_1, V_2) = \frac{|V1 \cap V2|}{|V1 \cup V2|} \quad (2.5)$$

Dengan :

V_1 : Dokumen ke-1

V_2 :Dokumen ke-2

$V_1 \cap V_2$:*Intersection object*. Elemen yang dimiliki oleh dokumen 1 dan 2

$V_1 \cup V_2$:*Union object*. Semua elemen yang terdapat dalam kedua dokumen (dokumen 1 dan 2).

3) Membentuk sparse matriks yang digunakan untuk menghubungkan setiap *cluster*.
4) Melakukan proses *clustering* menggunakan *hierarcial algorithm* pada data menggunakan object yang telah didapatkan berdasarkan koefisien *Jaccard*.
5) Menetapkan point yang tersisa terhadap *cluster* yang telah ditemukan.
6) Mengulangi langkah 1 sampai dengan 5 hingga jumlah *cluster* yang diinginkan tercapai.

Akan tetapi menurut Rizwan Ahmad [RIZ-10], terdapat batasan pada Algoritma *ROCK* (*RObust Clustering using linKs)* yaitu:

1) Algoritma *ROCK* (*RObust Clustering using linKs)* menggunakan *sparse* matriks untuk menyimpan *link – link cluster*.*Sparse* matriks membutuhkan lebih banyak ruang sehingga dikatakan tingkat efisiensinya rendah.
2) *Similarity measure* dihitung menggunakan koefisien *Jaccard*.

2.12. Algoritma E-ROCK (Enhached- RObust Clustering using linKs)

Algoritma *E-ROCK (Enhached- RObust Clustering using linKs)* merupakan perangkat tambahan yang ditemukan untuk mengatasi keterbatasan dari Algoritma *ROCK (RObust Clustering using linKs)* [RIZ-10].

Pertama, Algoritma *ROCK (RObust Clustering using links)* menggunakan sampel acak dari data, kemudian dilakukan perhitungan antar titik dalam sampel tersebut. Pada Algoritma *E-ROCK (RObust Clustering using links)* disini seluruh isi data dokumen teks yang ada akan dimanfaatkan untuk *clustering*. Setiap point data dalam data dokumen teks diperlakukan sebagai sebuah *cluster* yang terpisah yang berarti bahwa setiap dokumen diperlakukan sebagai sebuah *cluster*.

Kemudian link antara *cluster* pun dihitung. *Cluster* dengan jumlah link tertinggi kemudian digabungkan. Proses ini berlangsung sampai nomor tertentu dari *cluster* yang telah terbentuk. Jadi dengan menguraikan semua *point* yang ada dalam data dokumen teks dan seluruh topik akan menjadi lebih akurat. Yang ketiga adalah Algoritma *ROCK (RObust Clustering using linKs)* menggunakan *sparse* matriks untuk informasi link.*Sparse* matriks membutuhkan lebih banyak ruang sehingga tingkat efisiensinya cenderung rendah. Pada *E-ROCK (Enhached- RObust Clustering using linKs)* matriks *adjacency* digunakan untuk melacak kemiripan antar tetangga, yaitu pada *sparse* matriks.Matriks *adjacency* digunakan pada data yang lebih besar kapsitasnya. Selain tingkat efisiensi ruang yang lebih mudah, Algoritma *E-ROCK (Enhached- RObust Clustering using linKs)* dapat dengan mudah menemukan semua titik yang berdekatan [RIZ-10].

Langkah-langkah proses *clustering* pada Algoritma *E-ROCK (Enhached-RObust Clustering using linKs)* adalah: [RIZ-10].

1) Memanggil dokumen teks yang telah melalui proses *preprocessing*.
2) Menghitung link pada setiap dokumen dengan menggunakan rumus *cosine similarity measure* sebagai berikut : [RIZ-10].

$$CosSim\ (V_1,V_2) = \frac{\sum_{i=1}^{k} Vi1.Vi2}{|\sum_{i=1}^{k}\sqrt{vi1^2}|.||\sum_{i=1}^{k}\sqrt{vi2^2}||} \qquad (2.6)$$

Dengan :

V_1 : Dokumen ke-1

V_2 : Dokumen ke-2

$\sum_{i=1}^{k} Vi1\ .Vi2$: Jumlah antara perkalian tiap hasil nilai pembobotan antara dokumen 1 dan 2

$|\sqrt{vi1^2}|$: Akar dari hasil penjumlahan seluruh nilai pembobotan tiap term pada kelompok ke-1

$|\sqrt{vi2^2}|$ *:* Akar dari hasil penjumlahan seluruh nilai pembobotan tiap term pada dokumen ke-1.

3) Setelah perhitungan link pada semua dokumen telah selesai, tiap dokumen dianggap sebagai satu *cluster*.

4) Dilakukan pengekstraksi pada dua *cluster* yang terbaik nilai *cosine similarity measure* yang akan membentuk satu cluster dengan nama cluster baru. Prinsip ini dilakukan berdasarkan perhitungan menggunakan rumus *goodness measure* sebagai berikut: [RIZ-10].

$$g(Ci,Cj) = \frac{link\ (Ci,Cj)}{(ni+nj)^{1+2\,f(\theta)} - (ni)^{1+2\,f(\theta)} - (nj)^{1+2\,f(\theta)}} \quad (2.7)$$

Dimana :

$link\ (Ci,Cj)$ = jumlah *cross* link antar cluster C*i* dan C*j*

$g(Ci,Cj)$ = *goodnessmeasure* dari pasangan C*i* dan C*j*

n_i = Jumlah data obyek yang terdapat dalam C*i*

n_j = Jumlah data obyek yang terdapat dalam C*j*

Θ = Nilai *threshold*

$f\Theta$ $= \frac{1-\theta}{1+\theta}$

5) Dimisalkan dokumen 1 sebagai *cluster* u dan dokumen 2 sebagai *cluster* v setelah melalui proses *goodness measure*, maka: [RIZ-10].

i. Dilakukan pembentukan *cluster* baru (w), yaitu gabungan antara *cluster* u dan v dengan nilai *goodness measure* terbesar (dokumen1, dokumen2) secara hiraki.

ii. Dilakukan penghapusan *cluster* u dan v pada sistem.

iii. Menambahkan *cluster* w pada matriks *links.*

iv. Dilakukan update terhadap matriks *link.*

v. Proses 4 dan 5 diulang lagi sampai jumlah *cluster yang diinginkan* telah terbentuk.

Setelah proses *clustering* terbentuk, dilakukan pelabelan tiap cluster menggunakan frekuensi kata yang paling sering muncul pada tiap-tiap *cluster* [RIZ-10].

2.13. Pengujian Kinerja/ Evaluasi

Evaluasi sistem *clustering* dapat dilakukan menggunakan pengukuran *error ratio*. *Error ratio* menampilkan tingkat kesalahan hasil *clustering* dari sistem dibandingkan dengan label pada data.Evaluasi menggunakan metode ini hanya

dapat dilakukan pada data yang telah diberi label. *Error ratio* dapat dituliskan menggunakan persamaan sebagai berikut : [BAR-10].

$$error\ ratio = \frac{\text{jumlah data yang salah dikategorikan}}{\text{jumlah total data}} \quad (2.8)$$

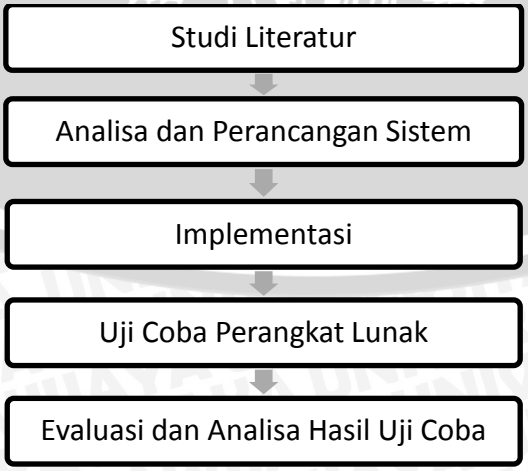
BAB III
METODOLOGI DAN PERANCANGAN

Pada bab ini akan dibahas mengenai metode dan perancangan yang digunakan serta langkah-langkah yang dilakukan untuk membuat aplikasi pembentukan topic pada artikel ilmu computer menggunakan algoritma *E-ROCK (Enhached- RObust Clustering using linKs).*

Penelitian dilakukan dengan langkah-langkah sebagai berikut:

1. Membaca dan mempelajari literatur mengenai topik, jurnal ilmu komputer, *preprocessing*, pembobotan, *clustering*, dan algoritma *E-ROCK (Enhached- RObust Clustering using linKs).*
2. Melakukan analisa dan perancangan sistem dan rumus yang telah ada untuk membangun perangkat lunak berupa sistem informasi yang digunakan untuk pembentukan topik pada artikel jurnal ilmu komputer menggunakan algoritma *E-ROCK (Enhached- RObust Clustering using linKs).*
3. Membuat sistem perangkat lunak berdasarkan analisa dan perancangan yang telah dibuat sebelumnya.
4. Pengujian metode *E-ROCK (Enhached- RObust Clustering using linKs)* dang mengumpulkan data hasil pengujian yang telah dilakukan.
5. Evaluasi hasil.

Adapun langkah- langkah tersebut di atas dapat dilihat di gambar 3.1



Gambar 3. 1 Diagram Sistem

3.1 Perancangan Sistem

3.1.1 Deskripsi Sistem

Sistem yang akan dibuat merupakan sistem yang digunakan untuk pembentukan topik pada jurnal ilmiah ilmu komputer menggunakan algoritma *E-ROCK*. Pada uji coba penelitian ini digunakan data berupa data kategori topik dalam bentuk jurnal ilmiah. Jurnal ilmiah yang digunakan adalah jurnal ilmiah mengenai materi-materi dalam ilmu komputer berbahasa Indonesia.Data diambil dari http://google.com dengan *keyword* berupa kategori-kategori topik yang terdapat dalam ilmu komputer. Untuk pelabelan data didapatkan dari data yang sebelumnya telah diberi label oleh ahli bahasa Indonesia Universitas Brawijaya.

3.1.2 Batasan Sistem

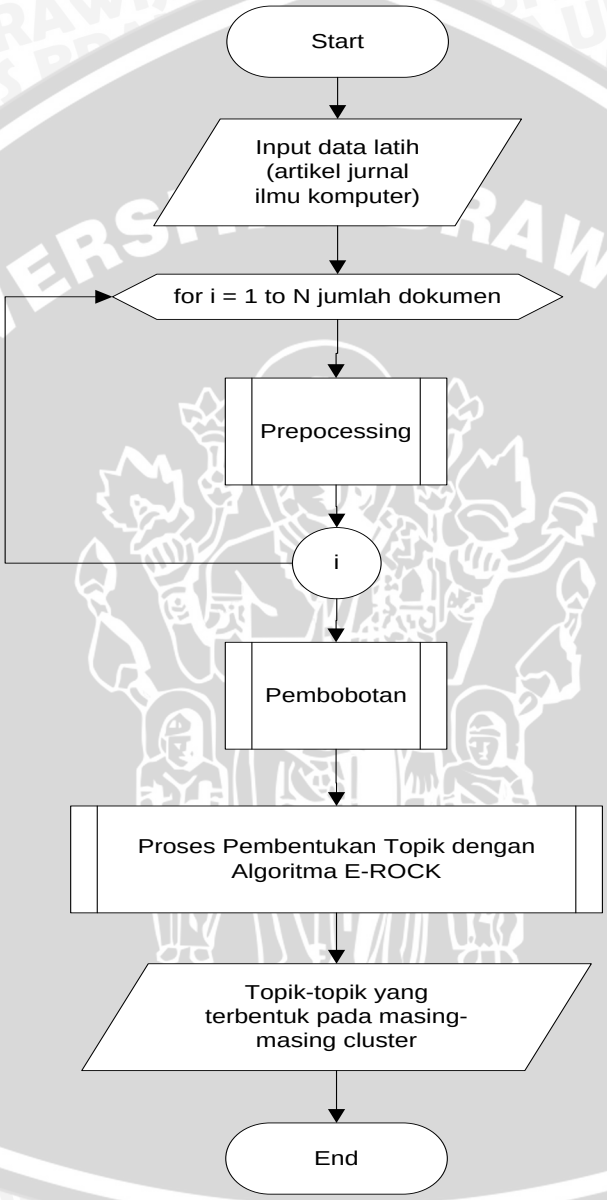
Batasan sistem yang dikembangkan adalah sebagai berikut:

1. Data yang digunakan adalah jurnal ilmiah mengenai materi-materi didalam ilmu komputer.
2. Data yang digunakan adalah data data kategorial berdasarkan topik- topik yang terdapat dalam ilmu komputer.
3. Untuk pelabelan data didapatkan dari data yang sebelumnya telah diberi label oleh ahli bahasa Indonesia Universitas Brawijaya.
4. Sistem mengabaikan data selain berbentuk teks.
5. Sistem hanya menguji kata- kata dalam artikel jurnal ilmiah yang telah dilakukan proses *preprocessing* terlebih dahulu.
6. Setelah dilakukan proses preprocessing, dilakukan proses pembobotan sebagai dasar pembentukan matriks similarity pada proses clustering menggunakan *E-ROCK (Enhached- RObust Clustering using linKs).*
7. Proses pengelompokan (*clustering*) hanya dengan menggunakan Algoritma *E-ROCK (Enhached- RObust Clustering using linKs).*
8. Rentang *threshold* yang digunakan adalah antara 0.1 sampai dengan 0.95.
9. Jumlah *cluster* ditetapkan sebanyak 10 *cluster*.

Algoritma *E-ROCK* (*Enhanched-Robust Clustering using linKs*) digunakan pada artikel ilmu komputer tersebut yang akhirnya membentuk data hasil *clustering* artikel ilmu komputer berdasarkan topik yang terbentuk. Proses

pembelajaran data dilakukan dengan menghitung kedekatan kemunculan masing-masing kata pada tiap artikel berdasarkan pada artikel-artikel yang telah ditentukan sebelumnya.Selanjutnya topik akan terbentuk berdasarkan frekuensi kata terbesar yang dimiliki oleh tiap-tiap *cluster*.

Berikut *flowchart* perancangan sistem dapat dilihat pada gambar 3.2



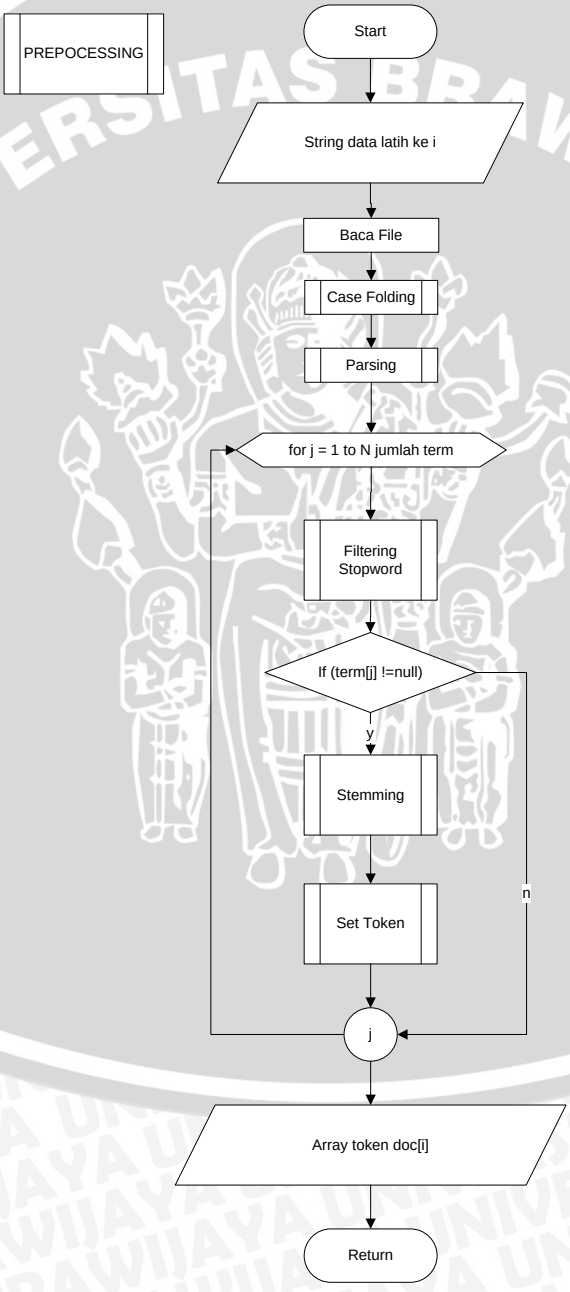
Gambar 3. 2 *Flowchart* proses perancangan sistem.

3.2 Perancangan Penelitian

Terdapat beberapa proses dalam perancangan penelitian, yaitu:

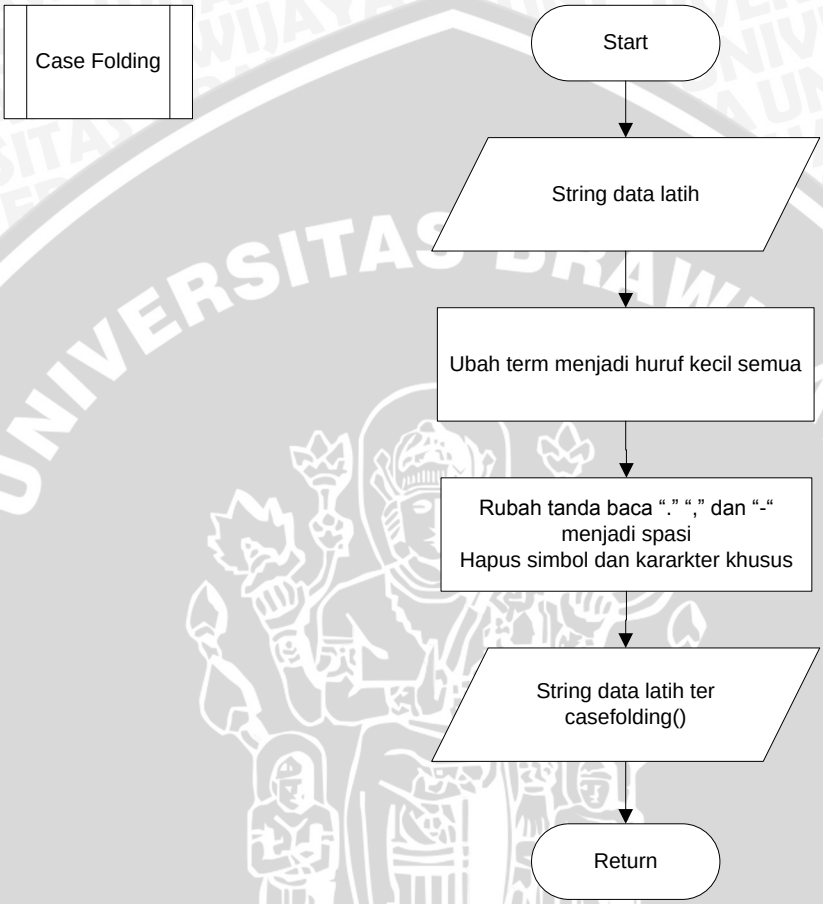
3.2.1 Proses Prepocessing

Prepocessing merupakan proses menjadikan data agar bisa diolah. *Prepocessing* meliputi *case folding,parsing, removing stop words* , dan *stemming*. *Flowchart* proses preprocessing dapat dilihat pada gambar 3.3



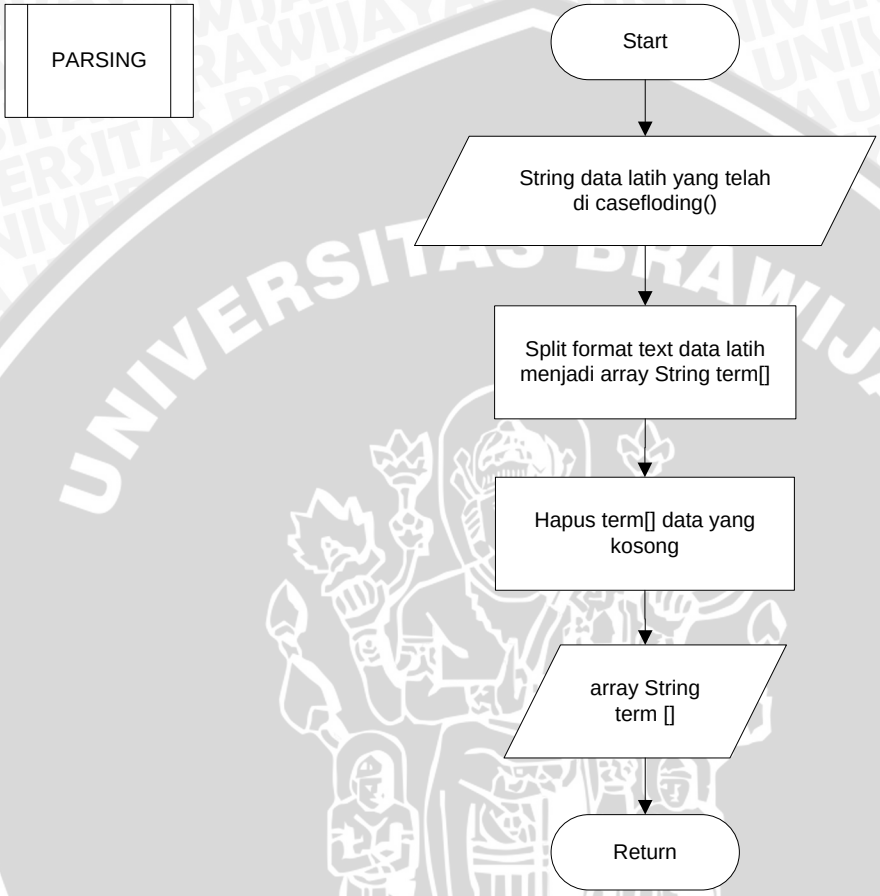
Gambar 3. 3 *Flowchart* proses preprocessing.

1. *Case folding* merupakan proses mengubah semua huruf yang terdapat dalam dokumrn menjadi huruf besar semua atau huruf kecil semua.Flowchart case folding dapat dilihat pada gambar 3.4



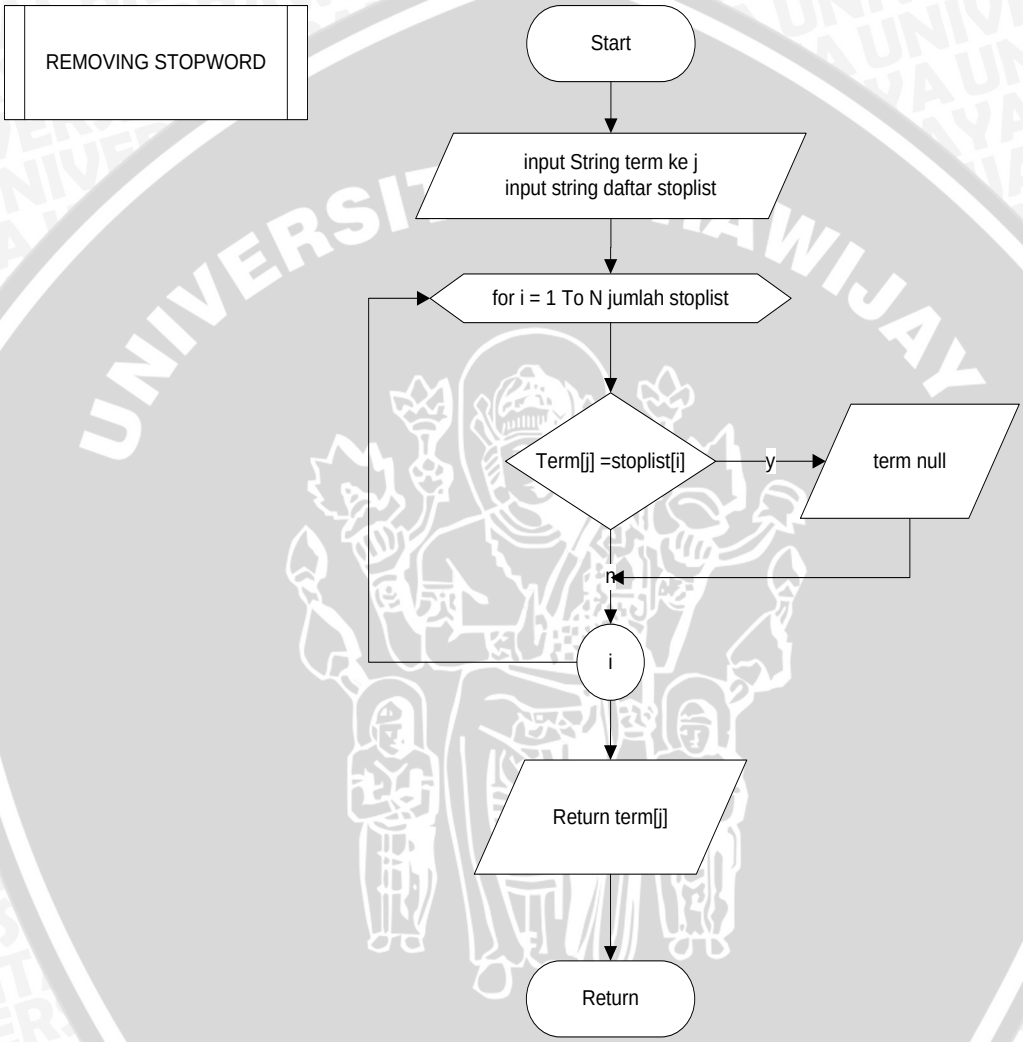
Gambar 3. 4 *Flowchart* proses case folding.

2. *Parsing* merupakan proses memilah kata-kata dalam dokumen teks menjadi per-unit kata dan menghasilkan kumpulan kata.*Flowchartparsing* dapat dilihat pada gambar 3.5.



Gambar 3. 5 *Flowchart* proses parsing.

3. *Removing Stopword* atau penghilangan *stopword* merupakan proses penghilangan kata-kata yang sering muncul dan tidak diperlukan(kata sandang, kata sambung, kata ganti).*Flowchart* removing stopword dapat dilihat gambar 3.6.



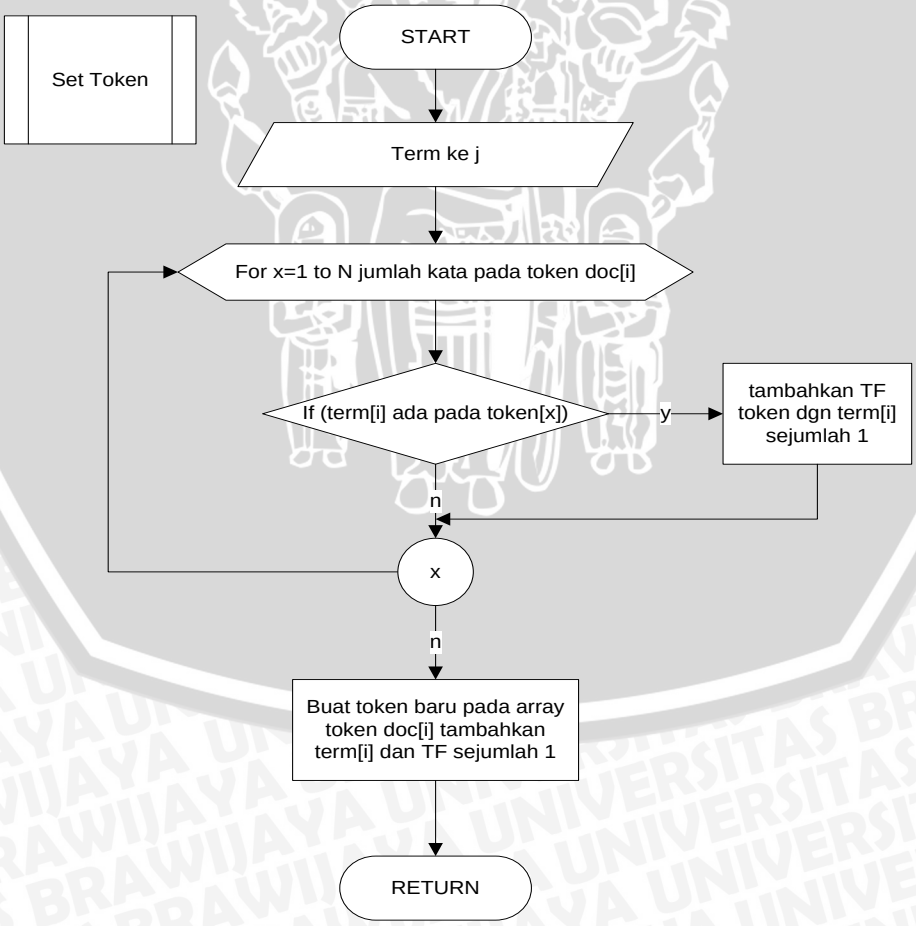
Gambar 3. 6 *Flowchart* proses removing stopword.

4. *Stemming* merupakan proses pencarian kata dasar dari kata-kata yang terdapat dalam dokumen. Stemming yang gunakan pada skripsi kali ini adalah stemming menggunakan algoritma Nazief-Andriani. Proses stemming digunakan untuk mendapatkan kata dasar daro setiap kata yang terdapat dalam sebuah dokumen.Kemudian kata-kata tersebut dicocokka dengan kamus

stemming , dimana kamus stemming merupakan kamus yang berisi kata dasar, diambil dari Kamus Besar Bahasa Indonesia *online* (KBBI *online*). Jika kata yang dimaksud cocok dengan yang terdapat di kamus maka proses selesai, jika tidak maka akan masuk kedalam proses *stemming* dengan menggunakan algoritma *Nazief-Andriani*.

3.2.2*Set Token*

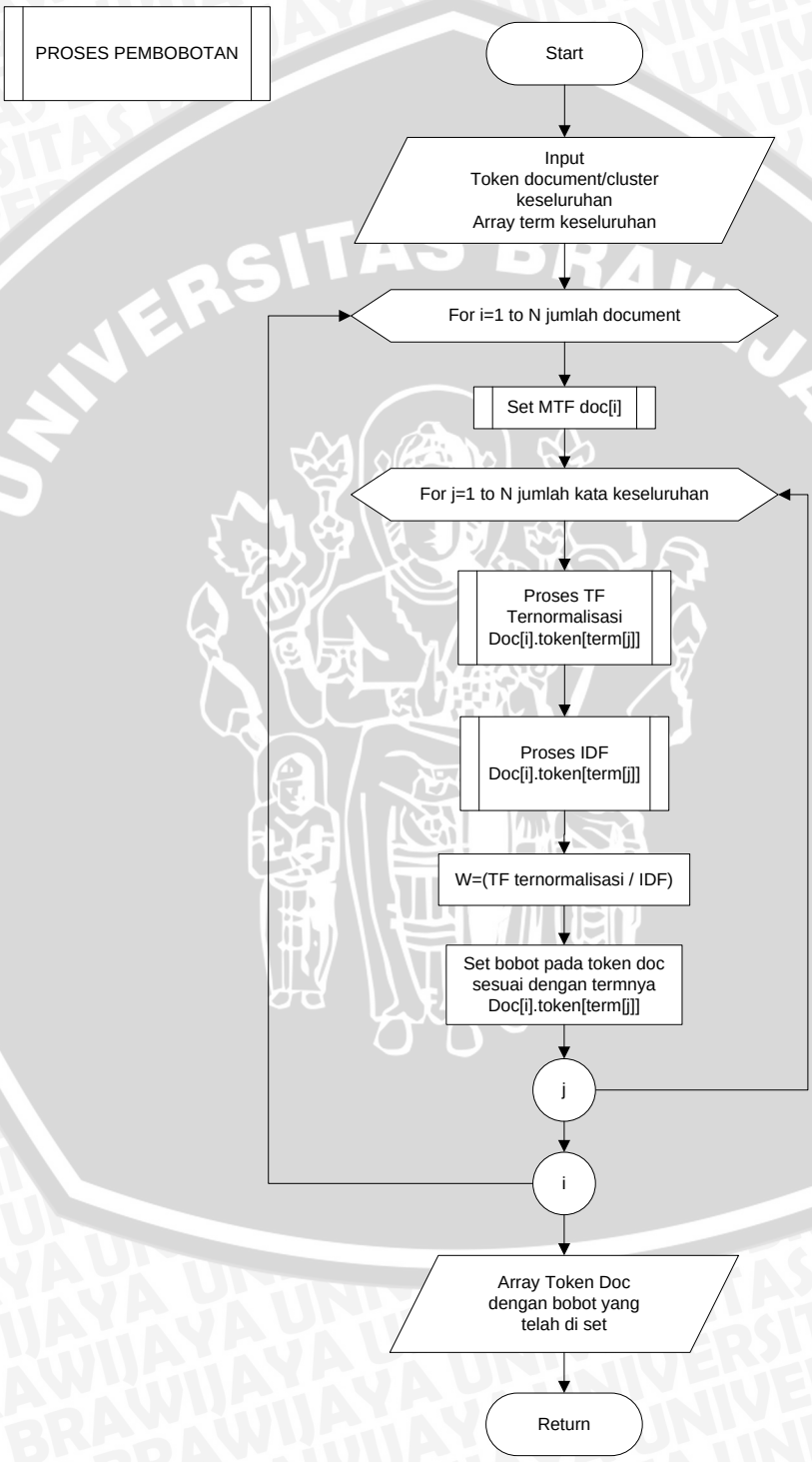
Set token merupakan proses perhitungan *term frequence (Tf)* pada dokumen.Apabila dalam *array token doc* belum terdapat *term* yang diproses, maka akan dibuat *token* baru pada *array token doc* dengan set awal *tf*=1.Akan tetapi apabila *term* tersebut sudah ada sebelumnya maka hanya akan menambahkan *tf* sejumlah 1 pada *term* yang sudah ada tersebut. *Flowchartset token* ditunjukkan pada gambar 3.7.



Gambar 3.7.*Flowchart* proses *set token.*

3.2.3 Proses Pembobotan

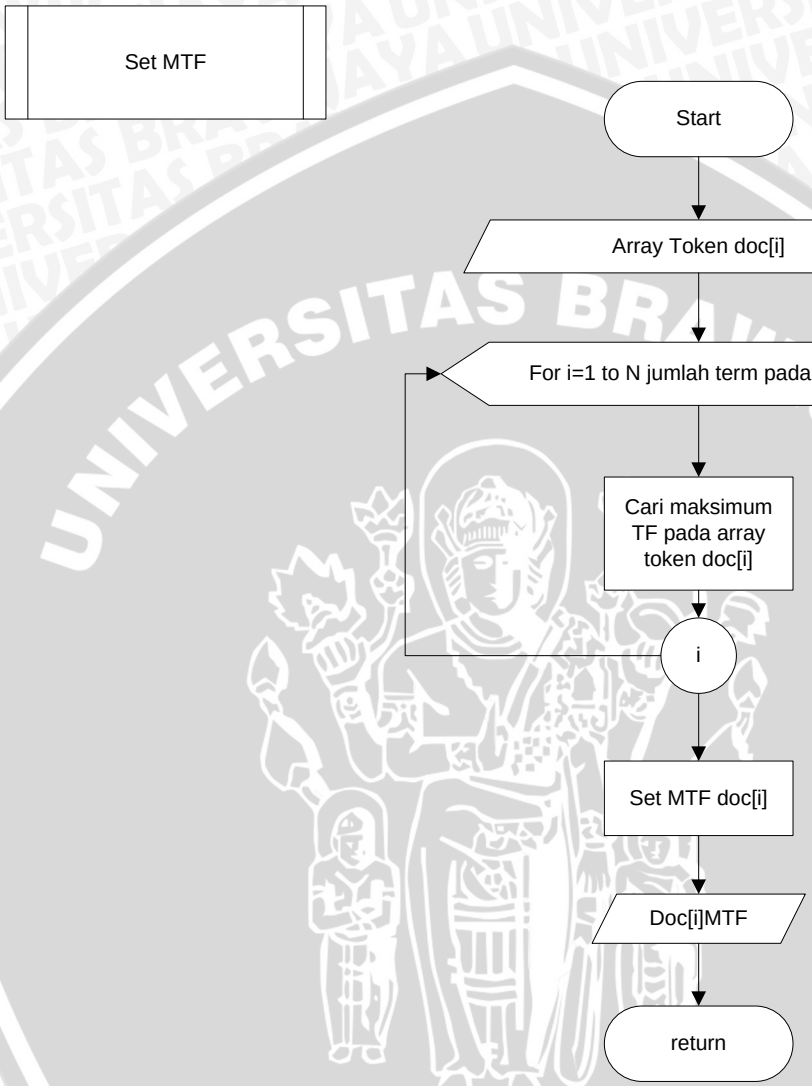
Pada proses pembobotan ini terdiri dari proses *tf*-ternormalisasi, proses *idf*, dan proses *tf-idf*. *Flowchart* untuk pembobotan kata dalam skripsi dapat dilihat pada gambar 3.8.



Gambar 3. 7 *Flowchart* proses Pembobotan.

a) Proses *Max Term Frequence (MTF).*

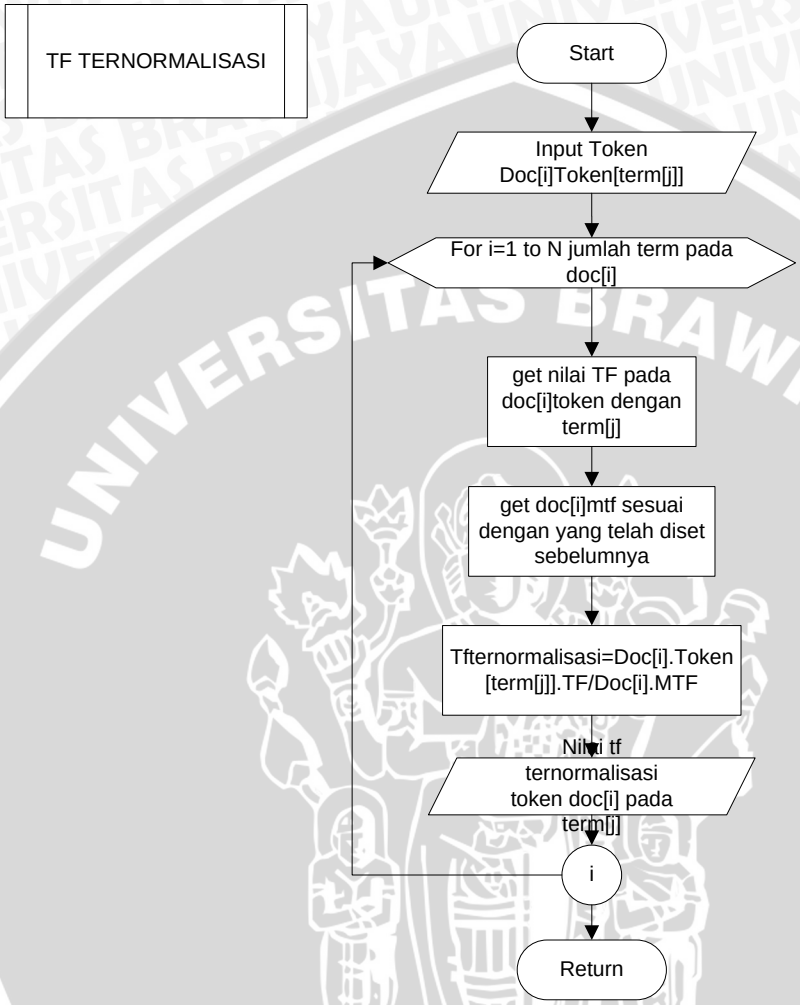
Proses *Max Term Frequence (MTF).*dapat dilihat pada *flowchart* gambar 3.9.



Gambar 3. 8 *Flowchart* proses Max Term Frequence (MTF).

b) Proses *TF*-Ternormalisasi.

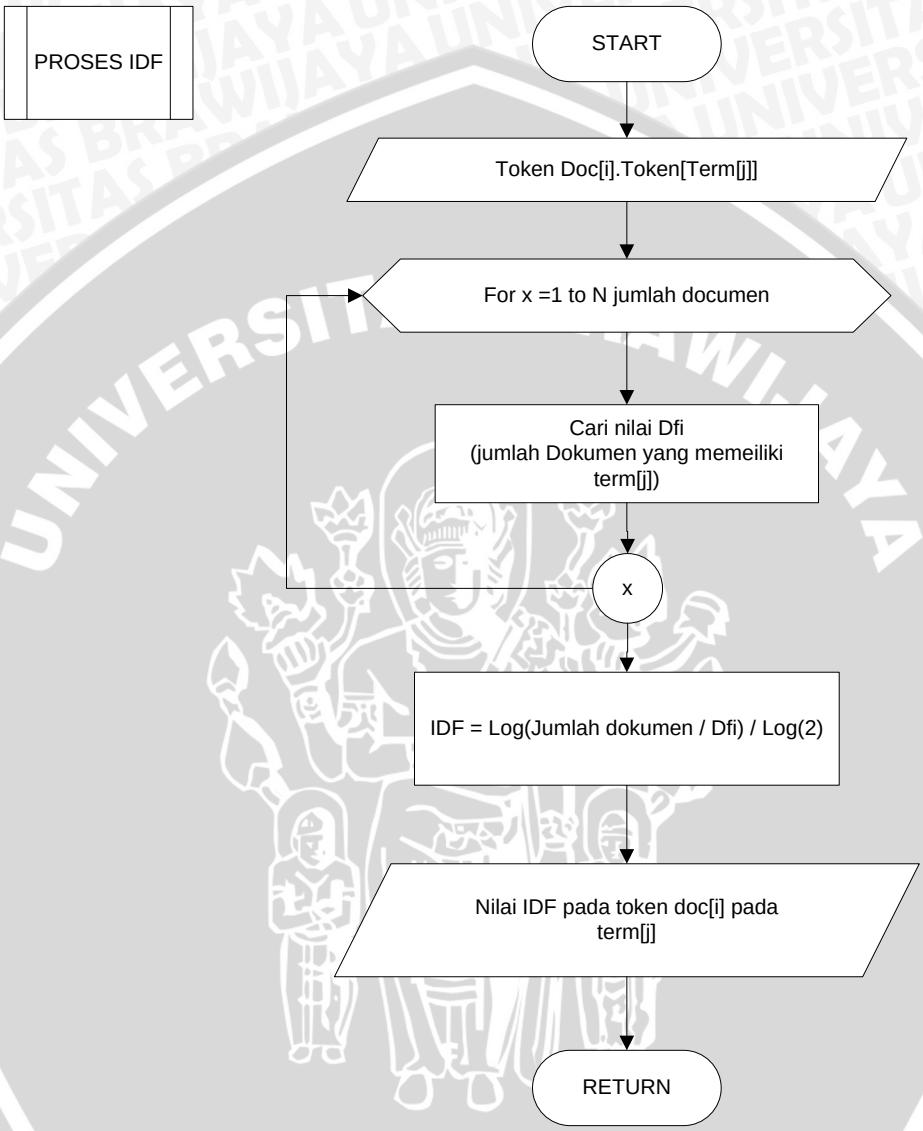
Proses *TF*-Ternormalisasi dapat dilihat pada *flowchart* gambar 3.10.



Gambar 3. 9 *Flowchart* proses TF-Ternormalisasi

c) Proses *IDF*

Proses *IDF* dapat dilihat pada *flowchart* gambar 3.11.

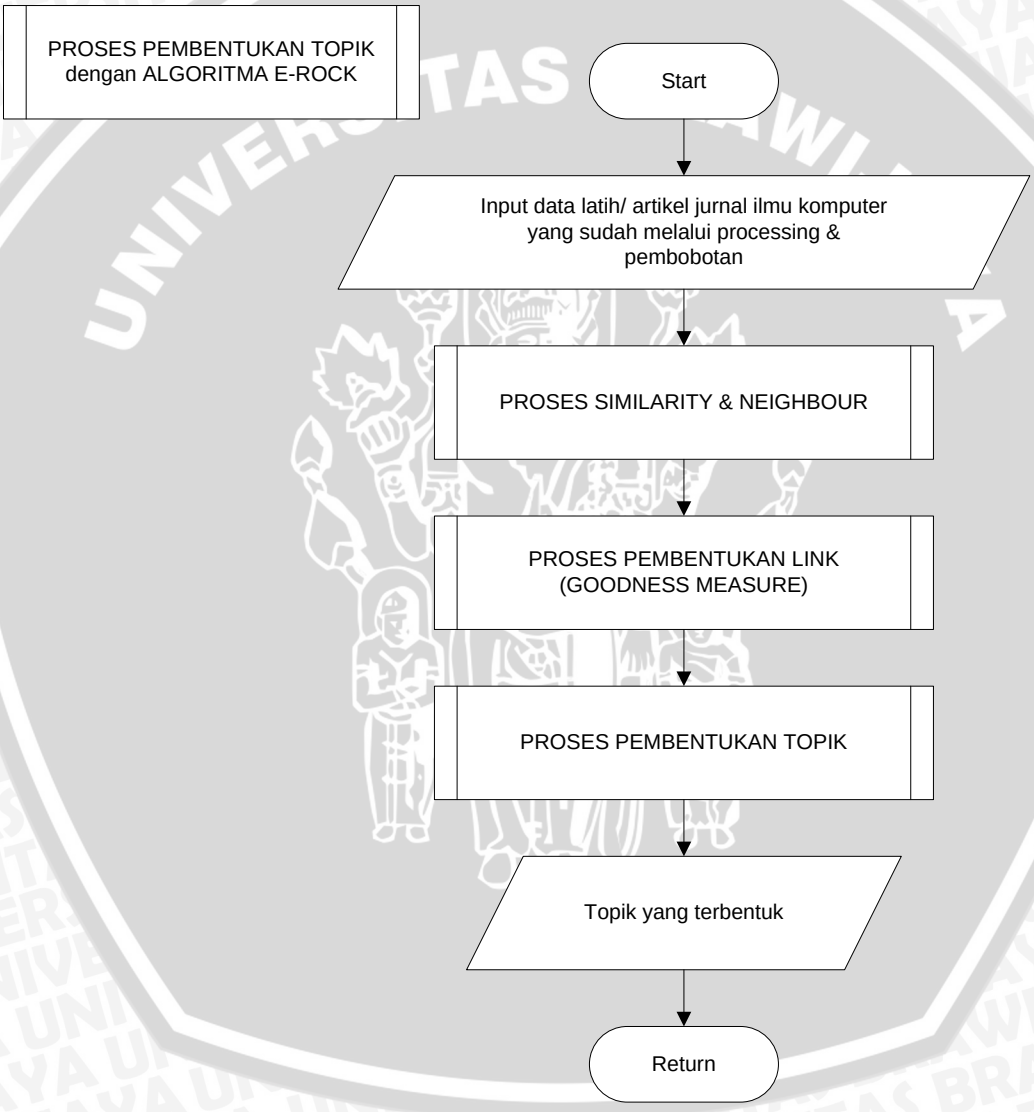


Gambar 3. 10 *Flowchart* prosesIDF

3.2.4 Proses *Clustering*

Proses *clustering* menggunakan algoritma *E-ROCK (Enhached- RObust Clustering using linKs)* melalui 3 tahapan yaitu *proses similarity, neighbor*, dan *links*.

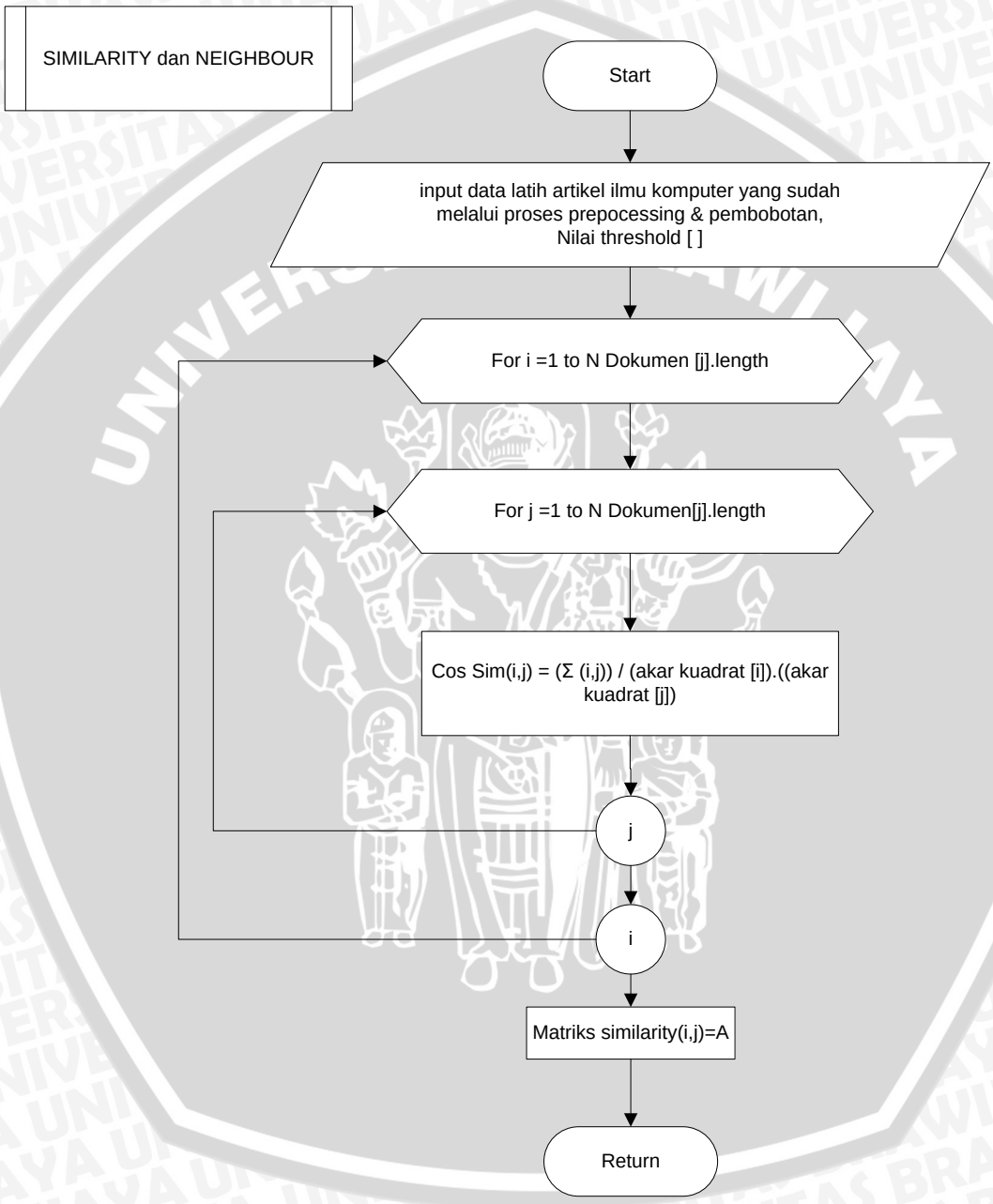
Berikut *flowchart* pembentukan topik menggunakan algoritma *E-ROCK (Enhached- RObust Clustering using linKs)* dapat dilihat pada gambar 3.12.



Gambar 3. 11 *Flowchart* proses pembentukan topik menggunakan algoritma

a) Proses *Similarity & Neighbor*

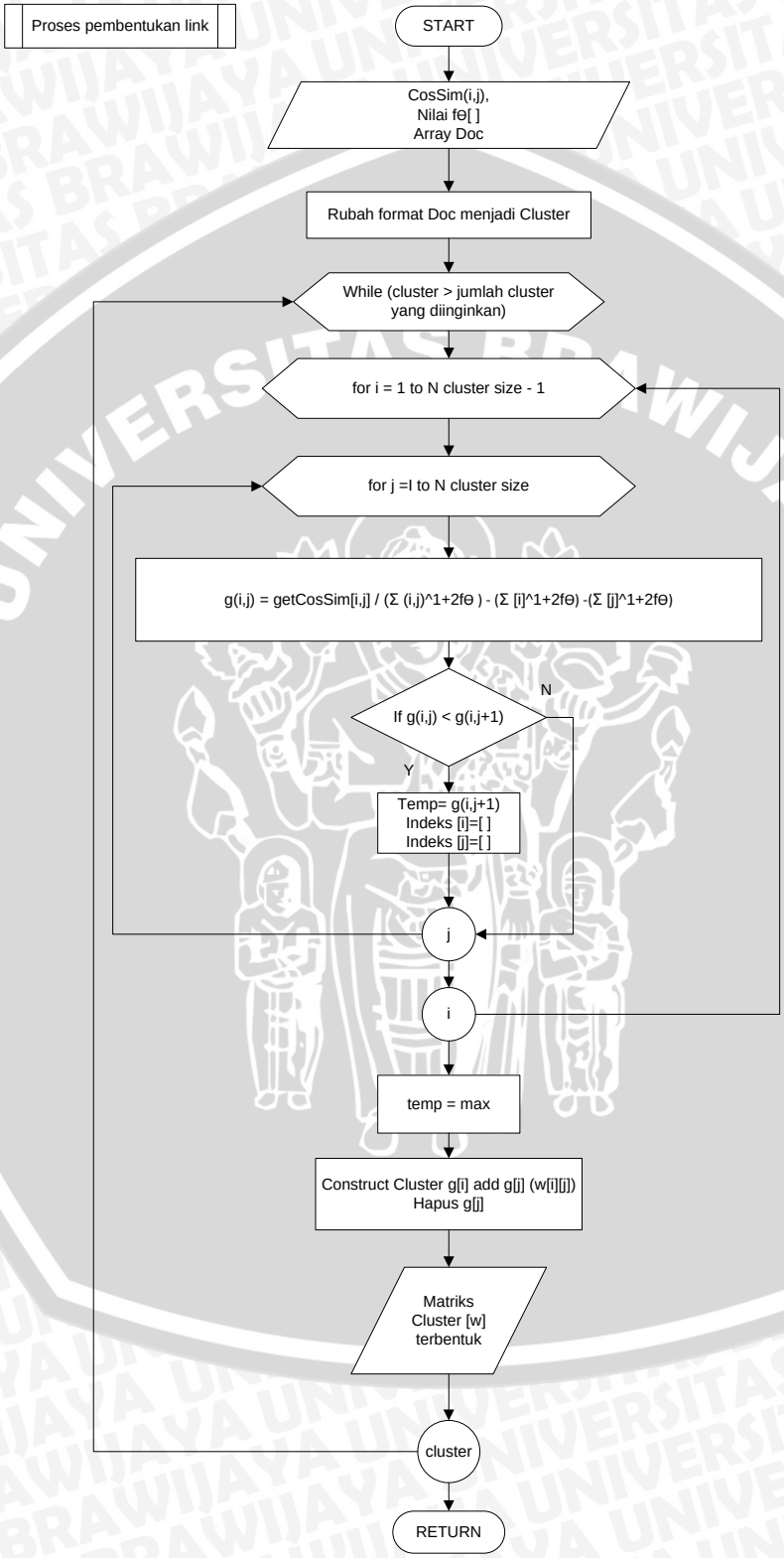
Berikut *flowchart* proses *similarity dan neighbor* dapat dilihat pada gambar 3.13.



Gambar 3. 12 *Flowchart* proses Similarity & Neighbor

b) Proses Pembentukan *link*

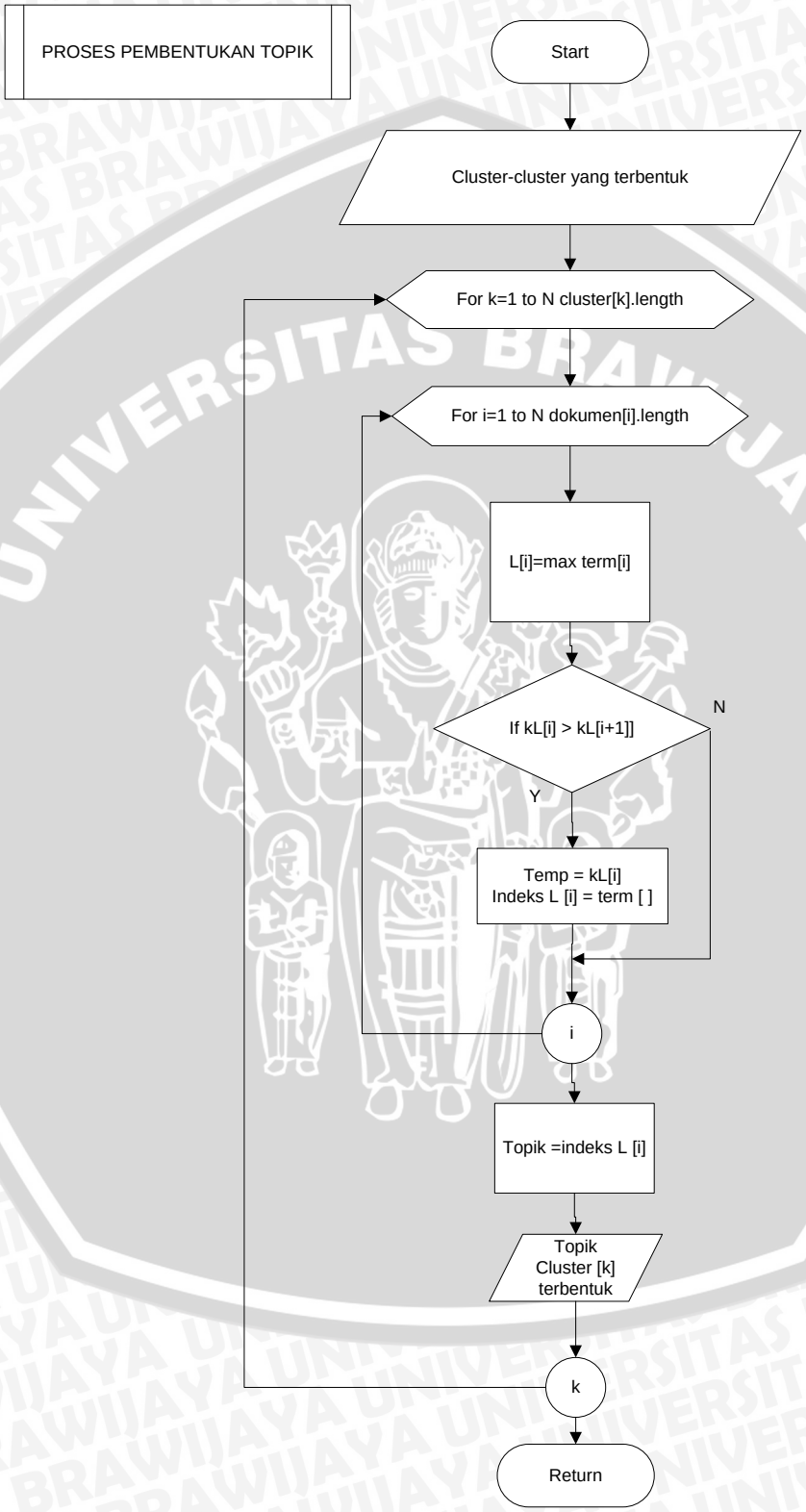
Untuk prosess pembentukan *links* dapat dilihat pada gambar 3.14.



Gambar 3. 13 *Flowchart* proses Pembentukan link.

c) Proses Pembentukan Topik

Sedangkan untuk proses pembentukan topik dapat dilihat pada gambar 3.15.



Gambar 3. 14 *Flowchart* proses pembentukan topik.

3.3 Implementasi Pada Data

3.3.1 Data

Dalam proses implementasi pada data terdapat 2 kategori topik, yaitu

a) Apriori.
b) Fuzzy.

Dalam faktanya yang termasuk kategori topik apriori adalah dokumen 1 dan 2, sedangkan yang termasuk kategori topik fuzzy adalah dokumen 3, 4, dan 5.

Sebagai contoh, berikut isi dokumen 1:

Tabel 3. 1 Dokumen 1

Sebagai salah satu aplikasi data mining, market basket analysis umumnya dilakukan dengan memakai metode Apriori. Metode ini mencari asosiasi antar item dengan hanya menghitung berapa kali item-item tersebut muncul dalam keseluruhan transaksi tanpa memperhatikan quantitas item dalam transaksi. Oleh karena itu, peneliti mengusulkan metode Fuzzy c-Covering. Fuzzy c-Covering merupakan salah satu metode yang dipakai untuk mengklasifikasikan elemen - elemen dari suatu himpunan universal menjadi partisi-partisi berupa fuzzy sets. Fuzzy c-Covering sendiri merupakan generalisasi dari metode fuzzy cpartition yang telah dikenal sebelumnya.Dari hasil pengujian, dapat disimpulkan bahwa semakin kecil minimum support dan confidence yang ditentukan, semakin banyak rule yang dapat dihasilkan dan waktu yang diperlukan semakin banyak.Selain itu, semakin tinggi jumlah kombinasi yang dicari, semakin sedikit waktu yang dibutuhkan

3.3.2 Case Folding

Proses *case folding* merupakan proses perubahan pada data /data latih menjadi huruf besar atau huruf kecil semua dan mengubah tanda baca((.),(,),(-), dll) menjadi spasi.Pada kasus ini semua huruf dijadikan huruf kecil.

Pada proses *case folding* ini diambil salah satu contoh data, yaitu dokumen 1 dapat dilihat pada tabel 3.2.

Tabel 3. 2 Proses Case Folding

sebagai salah satu aplikasi data mining market basket analysis umumnya dilakukan dengan memakai metode apriori metode ini mencari asosiasi antar item dengan hanya menghitung berapa kali item item tersebut muncul dalam keseluruhan transaksi tanpa memperhatikan quantitas item dalam transaksi oleh karena itu peneliti mengusulkan metode fuzzy c covering fuzzy c covering merupakan salah satu metode yang dipakai untuk mengklasifikasikan elemen elemen dari suatu himpunan universal menjadi partisi partisi berupa fuzzy sets fuzzy c covering sendiri merupakan generalisasi dari metode fuzzy cpartition yang telah dikenal sebelumnya dari hasil pengujian dapat disimpulkan bahwa semakin kecil minimum support dan confidence yang ditentukan semakin banyak rule yang dapat dihasilkan dan waktu yang diperlukan semakin banyak selain itu semakin tinggi jumlah kombinasi yang dicari semakin sedikit waktu yang dibutuhkan.

3.3.3 Parsing

Proses *parsing* merupakan proses pemilahan kalimat menjadi per-suku kata. Pada proses *parsing* ini diambil salah satu contoh data, yaitu dokumen 1dapat dilihat pada tabel 3.3.

Tabel 3. 3 Proses Parsing

sebagai	salah	satu
aplikasi	data	mining
market	basket	analysis
umumnya	dilakukan	dengan
memakai	metode	apriori
metode	ini	mencari
asosiasi	antar	item
tersebut	muncul	dalam
keseluruhan	transaksi	tanpa
memperhatikan	quantitas	item
dalam	transaksi	oleh
karena	itu	peneliti
mengusulkan	metode	fuzzy
c	covering	fuzzy
c	covering	merupakan
salah	satu	metode
yang	dipakai	untuk
mengklasifikasikan	elemen	elemen

dari	suatu	himpunan
universal	menjadi	partisi
partisi	berupa	fuzzy
sets	fuzzy	c
covering	sendiri	merupakan
generalisasi	dari	metode
fuzzy	cpartition	yang
telah	dikenal	sebelumnya
dari	hasil	pengujian
dapat	disimpulkan	bahwa
semakin	kecil	minimum
support	dan	confidence
yang	ditentukan	semakin
banyak	rule	yang
dapat	dihasilkan	dan
waktu	yang	diperlukan
semakin	banyak	selain
itu	semakin	tinggi
jumlah	kombinasi	yang
dicari	semakin	sedikit
waktu	yang	dibutuhkan

3.3.4 Removing Stopword

Proses *removing stopword* merupakan proses penghilangan *stopword* atau penghilangan kata-kata yang sering muncul yang tidak perlu (kata sandang , kata sambung , kata ganti). Pada proses *removing stopword* ini diambil salah satu contoh data, yaitu dokumen 1 dapat dilihat pada tabel 3.4.

Tabel 3. 4 Proses Removing Stopword

mining	market	basket
analysis	metode	apriori
item	transaksi	quantitas
peneliti	fuzzy	covering
dipakai	mengklasifikasikan	elemen
himpunan	universal	partisi
sets	generalisasi	cari
cpartition	pengujian	minimum
confidence	rule	dihasilkan
kombinasi		

3.3.5 Stemming

Pada proses *stemming* dilakukan transformasi kata-kata yang terdapat pada suatu dokumen ke kata-kata akarnya *(root word)* dengan menggunakan aturan-aturan tertentu. Pada skripsi ini digunakan *stemmingNazrief-Andriani* dapat dilihat pada tabel 3.5.

Tabel 3. 5 Proses Stemming Nazief -Andriani

Mining	market	basket
Analysis	metode	apriori
Item	transaksi	quantitas
Peneliti	fuzzy	covering
Pakai	klasifikasi	elemen
Himpun	universal	partisi
Sets	cari	generalisasi
cpartition	penguji	minimum
confidence	rule	hasil
kombinasi		

3.3.6 Proses Pembobotan

Pada proses pembobotan dilakukan beberapa langkah, yaitu:

a) Menghitung *Term Frequence* (*tf*).

Term Frequence(*Tf*) merupakan pembobotan tiap *term* yang didasarkan pada perhitungan jumlah term yang muncul pada suatu dokumen .Berikut perhitungan *Term Frequence*(*Tf*) iterasi 0 dalam tabel 3.1 pada dokumen 1 sampai dengan 5dapat dilihat pada tabel 3.6 .

Tabel 3. 6 Tabel Term Frequence (tf)

Term	*Term Frequence*(*Tf*)				
	D1	D2	D3	D4	D5
mining	3	0	0	0	1
analisis	2	0	0	0	0
perangkat	1	0	0	0	0
lunak	1	0	0	0	0
pola	1	0	0	0	0
himpun	1	0	0	0	1
peneliti	4	0	0	0	1
association	0	0	0	0	0

rule	1	0	0	0	1
implementasi	1	0	0	0	0
apriori	5	0	0	0	1
candidate	4	0	0	0	0
generation	1	0	0	0	0
sampel	1	0	0	0	0
dataset	1	0	0	0	0
atribut	1	0	0	0	0
transaksi	1	0	0	0	2
item	1	0	0	0	4
bottleneck	1	0	0	0	0
efektifitas	1	0	0	0	0
rdbms	1	0	0	0	0
oracle	1	0	0	0	0
tools	1	0	0	0	0
tkprof	1	0	0	0	0
performansi	1	0	0	0	0
query	1	0	0	3	0
operasi	1	0	0	0	0
io	1	0	0	0	0
metode	1	0	0	0	0
counting	1	0	0	0	0
pencari	0	3	0	0	0
frequent	0	5	0	0	0
itemsets	0	3	0	0	0
fp	0	5	0	0	0
growth	0	4	0	0	0
tree	0	1	0	0	0
itemset	0	2	0	0	0
database	0	2	0	2	0
paradigma	0	1	0	0	0
langkah	0	1	0	0	0
scanning	0	1	0	0	0
banding	0	2	0	0	0
kompleks	0	1	0	0	0
alternatif	0	0	1	0	0
karyawan	0	0	2	0	0
fuzzy	0	0	4	5	5
analytical	0	0	3	0	0
hierarchy	0	0	3	0	0
process	0	0	3	0	0
subjektifitas	0	0	1	0	0
kriteria	0	0	1	3	0
konvensional	0	0	1	0	0
konsep	0	0	1	0	0

pakai	0	0	1	0	1
ahp	0	0	2	0	0
model	0	0	1	0	0
bobot	0	0	1	0	0
nonadditive	0	0	1	0	0
sistem	0	0	0	1	0
pasticrip	0	0	0	1	0
deterministic	0	0	0	1	0
presisi	0	0	0	1	0
samar	0	0	0	2	0
variable	0	0	0	4	0
logika	0	0	0	2	0
proyek	0	0	0	1	0
bernilai	0	0	0	2	0
informasi	0	0	0	2	0
kinerja	0	0	0	3	0
pegawai	0	0	0	5	0
absensi	0	0	0	1	0
ambigu	0	0	0	1	0
derajat	0	0	0	1	0
keanggotaan	0	0	0	1	0
tiap	0	0	0	1	0
market	0	0	0	0	1
basket	0	0	0	0	1
analysis	0	0	0	0	1
asosiasi	0	0	0	0	1
quantitas	0	0	0	0	1
covering	0	0	0	0	3
klasifikasi	0	0	0	0	1
elemen	0	0	0	0	2
universal	0	0	0	0	1
partisi	0	0	0	0	2
sets	0	0	0	0	1
generalisasi	0	0	0	0	1
cpartition	0	0	0	0	1
penguji	0	0	0	0	1
minimum	0	0	0	0	1
confidence	0	0	0	0	1
hasil	0	0	0	0	1
kombinasi	0	0	0	0	1
cari	0	0	0	0	1

b) Menghitung *Max Term Frequence* (*Max tfi*).

Max tfi merupakan nilai dari *term frequence(tf)* yang sering muncul pada sebuah dokumen. Berdasarkan pengertian tersebut diatas didapatkan*Max tfi* tiap-tiap dokumen pada iterasi 0 sebagai berikut yang terlihat pada tabel 3.7.

Tabel 3. 7 Max Term Frequence (Max tfi)

Max tfi				
D1	D2	D3	D4	D5
5	5	4	5	5

c) Menghitung *Document Frequence* (*dfi*).

Document Frequence (*dfi)* merupakan pengukuran jumlah *term* yang muncul pada data latih. Penilaiannya menggunakan seluruh dokumen uji yang digunakan. Berdasarkan pengertian tersebut diatas didapatkan*Document Frequence* (*dfi)* tiap-tiap *term* pada iterasi 0 sebagai berikut yang terlihat pada tabel 3.8.

Tabel 3. 8 Document Frequence (dfi)

Term	Jumlah dokumen yang terdapat term *i*
Mining	2
Analisis	1
perangkat	1
lunak	1
pola	1
himpun	2
peneliti	2
association	1
rule	2
implementasi	1
apriori	2
candidate	1
generation	1
sampel	1
dataset	1
atribut	1
transaksi	2
item	2
bottleneck	1
efektifitas	1

rdbms	1
oracle	1
tools	1
tkprof	1
performansi	1
query	2
operasi	1
io	1
metode	1
counting	1
pencari	1
frequent	1
itemsets	1
fp	1
growth	1
tree	1
itemset	1
database	2
paradigma	1
langkah	1
scanning	1
banding	1
kompleks	1
alternatif	1
karyawan	1
fuzzy	3
analytical	1
hierarchy	1
process	1
subjektifitas	1
kriteria	2
Konvensional	1
Konsep	1
Pakai	2
Ahp	1
Model	1
Bobot	1
Nonadditive	1
System	1
Pasticrip	1
Deterministic	1
Presisi	1
Samar	1
Variable	1
Logika	1

Proyek	1
Bernilai	1
Informasi	1
Kinerja	1
Pegawai	1
Absensi	1
Ambigu	1
Derajat	1
Keanggotaan	1
Tiap	1
Market	1
Basket	1
Analysis	1
Asosiasi	1
Quantitas	1
Covering	1
Klasifikasi	1
Elemen	1
Universal	1
Partisi	1
Sets	1
Generalisasi	1
Cpartition	1
penguji	1
minimum	1
confidence	1
hasil	1
kombinasi	1
cari	1

d) Menghitung t*f* Ternormalisasi.

Berdasarkan rumus 2.1 yang terdapat pada bab 2, pada tabel 3.9 didapatkan perhitungan, Sebagai contoh adalah pada kata ‘algoritma’, didapatkan nilai *tf- ternormalisasi* iterasi 0 sebagai berikut :

$$tf - ternormalisasi(\ mining) = \frac{tfij}{\max tfij} = \frac{3}{5}$$

Tabel 3. 9 Tabel Term Frequence(tf)-Ternormalisasi

Term	*Term Frequence* (*Tf*) - Ternormalisasi				
	D1	D2	D3	D4	D5
mining	3/5	0	0	0	1/5
analisis	2/5	0	0	0	0
perangkat	1/5	0	0	0	0
lunak	1/5	0	0	0	0
pola	1/5	0	0	0	0
himpun	1/5	0	0	0	1/5
peneliti	4/5	0	0	0	1/5
association	0	0	0	0	0
rule	1/5	0	0	0	1/5
implementasi	1/5	0	0	0	0
apriori	1	0	0	0	1/5
candidate	4/5	0	0	0	0
generation	1/5	0	0	0	0
sampel	1/5	0	0	0	0
dataset	1/5	0	0	0	0
atribut	1/5	0	0	0	0
transaksi	1/5	0	0	0	2/5
item	1/5	0	0	0	4/5
bottleneck	1/5	0	0	0	0
efektifitas	1/5	0	0	0	0
rdbms	1/5	0	0	0	0
oracle	1/5	0	0	0	0
tools	1/5	0	0	0	0
tkprof	1/5	0	0	0	0
performansi	1/5	0	0	0	0
query	1/5	0	0	3/5	0
operasi	1/5	0	0	0	0
io	1/5	0	0	0	0
metode	1/5	0	0	0	0
counting	1/5	0	0	0	0
pencari	0	3/5	0	0	0
frequent	0	1	0	0	0
itemsets	0	3/5	0	0	0
fp	0	1	0	0	0
growth	0	4/5	0	0	0
tree	0	1/5	0	0	0
itemset	0	2/5	0	0	0
database	0	2/5	0	2/5	0
paradigma	0	1/5	0	0	0
langkah	0	1/5	0	0	0

scanning	0	1/5	0	0	0
banding	0	2/5	0	0	0
kompleks	0	1/5	0	0	0
alternatif	0	0	¼	0	0
karyawan	0	0	½	0	0
fuzzy	0	0	1	1	1
analytical	0	0	¾	0	0
hierarchy	0	0	¾	0	0
process	0	0	¾	0	0
subjektifitas	0	0	¼	0	0
kriteria	0	0	¼	3/5	0
konvensional	0	0	¼	0	0
konsep	0	0	¼	0	0
pakai	0	0	¼	0	1/5
ahp	0	0	½	0	0
model	0	0	¼	0	0
bobot	0	0	¼	0	0
nonadditive	0	0	¼	0	0
sistem	0	0	0	1/5	0
pasticrip	0	0	0	1/5	0
deterministic	0	0	0	1/5	0
presisi	0	0	0	1/5	0
samar	0	0	0	2/5	0
variable	0	0	0	4/5	0
logika	0	0	0	2/5	0
proyek	0	0	0	1/5	0
bernilai	0	0	0	2/5	0
informasi	0	0	0	2/5	0
kinerja	0	0	0	3/5	0
pegawai	0	0	0	1	0
absensi	0	0	0	1/5	0
ambigu	0	0	0	1/5	0
derajat	0	0	0	1/5	0
keanggotaan	0	0	0	1/5	0
tiap	0	0	0	1/5	0
market	0	0	0	0	1/5
basket	0	0	0	0	1/5
analysis	0	0	0	0	1/5
asosiasi	0	0	0	0	1/5
quantitas	0	0	0	0	1/5
covering	0	0	0	0	3/5
klasifikasi	0	0	0	0	1/5
elemen	0	0	0	0	2/5
universal	0	0	0	0	1/5
partisi	0	0	0	0	2/5

sets	0	0	0	0	1/5
generalisasi	0	0	0	0	1/5
cpartition	0	0	0	0	1/5
penguji	0	0	0	0	1/5
minimum	0	0	0	0	1/5
confidence	0	0	0	0	1/5
hasil	0	0	0	0	1/5
kombinasi	0	0	0	0	1/5
cari	0	0	0	0	1/5

e) Menghitung *Inverse Document Frequency* (*IDF*)

Inverse Document Frequency (*IDF*) mengukur *term* yang jarang muncul pada data latih. Penilaiannya menggunakan seluruh dokumen uji yang digunakan. Berdasarkan rumus 2.3 didapatkan perhitungan.Sebagai contoh adalah pada kata 'mining', didapatkan nilai *IDF* sebagai berikut:

$$IDFmining = Log_2\left(\frac{N}{dfmining}\right)$$

$$= log_2\left(\frac{5}{2}\right) = 1.3219$$

Nilai *Inverse Document Frequency* (*IDF*) iterasi 0 dapat dilihat pada tabel 3.10 sebagai berikut:

Tabel 3. 10 Tabel Inverse Document Frequency (IDF)

Term	Jumlah dokumen yang terdapat term *i*	*IDF*
mining	2	1.3219
analisis	1	2.3219
perangkat	1	2.3219
lunak	1	2.3219
pola	1	2.3219
himpun	2	1.3219
peneliti	2	1.3219
association	1	2.3219
rule	2	1.3219
implementasi	1	2.3219
apriori	2	1.3219
candidate	1	2.3219
generation	1	2.3219
sampel	1	2.3219

dataset	1	2.3219
atribut	1	2.3219
transaksi	2	1.3219
item	2	1.3219
bottleneck	1	2.3219
efektifitas	1	2.3219
rdbms	1	2.3219
oracle	1	2.3219
tools	1	2.3219
tkprof	1	2.3219
performansi	1	2.3219
query	2	1.3219
operasi	1	2.3219
io	1	2.3219
metode	1	2.3219
counting	1	2.3219
pencari	1	2.3219
frequent	1	2.3219
itemsets	1	2.3219
fp	1	2.3219
growth	1	2.3219
tree	1	2.3219
itemset	1	2.3219
database	2	1.3219
paradigma	1	2.3219
langkah	1	2.3219
scanning	1	2.3219
banding	1	2.3219
kompleks	1	2.3219
alternatif	1	2.3219
karyawan	1	2.3219
fuzzy	3	0.7369
analytical	1	2.3219
hierarchy	1	2.3219
process	1	2.3219
subjektifitas	1	2.3219
kriteria	2	1.3219
konvensional	1	2.3219
konsep	1	2.3219
pakai	2	1.3219
ahp	1	2.3219
model	1	2.3219
bobot	1	2.3219
nonadditive	1	2.3219
sistem	1	2.3219

pasticrip	1	2.3219
deterministic	1	2.3219
presisi	1	2.3219
samar	1	2.3219
variable	1	2.3219
logika	1	2.3219
proyek	1	2.3219
bernilai	1	2.3219
informasi	1	2.3219
kinerja	1	2.3219
pegawai	1	2.3219
absensi	1	2.3219
ambigu	1	2.3219
derajat	1	2.3219
keanggotaan	1	2.3219
tiap	1	2.3219
market	1	2.3219
basket	1	2.3219
analysis	1	2.3219
asosiasi	1	2.3219
quantitas	1	2.3219
covering	1	2.3219
klasifikasi	1	2.3219
elemen	1	2.3219
universal	1	2.3219
partisi	1	2.3219
sets	1	2.3219
generalisasi	1	2.3219
cpartition	1	2.3219
penguji	1	2.3219
minimum	1	2.3219
confidence	1	2.3219
hasil	1	2.3219
kombinasi	1	2.3219
cari	1	2.3219

f) Menghitung *TF-IDF*

Berdasarkan rumus 2.4 pada bab 2, didapatkan perhitungan. Sebagai contoh adalah pada kata 'mining', didapatkan nilai *TF-IDF* sebagai berikut:

$TF\text{-}IDF_{mining} = tf_{mining}\text{-}ternormailsasi.idf_{mining}$

= 3/5* 1.3219

= 0.79315

Pada tabel 3.11 didapatkan perhitungan *TF-IDF* iterasi 0 sebagai berikut:

Tabel 3. 11 Tabel TF-IDF

Term	*TF-IDF*				
	D1	D2	D3	D4	D5
mining	0.79315	0	0	0	0.2643
analisis	0.92877	0	0	0	0
perangkat	0.46438	0	0	0	0
lunak	0.46438	0	0	0	0
pola	0.46438	0	0	0	0
himpun	0.2643	0	0	0	0.2643
peneliti	1.0575	0	0	0	0.2643
association	0.46438	0	0	0	0
rule	0.2643	0	0	0	0.2643
implementasi	0.46438	0.7369	0	0	0
apriori	0.7369	0.2643	0	0	0.1473
candidate	1.0575	0.2643	0	0	0
generation	1.0575	0	0	0	0
sampel	0.46438	0	0	0	0
dataset	0.46438	0	0	0	0
atribut	0.46438	0	0	0	0
transaksi	0.2643	0	0	0	0.52877
Item	0.2643	0	0	0	1.0575
bottleneck	0.46438	0	0	0	0
efektifitas	0.46438	0	0	0	0
rdbms	0.46438	0	0	0	0
oracle	0.46438	0	0	0	0
tools	0.46438	0	0	0	0
tkprof	0.46438	0	0	0	0
performansi	0.46438	0	0	0	0
query	0.2643			0.79315	0
operasi	0.46438	0	0	0	0
Io	0.46438			0.14793	0
metode	0.14739	0	0	0	0.7369

counting	0.46438	0	0	0	0
pencari	0	1.3931	0	0	0
frequent	0	2.3219	0	0	0
itemsets	0	1.3931	0	0	0
Fp	0	2.3219	0	0	0
growth	0	1.8575	0	0	0
Tree	0	0.46438	0	0	0
itemset	0	0.928	0	0	0
database	0	0.52877	0	0.528711	0
paradigma	0	0.46438	0	0	0
langkah	0	0.46438	0	0	0
scanning	0	0.46438	0	0	0
banding	0	0.92877	0	0	0
kompleks	0	0.46438	0	0	0
alternatif	0	0	0.5804	0	0
karyawan	0	0	1.1609	0	0
fuzzy	0	0	0.7369	0.7369	0.7369
analytical	0	0	1.71444	0	0
hierarchy	0	0	1.71444	0	0
process	0	0	1.71444	0	0
subjektifitas	0	0	0.5804	0	0
kriteria	0	0	0.3304	0.79315	0
konvensional	0	0	0.5804	0	0
konsep	0	0	0.5804	0	0
pakai	0	0	0.3304	0	0.2643
ahp	0	0	1.1609	0	0
model	0	0	0.5804	0	0
bobot	0	0	0.5804	0	0
nonadditive	0	0	0.5804	0	0
sistem	0	0	0	0.4643	0
pasticrip	0	0	0	0.4643	0
deterministic	0	0	0	0.4643	0
presisi	0	0	0	0.4643	0
samar	0	0	0	0.92877	0
variable	0	0	0	1.8575	0
logika	0	0	0	0.92877	0
proyek	0	0	0	0.4643	0
bernilai	0	0	0	0.92877	0
informasi	0	0	0	1.3219	0
kinerja	0	0	0	2.3219	0
pegawai	0	0	0	0.4643	0
absensi	0	0	0	0.4643	0
ambigu	0	0	0	0.4643	0
derajat	0	0	0	0.4643	0
keanggotaan	0	0	0	0.4643	0

tiap	0	0	0	0.4643	0
market	0	0	0	0	0.4643
basket	0	0	0	0	0.4643
analysis	0	0	0	0	0.4643
asosiasi	0	0	0	0	0.4643
quantitas	0	0	0	0	0.4643
covering	0	0	0	0	1.3931
klasifikasi	0	0	0	0	0.4643
elemen	0	0	0	0	0.92877
universal	0	0	0	0	0.4643
partisi	0	0	0	0	0.92877
sets	0	0	0	0	0.4643
generalisasi	0	0	0	0	0.4643
cpartition	0	0	0	0	0.4643
penguji	0	0	0	0	0.4643
minimum	0	0	0	0	0.4643
confidence	0	0	0	0	0.4643
hasil	0	0	0	0	0.4643
kombinasi	0	0	0	0	0.4643
Cari	0	0	0	0	0.4643

3.3.7 Menghitung Similarity Value & Neighbor

Setelah didapatkan nilai TF-IDF, maka dilakukan pembentukan matriks *similarity measure dan neighbor* menggunakan rumus 2.6, sehingga didapatkan perhitungan sebagai berikut,

Sebagai contoh untuk perhitungan (D, D2) didapatkan

(V_{D1}^2) = nilai *TF-IDF* term yang dikuadratkan yang terdapat pada dokumen 1.

(V_{D2}^2) = nilai *TF-IDF* term yang dikuadratkan yang terdapat pada dokumen 2.

(D1,D2)= nilai kesamaan tiap term yang terdapat pada dokumen 1 dan dokumen 2 berdasarkan nilai *TF-IDF* tiap term

Sehingga apabila diaplikasikan pada rumus 2.6, yaitu :

$$\textit{Cos Sim}\ (D1,D2) = \frac{\sum(D1,D2)}{\sqrt{D1.D1}\ .\ \sqrt{D2.D2}} = \frac{1.74742}{(1.8731)(3.42837)} = 0.00758$$

Pada tabel 3.12 didapatkan hasil perhitungan*cosine distance similarity* sebagai berikut:

Tabel 3. 12 Tabel *Cosine Similarity* iterasi 0.

	D1	D2	D3	D4	D5
D1	1	0.00758	0	0.00173302	0.0129
D2		1	0	0.0013899	0.000735
D3			1	0.00484	0.00517
D4				1	0.004799
D5					1

3.3.8. Membentuk *link cluster* menggunakan *goodness measure*.

Pada pembentukan *link cluster* ini, dimisalkan nilai *threshold* adalah 0.08 dan jumlah *cluster* sebesar 2. Dilakukan perhitungan pembentukan link cluster menggunakan rumus goodness measure pada seluruh link yang terbentuk.

Seperti yang tertulis pada rumus 2.7, sebelum dilakukan perhitungan penentuan link cluster, terlebih dahulu menentukan nilai $f(\Theta)$, yaitu :

$$f(\Theta) = \frac{1-\Theta}{1+\Theta} = \frac{1-0.08}{1+0.08} = \frac{0.92}{1.08} = 0.851$$

Sebagai contoh perhitungan goodness measure dilakukan pada nilai (D1, D2) yang memiliki nilai similarity 0.18857, sehingga didapatkan nilai goodness measure sebagai berikut :

$$g(D1,D_2) = \frac{0.00758}{(1+1)^{1+2\,(0.851)} - (1)^{1+2\,(0.851)} - (1)^{1+2\,(0.851)}} =$$

$$\frac{0.00758}{(2)^{2.702} - (1)^{2.702} - (1)^{2.702}}$$

$$= \frac{0.00758}{4.507} = 0.00168$$

Didapatkan nilai *goodness measure* untuk semua *link*, dapat dilihat pada tabel 3.13 perhitungan *goodness measure* iterasi 0.

Tabel 3. 13 Tabel Perhitungan *goodness measure* iterasi 0

Pair	*Goodness Measure*
D1,D2	0.00168
D1,D3	0
D1,D4	0.000383
D1,D5	0.00285
D2,D3	0
D2,D4	0.000307
D2,D5	0.00016288
D3,D4	1.00107
D3,D5	0.00114515
D4,D5	0.001063

Dasar proses clustering pada algoritma *E-ROCK (Enhanched-RObust Clustering using linKs)* adalah menggunakan nilai goodness measure terbesar pada sebuah link yang ada. Seperti yang telah terlihat pada tabel 3.17 di atas bahwa nilai *goodnessmeasure* terbesar ada pada link (D1, D5) yaitu sebesar 0.00285.Sehingga pembentukan cluster iterasi 0 terdapat pada link (D1, D5).

3.3.9 Looping proses 3.3.7 sampai 3.3.8 sampai dengan jumlah *cluster* yang diinginkan terbentuk.

a) *Cosine Similarity* iterasi 1.

Setelah didapatkan pembentukan cluster iterasi 0 yaitu pada dokumen 1 dan dokumen 2 (D1, D2) dilakukan penggabungan pada indeks dokumen 1 dan dokumen 2 membentuk baris dan kolom baru berdasarkan nilai indeks terbesar antara indeks dokumen 1 dan dokumen 2.Berikut hasil *cosine similarity* iterasi 1 dapat dilihat pada tabel 3.14.

Tabel 3. 14 Tabel *Cosine Similarity* iterasi 1.

	D1,D5	D2	D3	D4
D1,D5	1	0.00758	0	0.00173302
D2		1	0	0.0013899
D3			1	0.00484
D4				1

b) Perhitungan *goodness measure* iterasi 1.

Setelah matriks *similarity* iterasi 1 terbentuk, dilakukan perhitungan *goodness measure* iterasi 1 sebagai berikut:

$$g((D1,D_5),D2) = \frac{0.00758}{(2+1)^{1+2\,(0.851)} - (2)^{1+2\,(0.851)} - (1)^{1+2\,(0.851)}}$$

$$\frac{0.00758}{(3)^{2.702} - (2)^{2.702} - (1)^{2.702}}$$

$$= \frac{0.00758}{11.9547} = 0.000633$$

Berikut hasil perhitungan *goodness measure* iterasi 1 dapat dilihat pada tabel 3.15.

Tabel 3. 15 Tabel PerhitunganGoodness Measureiterasi 1.

Pair	*Goodness Measure*
(D1, D5), D2	0.000633
(D1, D5), D3	0.0004314
(D1, D5), D4	0.0004005
D2, D3	0
D2, D4	0.000307
D3, D4	0.00107

c) *Cosine similarity* iterasi 2.

Setelah melakukan proses pembobotan untuk iterasi kedua didapatkan tabel *similarity* iterasi 2 yang terlihat pada tabel 3.16 di bawah ini,

Tabel 3. 16 Tabel Cosine Similarity iterasi 2.

	D1,D5	D2	D3,D4
D1,D5	1	0.00758	0.00173302
D2		1	0.0013899
D3, D4			1

d) Perhitungan *goodnessmeasure* iterasi 2.

Setelah matriks *similarity* iterasi 2 terbentuk, dilakukan perhitungan *goodness measure* iterasi 2 sebagai berikut :

$$g((D1,D_5),D2) = \frac{0.00758}{(2+2)^{1+2\,(0.851)} - (2)^{1+2\,(0.851)} - (2)^{1+2\,(0.851)}} = \frac{0.00758}{(4)^{2.702} - (2)^{2.702} - (2)^{2.702}}$$

$$= \frac{0.00758}{29.327} = 0.0006332$$

Didapatkan hasil perhitungan *Goodness Measure* iterasi 2 seperti yang terlihat pada tabel 3.17 di bawah ini:

Tabel 3. 17 Tabel Perhitungan Goodness Measure iterasi 2.

Pair	*Goodness Measure*
(D1, D5),D2	0.0006332
(D1, D5), (D3,D4)	0.00017577
D2,(D3,D4)	0.0001599

e) Hasil akhir.

Karena jumlah cluster yang diinginkan sebanyak 2, maka proses iterasi berhenti sampai pada iterasi 2 dengan hasil akhir seperti yang terlihat pada tabel 3.18 berikut:

Tabel 3. 18 Tabel hasil akhir.

	(D1,D5),D2	D3,D4
(D1,D5),D2	~	~
D3,D4		~

Nb: jumlah cluster telah terpenuhi.

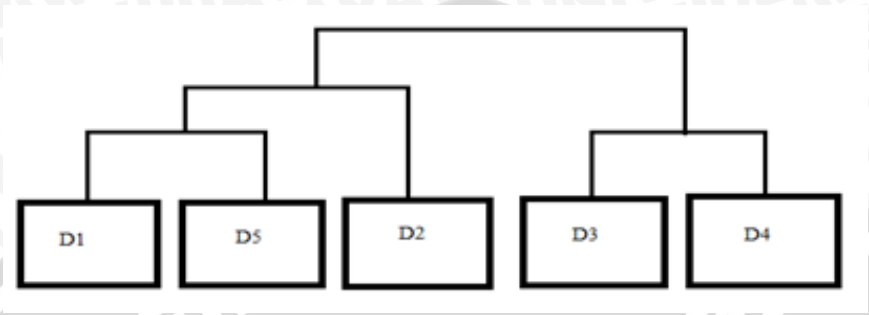
3.3.10 Melakukan pembentukan topik pada tiap-tiap *cluster*

Topik yang didapatkan berdasarkan perhitungan yang telah dilakukan di atas , didapatkan 2 cluster yaitu C((D1,D5),D2) dan C(D3,D4).Untuk *cluster* (((D1,D5),D2) didapatkan frekuensi term terbesar dengan kata '*apriori*' yang digunakan sebagai topik dalam cluster tersebut di atas dengan jumlah frekuensi term sebanyak 3(dapat dilihat pada tabel 3.6).

Sedangkan untuk *cluster* (D3,D4) didapatkan frekuensi term terbesar dengan kata '*fuzzy*' yang digunakan sebagai topik dalam cluster tersebut di atas dengan jumlah frekuensi term sebanyak 3(dapat dilihat pada tabel 3.6).

3.3.11 Evaluasi

Berikut ilustrasi hasil clustering menggunakan Algoritma *E-ROCK (Enhanched – RObust Clustering using linKs)* dapat dilihat pada gambar 3.16.



Gambar 3. 15 Dendogram clustering dokumenmenggunakan algoritma E-ROCK.

Kemudian didapatkan tabel keterangan seperti yang terlihat pada tabel 3.19 di bawah ini:

Tabel 3. 19 Tabel Keterangan.

Dokumen	*Cluster*	
	Fakta	Hasil
1	Apriori	Apriori
2	Apriori	Apriori
3	Fuzzy	Fuzzy
4	Fuzzy	Fuzzy
5	Fuzzy	Apriori

Sehingga didapatkan *error ratio* sesuai dengan rumus 2.8 yang terdapat dalam bab 2 sebagai berikut:

error ratio$= \frac{1}{5} = 0.2 = 20\ \%$

3.4 Perancangan Uji Coba

Pada sub bab ini akan dilakukan perancangan pengujian perangkat pada sistem clustering menggunakan metode *E-ROCK*.Pengujian terhadap perangkat lunak dibagi menjadi dua yaitu pengujian kebenaran perangkat lunak disesuaikan dengan perhitungan manual yang dilakukan sebelumnya dan pengujian algoritma *E-ROCK* yang dimasukkan ke dalam perangkat lunak.

3.4.1 Bahan Pengujian

Bahan yang digunakan adalah data berbentuk dokumen teks yaitu kumpulan artikel jurnal ilmiah ilmu komputer berbahasa Indonesia.

3.4.2 Tujuan Pengujian

Beberapa hal yang menjadi tujuan dari pelaksanaan pengujian terhadap clustering menggunakan algoritma *E-ROCK* ini adalah:

a) Menerapkan program yang telah dibuat dengan menggunakan data yang telah disiapkan / sudah ada sebelumnya.
b) Mengevaluasi pengaruh *threshold* terhadap *error ratio*yang dihasilkan.
c) Mengetahui topik dari masing-masing *cluster* yang terbentuk berdasarkan frekuensi kata yang sering muncul pada masing-masing *cluster* tersebut.

3.4.3 Kriteria Pengujian.

Pengujian yang dilaksanakan meliputi :

a) Mengevaluasi pengaruh *threshold*terhadap *error ratio*yang dihasilkan pada jumlah cluster yang terbentuk. Sebelumnya telah ditentukan jumlah cluster yang diinginkan adalah sebanyak 10 cluster. Sedangkan rentang threshold yang digunakan adalah mulai 0.1 sampai dengan 0.95 dengan tiap-tiap kenaikan sebesar 0,05. Berikut tabel kriteria pengujian pertama dapat dilihat pada tabel 3.20 sebagai berikut :

Tabel 3. 20 Tabel kriteria pengujian tahap pertama

Threshold	Jumlah iterasi	*Error ratio*
i		
.		
.		
.		
k		

b) Mengevaluasi pengaruh *threshold*terhadap *error ratio* dari topikyang dihasilkan. Sebelumnya telah ditentukan jumlah cluster yang diinginkan adalah sebanyak 10 cluster. Sedangkan rentang threshold yang digunakan

adalah mulai 0.1 sampai dengan 0.95 dengan tiap-tiap kenaikan sebesar 0,05. Berikut tabel kriteria pengujian kedua dapat dilihat pada tabel 3.21 sebagai berikut :

Tabel 3. 21 Tabel kriteria pengujian tahap kedua

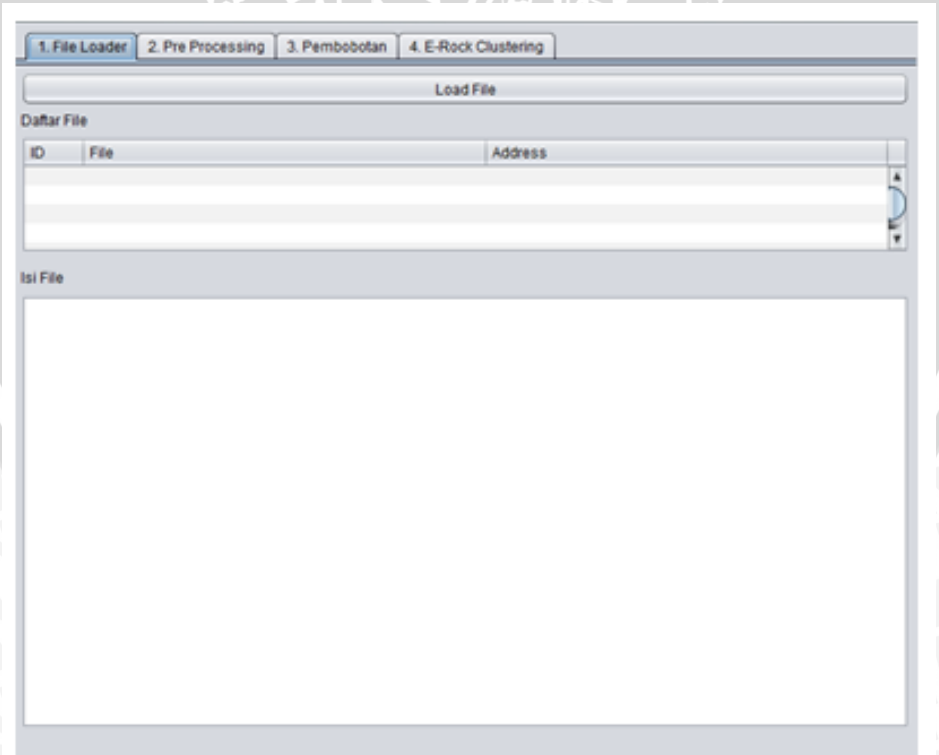
Threshold	Jumlah iterasi	*Error ratio*	Topik yang salah
I			
.			
.			
K			

3.5 Perancangan Interface

Dalam sub bab ini dijelaskan mengenai perancangan antar muka (*interface)* pada sistem pembentukan topik menggunakan algoritma *E-ROCK.* Terdapat 4 *tab,* yaitu:

a) File Loader.

Berikut tampilan *tab file loader* dapat dilihat pada gambar 3.16.

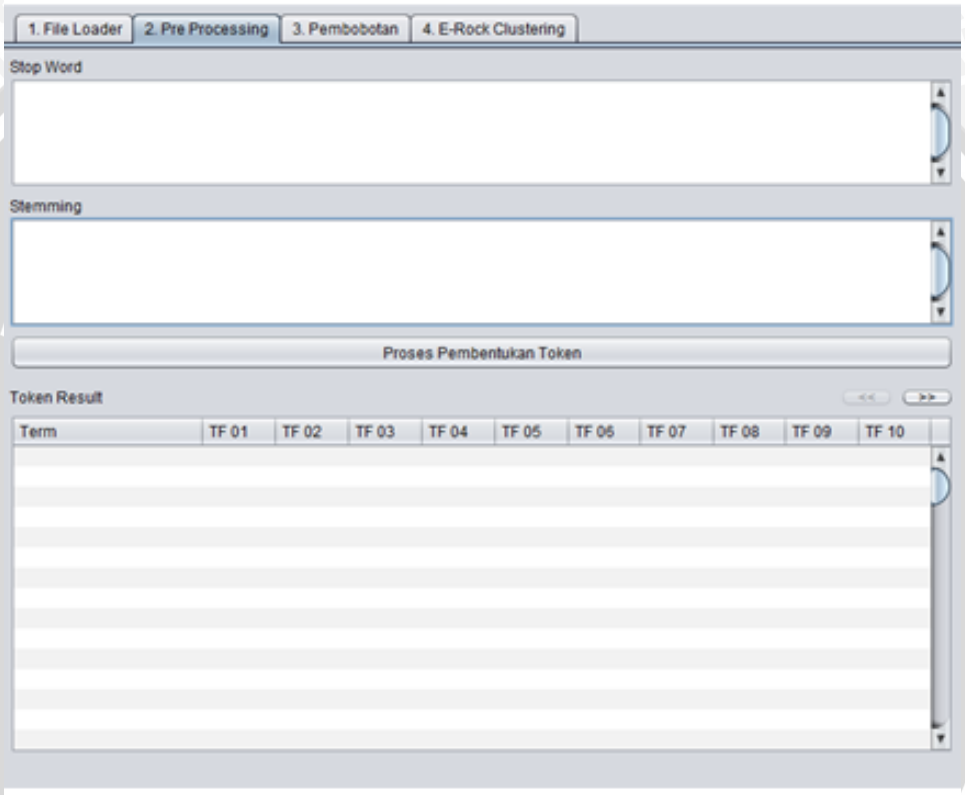


Gambar 3. 16 Tampilan interface tab *file loader.*

Tab ini digunakan untuk membuka dokumen- dokumen yang akan diproses. Terdapat *gridview* yang menampilkan daftar dokumen- dokumen yang akan diproses beserta alamat dokumen tersebut. Terdapat pula *JTextArea* yang berfungsi menampilkan isi dari dokumen.

a) Pre Processing.

Berikut tampilan *tab pre processing* dapat dilihat pada gambar 3.18.

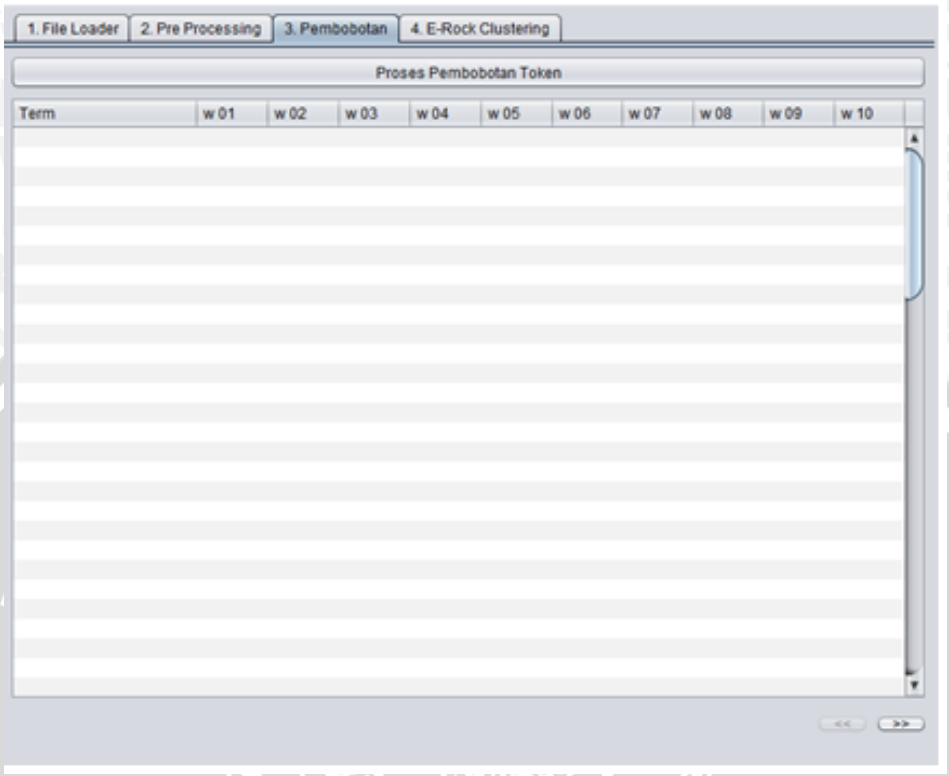


Gambar 3. 17 Tampilan interface tab pre processing.

Tab ini digunakan untuk memproses dokumen yang dinamakan proses *pre processing*. Terdapat *JTextArea* yang berisi daftar stoplist dan kata dasar dari Kamus Besar Bahasa Indonesia (KBBI).Terdapat gridview yang berisi kumpulan ter beserta frekuensinya pada tiap- tiap dokumen.

b) Pembobotan

Berikut tampilan *tab* pembobotan dapat dilihat pada gambar 3.18.

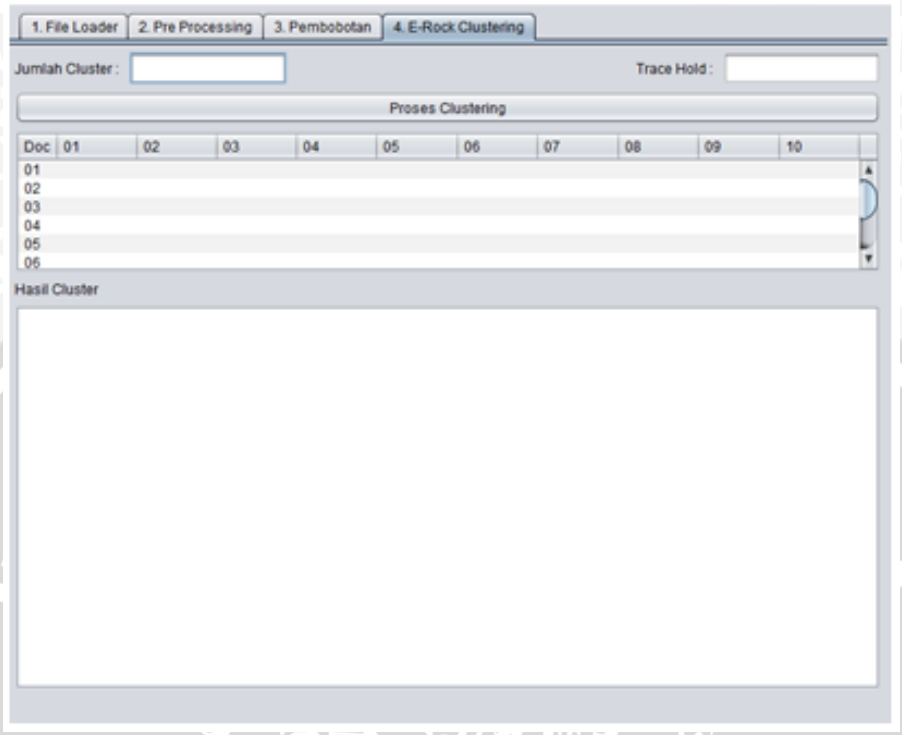


Gambar 3. 18 Tampilan tab pembobotan.

Pada *tab* ini ditampilkan nilai bobot tiap-tiap *term* pada masing- masing dokumen.

c) *E-Rock Clustering*.

Berikut tampilan *tab E-Rock clustering* dapat dilihat pada gambar 3.19.



Gambar 3. 19 Tampilan tab E-Rock Clustering.

Pada *tab* ini terdapat masukan jumlah *cluster* dan *trace hold* yang digunakan untuk pembentukan *cluster* sehingga didapatkan topik pada tiap- tiap *cluster*.

Pada *JTextArea* ditampilkan hasil cluster beserta topic yang dibentuk pada masing- masing cluster. Terdapat pula perhitungan *goodness measure* dan hasil *error ratio* yang dihasilkan.

BAB IV
IMPLEMENTASI DAN PEMBAHASAN

4.1 Lingkungan Implementasi

Lingkungan implementasi yang akan dijelaskan dalam sub-bab ini meliputi lingkungan implementasi perangkat keras dan perangkat lunak.

4.1.1 Lingkungan Implementasi Perangkat Keras

Perangkat keras yang digunakan dalam pembuatan sistem temu kembali informasi berupa teks pada lirik lagu ini meliputi :

1. Processor Intel® Core™ Duo CPU T6500 @ 2.10GHz (2CPUs)
2. Memory 2048MB RAM
3. Harddisk 320 GB

4.1.2 Lingkungan Implementasi Perangkat Lunak

Perangkat lunak yang digunakan dalam pembuatan sistem penentuan lirik lagu dengan sistem temu kembali ini meliputi :

1. Sistem Operasi *Windows XP Home.*
2. *Netbeans 6.8.*
3. *Text editor wordpad.*

4.2 Implementasi Program

Berdasarkan analisis dan perancangan yang dijelaskan pada bab 3, maka pada sub-bab ini akan dijelaskan implementasi dari perancangan proses yang telah dijabarkan pada bab 3. Pada implementasi program terbentuk sebanyak 2 *package* yang terdiri atas *package* untuk menyimpan fungsi-fungsi program dan *package* untuk menyimpan *code user interface* dari program. Kelas – kelas yang dibangun dijelaskan dalam tabel 4.1

Tabel 4. 1 Kelas – kelas yang dibangun

No	Kelas	Keterangan
1	`StopWord` (Implementasi 4.2.2)	Memfilter data latih dengan menghilangkan term - term yang termasuk dalam stoplist.txt. karena penggunaan class tersebut lebih menyesuaikan dengan karakter class bukan pada objectnya maka class ini bersifat static.
2	`Tokenizing` (Implementasi 4.2.3)	Class object data dokumen yang memiliki karakteristik token (term, TF, w, MTF).
3	`Stemming` (Implementasi 4.2.4)	Algoritma stemming menggunakan dasar imbuhan bahasa Indonesia. karena penggunaan class tersebut lebih menyesuaikan dengan karakter class bukan pada objectnya maka class ini bersifat static.
4	`PreProcess-ing` (Implementasi 4.2.5)	Proses awal tokenizing dokumen berupa parsering, case folding, filtering stopwoed dan proses stemming term.
5	`Pembobotan` (Implementasi 4.2.6)	Memproses data Token untuk memperoleh nilai bobot/TF-IDF/w dengan rumus 2.2, 2.3, 2.4. karena penggunaan class tersebut lebih menyesuaikan dengan karakter class bukan pada objectnya maka class ini bersifat static.
6	`CosineSim` (Implementasi 4.2.7)	Menghitung nilai cosine similarity antara dua document sesuai dengan rumus 2.6. karena penggunaan class tersebut lebih menyesuaikan dengan karakter class bukan pada objectnya maka class ini bersifat static.
7	`Cluster` (Implementasi 4.2.8)	Proses pengelompokkan dokumen terhadap cluster yang dibentuk berdasarkan proses pada algoritma E-Rock
8	`E-Rock` (Implementasi 4.2.9)	Proses algorima E-Rock dalam meng-*cluster* document, terdapat proses *Goodness Measure*, *loopingCosSim* dan Looping *Goodness Measure*.
9	`ErrorRasio` (Implementasi 4.2.10)	Proses menghitung *error ratio* hasil pengelompokan system menggunakan Algoritma E-Rock dan membandingkan dengan data asli (data yang telah diberi label).

4.2.1 Implementasi `StopWord`

Pada kelas `StopWord` terdapat beberapa fungsi yang dijelaskan pada tabel 4.2.

Tabel 4. 2 Fungsi – fungsi yang terdapat pada kelas StopWord.

No	Fungsi	Keterangan
a)	`void Setlength (int _length)`	Melakukan *set* jumlah data/term pada *stoplist*. (melakukan*set* jumlah kapasitas *array stoplist*).
b)	`int Getlength ()`	Untuk memperoleh data jumlah *stoplist* pada class StopWord.
c)	`void SetStopWord (int index, String term)`	Melakukan *set term* pada *list stopword* dengan *index array* berdasarkan *index* yang diinputkan.
d)	`String GetStopWord (int index)`	Memperoleh data *term* pada *array stoplist* berdasarkan *no/index* yang diinputkan.
e)	`boolean ItsStopWord (String term)`	Melakukan pengecekan apakan suatu *term* merupakan *term* yang tergolong pada daftar *stoplist*.

a) Melakukan pengecekan apakan suatu *term* merupakan *term* yang tergolong pada daftar *stoplist*.

Pada proses ini dilakukan pengecekan apakan suatu *term* merupakan *term* yang tergolong pada daftar *stoplist* dapat dilihat pada kode program 4.1.

```
public boolean ItsStopWord(String term) {
boolean result = false;
for (int i=1; i<=GetLength(); i++) {
if    (WordList[i]    ==    null    ?    term    !=    null    :
   WordList[i].equalsIgnoreCase(term)) {
   result = true;
   break;
         } }
   return result; }
```

Kode Program 4. 1 Kode program melakukan pengecekan apakan suatu term merupakan term yang tergolong pada daftar stoplist

4.2.2 Implementasi `Tokenizing`

Pada kelas `Tokenizing` terdapat beberapa fungsi yang dijelaskan pada tabel 4.3.

Tabel 4. 3 Fungsi – fungsi yang terdapat pada kelas Tokenizing.

No	Fungsi	Keterangan
a)	`int Index (String term)`	Memperoleh nilai *index* (no urut) suatu *term* pada daftar/*array token*. Apabila term tidak terdapat pada *array token* maka akan menghasilkan nilai -1
b)	`int GetLastIndex()`	Mencari posisi terakhir pada *array token* yang masih kosong (belum memiliki *set* data).
c)	`void SortTerm()`	Untuk mengurutkan data *token* berdasarkan *term*nya (belum terpakai, dapat berguna untuk *invertion search*)
d)	`void SetTerm (int index, String term)`	Melakukan *set term* pada *array token* pada posisi *array* dengan *index* yang berdasarkan *index* yang dimasukkan.
e)	`void SetTerm (String term)`	Melakukan *set term* pada *array token* dengan terlebih dahulu mencari posisi *index* term tersebut pada *arraytoken*.
f)	`String GetTerm (int index)`	Memperoleh nilai *term* pada *array token* dengan posisi *index* sesuai dengan *index* yang dimasukkan.
g)	`void SetFrequensi (int index, int Frequensi)`	Melakukan *setTF (Term Frequent)* pada *array token* dengan terlebih dahulu mencari posisi *indexTF (Term Frequent)* tersebut pada *array token*.
h)	`void SetFrequensi (String term, int Frequensi)`	Melakukan *setTF (Term Frequent)* pada *array token* dengan terlebih dahulu mencari posisi *index* tersebut pada array *token* dengan menggunakan nilai *term* yang diinputkan.
i)	`int GetFrequensi (int index)`	Memperoleh nilai *term* pada *array token* dengan posisi *index* sesuai dengan *index* yang dimasukkan.
j)	`int GetFrequensi (String term)`	Memperoleh nilai *TF (Term Frequent)* pada *array token* dengan terlebih dahulu mencari posisi *index* tersebut pada *array token* dengan menggunakan nilai *term* yang diinputkan.
k)	`void AddFrequensi (int index, int Jumlah)`	Menambahkan *TF (Term Frequent)* pada *array token* pada posisi *index* yang telah diinputkan dengan jumlah yang sesuai dengan jumlah inputan. Misalnya:(token[index].TF + jumlah)
l)	`void AddFrequensi`	Menambahkan senilai satu *TF (Term Frequent)*,

	`(int index)`	pada *TF (Term Frequent)* dengan posisi *index* yang telah diinputkan pada *array token*. Misalnya : (token[index].TF + 1)
m)	`void AddFrequensi (String term, int Jumlah)`	Menambahkan *TF (Term Frequent)* dengan jumlah yang sesuai dengan jumlah inputan pada *array token* dengan terlebih dahulu mencari posisi *index term* tersebut pada *array token.*
n)	`void AddFrequensi (String term)`	Menambahkan senilai satu *TF (Term Frequent)* pada *TF (Term Frequent)array token* dengan terlebih dahulu mencari posisi *index term* tersebut pada *array token.*
o)	`void SetMTF()`	Untuk mencari dan menyimpan data maksimum *term* frekuensi (*MTF*) pada dokumen tersebut.
p)	`int GetMTF()`	Untuk memperoleh data maksimum *term* frekuensi (*MTF)* pd suatu dokumen berdasarkan proses SetMTF() yg telah dilakukan sebelumnya.
q)	`void SetBobot (String term, double bobot)`	Melakukan *set* bobot/*w/tf-idf* senilai dgn bobot yang diinputkan pada *array token* dengan terlebih dahulu mencari posisi *indexterm* tersebut pada *array token.*
r)	`double GetBobot (String term)`	Memperoleh bobot pada *array token* dengan mencari posisi *index* terlebih dahulu berdasarkan *term* yang telah diinputkan.
s)	`void SetLength (int _Length)`	Melakukan *set* jumah *token* / variasi *term*. (melakukan *set* jumlah kapasitas *array token*)
t)	`int GetLength()`	Memperoleh jumlah token yang ada pada *object* dokumen tersebut.
u)	`void SetToken (int index, String term)`	Melakukan *set* nilai *token* (term, dan menambah satu *TF (Term Frequent)* pada *term_frekuensi* nya) pada *array token* dengan terlebih dahulu mencari posisi *index term* tersebut pada array token.
v)	`void SetToken (int index, String term, int Frequensi)`	Melakukan *set* nilai *token* (*term*, dan menambah satu *TF (Term Frequent)* pada *term_frekuensi* nya) pada *array token* dengan *index* yang telah diinputkan.
w)	`void InsertToken (String term)`	Melakukan *settoken* berdasar *term* yang diinputkan, apabila *token* telah ada maka proses hanya akan menambahkan *TF (Term Frequent)* sejumlah satu.
x)	`void InsertToken (String term, int Frequensi)`	Melakukan *set* token berdasar *term* yang diinputkan, apabila token telah ada maka proses hanya akan menambahkan *TF (Term Frequent)* sejumlah frekuensi yang telah dimasukkan.

a) Mengurutkan data *token* berdasarkan *term*nya

Pada proses ini diurutkan data *token* berdasarkan *term*nya dapat dilihat pada kode program 4.2.

```
public void SortTerm() throws Exception { Term TempToken;
for (int i=1; i<=this.GetLength()-1; i++) {
for (int j=i+1; j<=this.GetLength(); j++) {
if (this.Token[i].Term.compareTo(this.Token[j].Term) > 0);
     TempToken    = this.Token[j];
     this.Token[j] =this.Token[i];
     this.Token[i] = TempToken;
              }
         }
    }
```

Kode Program 4. 2 Kode program mengurutkan data token berdasarkan termnya

b) Memperoleh nilai *term* pada *array token* dengan posisi *index* sesuai dengan *index* yang dimasukkan.

Pada proses ini diperoleh nilai *term* pada *array token* dengan posisi *index* sesuai dengan *index* yang dimasukkan dapat dilihat pada kode program 4.3.

```
public String GetTerm(int index) throws Exception {
if ((index <= this.GetLength()) && (index > 0)) {
   return this.Token[index].Term;
  }
else
      { return ""; }}
```

Kode Program 4. 3 Kode program memperoleh nilai term pada array token

c) Melakukan *setTF (Term Frequent)* pada *array token* dengan terlebih dahulu mencari posisi *index* tersebut pada array *token* dengan menggunakan nilai *term* yang diinputkan.

Pada proses dilakukan *setTF (Term Frequent)* pada *array token* dengan terlebih dahulu mencari posisi *index* tersebut pada array *token* dengan menggunakan nilai *term* yang diinputkan dapat dilihat pada kode program 4.4.

```
protected void SetFrequensi(String term, int Frequensi)
throws Exception
{
this.SetFrequensi(this.Index(term), Frequensi);
    }
```

Kode Program 4. 4 Kode program melakukan setTF (Term Frequent) pada array token

d) Memperoleh nilai *TF (Term Frequent)* pada *array token* dengan terlebih dahulu mencari posisi *index* tersebut pada *array token* dengan menggunakan nilai *term* yang diinputkan.

Pada proses ini diperoleh nilai *TF (Term Frequent)* pada *array token* dengan terlebih dahulu mencari posisi *index* tersebut pada *array token* dengan menggunakan nilai *term* yang diinputkan dapat dilihat pada kode program 4.5.

```
public int GetFrequensi(String term) throws Exception {
return this.GetFrequensi(this.Index(term));
```

Kode Program 4. 5 Kode program memperoleh nilai TF (Term Frequent) pada array token

e) Menambahkan *TF (Term Frequent)* pada *array token* pada posisi *index* yang telah diinputkan dengan jumlah yang sesuai dengan jumlah inputan.

Pada proses ini ditambahkan *TF (Term Frequent)* pada *array token* pada posisi *index* yang telah diinputkan dengan jumlah yang sesuai dengan jumlah inputan dapat dilihat pada kode program 4.6.

```
protected void AddFrequensi(int index, int Jumlah) throws
Exception {
if ((index <= this.GetLength()) && (index > 0)) {
    this.Token[index].TF += Jumlah; }
    }
```

Kode Program 4. 6 Kode program menambahkan TF (Term Frequent) pada array token

f) Menambahkan senilai satu *TF (Term Frequent)* pada *TF (Term Frequent)array token.*

Pada proses ini ditambahkan senilai satu *TF (Term Frequent)* pada *TF (Term Frequent)array token* dapat dilihat pada kode program 4.7.

```
protected void AddFrequensi(String term) throws Exception
{
this.AddFrequensi(this.Index(term));}
```

Kode Program 4. 7 Kode program menambahkan senilai satu TF (Term Frequent) pada TF (Term Frequent)array token.

g) Mencari dan menyimpan data maksimum *term* frekuensi (*MTF*) pada dokumen .

Pada proses ini dicari dan menyimpan data maksimum *term* frekuensi (*MTF*) pada dokumen dapat dilihat pada kode program 4.8.

```
public void SetMTF() throws Exception {
        int result = 0;
        for (int i=1; i<=this.GetLength(); i++) {
            if (result < this.GetFrequensi(i)) {
                result = this.GetFrequensi(i);
            }
        }
        this.MTF = result;
    }
```

Kode Program 4. 8 Kode program mencari dan menyimpan data maksimum term frekuensi (MTF).

h) Memperoleh data maksimum *term* frekuensi (*MTF)* pada suatu dokumen.

Pada proses ini diperoleh data maksimum *term* frekuensi (*MTF)* pada suatu dokumen dapat dilihat pada kode program 4.9.

```
public int GetMTF() throws Exception {
return this.MTF;    }
```

Kode Program 4. 9 Kode program memperoleh data maksimum term frekuensi (MTF).

i) Melakukan *set* bobot/*w/tf-idf* senilai dgn bobot yang diinputkan pada *array token* .

Pada proses ini dilakukan *set* bobot/*w/tf-idf* senilai dgn bobot yang diinputkan pada *array token* dapat dilihat pada kode program 4.10.

```
public void SetBobot(String term, double bobot) throws
Exception {
int i = this.Index(term);
if ((i <= this.GetLength()) && (i > 0)) {
            this.Token[i].W = bobot;}}
```

Kode Program 4. 10 Kode program melakukan set bobot/w/tf-idf senilai dgn bobot yang diinputkan pada array token.

j) Memperoleh bobot pada *array token* dengan mencari posisi *index* terlebih dahulu berdasarkan *term* yang telah diinputkan.

Pada proses ini diperoleh bobot pada *array token* dengan mencari posisi *index* terlebih dahulu berdasarkan *term* yang telah diinputkan dapat dilihat pada kode program 4.11.

```
public double GetBobot(String term) throws Exception {
int i = this.Index(term);
if ((i <= this.GetLength()) && (i > 0)) {
      return this.Token[i].W;
        } else { return 0; }
    }
```

Kode Program 4. 11 Kode program memperoleh bobot pada array token

4.2.3 Implementasi `Stemming`

Pada kelas `Stemming` terdapat beberapa fungsi yang dijelaskan pada tabel 4.4.

Tabel 4. 4 Fungsi – fungsi yang terdapat pada kelas Stemming.

No	Fungsi	Keterangan
a)	`void SetLength (int _Length)`	Melakukan *set* jumlah data/*term* pada *wordlist*. (Melakukan *set* jumlah kapasitas *array wordlist*)
b)	`int GetLength()`	Memperoleh jumlah *term* pada *wordlist.*
c)	`void SetStemming (int index, String term)`	Melakukan *setterm* pada *array wordlist* dengan posisi *index* berdasar *index* inputan.
d)	`String GetStemming (int index)`	Memperoleh *term* pada *array wordlist* dengan *index* yang telah di inputkan.
e)	`int Index (String term)`	Memperoleh posisi *index* suatu *term* pada *array wordlist.*
f)	`boolean isVokal (char huruf)`	Melakukan pengecekan apakah karakter inputan merupakan sebuah vokal atau tidak.
g)	`boolean KataDasar (String term)`	Proses pertama pada algoritma *StemmingNazief- Andriani* (2.5) Melakukan pengecekan apakah *term* inputan terdapat pada *array wordlist* (sebuah kata dasar).
h)	`String InflectionSuffixes (String term)`	Proses kedua pada algoritma *StemmingNazief- Andriani* (2.5). Memfilter akhiran –lah, -kah, -ku, -mu, -nya.
i)	`String DerivationSuffixes (String term, boolean status)`	Proses ketiga pada algoritma *StemmingNazief- Andriani* (2.5). Memfilter akhiran –i, -an, -kan.
j)	`Boolean AwalanDgnAkhiranYgAda (String term)`	Melakukan pengecekan apakah kata inputan memiliki imbuhan dengan akhiran yang tidak diijinkan, apabila ya maka proses ke-4 tidak dilakukan.
k)	`String DerivationPreffixes (String asal, String term, String awalan, int count)`	Proses keempat pada algoritma *StemmingNazief- Andriani* (2.5). Memfilter awalan di-, ke-, se-, te-, be-, me-, pe-.
l)	`String Stemming_NaziefAndriani (String term)`	Proses *StemmingNazief- Andriani* (2.5).

a) Melakukan pengecekan apakah karakter inputan merupakan sebuah vokal atau tidak.

Pada proses ini terjadi pengecekan apakah karakter inputan merupakan sebuah vokal atau tidak dapat dilihat pada kode program 4.12.

```
private static boolean isVokal(char huruf) throws Exception
{
return (huruf == 'a' || huruf == 'i' || huruf == 'u' ||
huruf == 'e' || huruf == 'o');
```

Kode Program 4. 12 Kode program Melakukan pengecekan apakah karakter inputan merupakan sebuah vokal.

b) Proses pertama pada algoritma *StemmingNazief- Andriani*.

Pada proses ini terjadi pengecekan apakah *term* inputan terdapat pada *array wordlist* (sebuah kata dasar) dapat dilihat pada kode program 4.13.

```
private static boolean KataDasar(String term) throws
Exception {
if (Stemming.Index(term) > -1) { return true; } else {
return false; }
    }
```

Kode Program 4. 13 Kode program Melakukan pengecekan apakah term inputan terdapat pada array wordlist (sebuah kata dasar).

c) Proses kedua pada algoritma *StemmingNazief- Andriani*.

Pada proses ini terjadi pemfilteran akhiran –lah, -kah, -ku, -mu, -nya. dapat dilihat pada kode program 4.14.

```
private static String InflectionSuffixes(String term)
throws Exception {
        String temp;
        // 2.a -lah, -kah
        if((term.endsWith("lah"))||(term.endsWith("kah")))
{           temp = term.substring(0, term.length() - 3);
            if (Stemming.Index(temp) > -1)
                return Stemming.InflectionSuffixes(temp);
                else return term;
        }
    else
        // 2.b -ku, -mu, -nya
        if((term.endsWith("ku"))||(term.endsWith("mu"))) {
```

```
            temp = term.substring(0, term.length() - 2);
            if (Stemming.Index(temp) > -1)
                return temp; else return term;
        } else if ((term.endsWith("nya"))) {
            temp = term.substring(0, term.length() - 3);
            if (Stemming.Index(temp) > -1)
                return temp; else return term;
        }
    else
        {
          return term; }
}
```

Kode Program 4. 14 Kode program memfilter akhiran –lah, -kah,-ku, -mu, -nya.

d) Proses ketiga pada algoritma *StemmingNazief- Andriani*.

Pada proses ini terjadi pemfilteran akhiran –i, -an, -kan. dapat dilihat pada kode program 4.15.

```
private  static  String  DerivationSuffixes(String  term,
boolean status) throws Exception {
        // 3. -i, -an
        String temp;
        if ((!status) && (term.endsWith("i"))) {
            temp = term.substring(0, term.length() - 1);
            if (Stemming.Index(temp) > -1)
                return temp; else return term;
        } else if ((!status) && (term.endsWith("an"))) {
            temp = term.substring(0, term.length() - 2);
            if (Stemming.Index(temp) > -1)
                return temp;
               else
           return Stemming.DerivationSuffixes(temp, true);
        } else
      //special case -kan
          if ((status) && (term.endsWith("k"))) {
              temp = term.substring(0, term.length() - 1);
            if (Stemming.Index(temp) > -1)
                return temp;
            else return
            Stemming.DerivationSuffixes(term.concat("an"),
            true);
        } else { return term; }
    }
```

Kode Program 4. 15 Kode program memfilter akhiran –i, -an, -kan.

e) Melakukan pengecekan apakah kata inputan memiliki imbuhan dengan akhiran yang tidak diijinkan, apabila ya maka proses keempat tidak dilakukan.

Pada proses ini terjadi pengecekan apakah kata inputan memiliki imbuhan dengan akhiran yang tidak diijinkan, apabila ya maka proses ke-4 tidak dilakukan.Prosesnya dapat dilihat pada kode program 4.16.

```
private static Boolean AwalanDgnAkhiranYgAda(String term)
throws Exception {
        // 4.a cek awalan dgn akhiran yang tidak diijinkan
        boolean P1=term.startsWith("be")
                        && term.endsWith("i");
        //term berakhiran -an bisa berasal dari term
        berakhiran -kan
        boolean P2 = (term.startsWith("di")
                        || term.startsWith("me")
                        || term.startsWith("te"))
                       && (term.endsWith("an")
                       && !term.endsWith("kan"));
        boolean P3 = (term.startsWith("ke")
                        || term.startsWith("se"))
                       && (term.endsWith("i")
                       || term.endsWith("kan"));
  // term.lenght <= 3 gak mungkin ada awalan dan akhiran....
        boolean P4 = (term.length() <= 3);
        return (P1 || P2 || P3 || P4);
    }
```

Kode Program 4. 16 Kode program melakukan pengecekan apakah kata inputan memiliki imbuhan dengan akhiran yang tidak diijinkan.

f) Proses keempat pada algoritma *StemmingNazief- Andriani*.

Pada proses ini terjadi pemfilteran awalan di-, ke-, se-, te-, be-, me-, pe-.Prosesnya dapat dilihat pada kode program 4.17.

```
private static String DerivationPreffixes(String asal,
String term, String awalan, int count) throws Exception {
// 4.a jika awalan memiliki akhiran tertentu (lihat daftar)
maka berhenti
// 4.b jika awalan saat ini sama dengan awalan sebelumnya
maka berhenti
// 4.c tiga awalan telah dihapus maka berhenti
// 4.d.iii jika 2 kata pertama bukan di-/ke-/se-/te-/be-/me-
/pe- maka berhenti
        boolean D4 = !term.startsWith("di")
                        || !term.startsWith("ke")
                        || !term.startsWith("se")
```

```
                    || !term.startsWith("te")
                    || !term.startsWith("be")
                    || !term.startsWith("me")
                    || !term.startsWith("pe");
// 4.f jika tidak ditemukan pada kamus lakukan pengulangan
penghapusan awalan....
                          if
                    ((!Stemming.WordList.contains(term))
                    && !Stemming.AwalanDgnAkhiranYgAda(term)
                    && (!term.startsWith(awalan))
                    && (count <= 3) && (!D4)) {
// 4.d.i awalan di- lanjutannya ke-/se-
// 4.e jika bukan none (dari tipe 4.d.ii) maka bila awalan
di-, ke-, se-, te-, ter-, ter-luluh dihapus
   if (term.startsWith("di")) {
                return
                    Stemming.DerivationPreffixes(asal,
                    term.substring(2), term.substring(0,1),
                     count++); }
            else if (awalan.matches("di")
                    && (term.startsWith("ke")
                    || term.startsWith("se"))) {
                    return
                    Stemming.DerivationPreffixes(asal,
                     term.substring(2), term.substring(0,1),
                    count++); }
// 4.d.ii awalan te-
            else if (term.startsWith("te")) {
                return
                  Stemming.DerivationPreffixes(asal,
                  term.substring(2), term.substring(0,1),
                  count++); }
            else if (awalan.matches("te")
 && (term.startsWith("r"))) {
                return
                  Stemming.DerivationPreffixes(asal,
                  term.substring(1),
                  awalan.concat(term.substring(0,1)),
                  count++); }
            // tipe 4.D none te-r-r
            else if (awalan.matches("ter")
&& term.startsWith("r")) {
                return
                  Stemming.DerivationPreffixes(asal,
                  term.substring(1),
                  awalan.concat(term.substring(0,1)), 3); }
// tipe 4.D none te-not vokal/r-er
            else if (awalan.matches("te")
                  && (!term.startsWith("r")
                  || !Stemming.isVokal(term.charAt(0)))) {
                return
                  Stemming.DerivationPreffixes(asal,
                  term.substring(1),
                  awalan.concat(term.substring(0,1)), 3); }
            else if
                  ((awalan.startsWith("te")
                  &&(!awalan.endsWith("r")
```

```
                    || Stemming.isVokal(term.charAt(2))))
                    && (!term.startsWith("er"))) {
                return
                    Stemming.DerivationPreffixes(asal,
                    term.substring(1),
                    awalan.concat(term.substring(0,1)), 3); }
// 4.g jika sudah dilakukan seleksi dan gagal ditemukan pada
stemming rekam kata sebagai kata dasar...
            else {
                Stemming.WordList.add(asal);
                return asal;
            }
        }
    }
```

Kode Program 4. 17 Kode program memfilter awalan di-, ke-, se-, te-, be-, me-, pe-.

g) Proses algoritma *StemmingNazief- Andriani* .

Pada proses ini terdapat algoritma *Stemming Nazief- Andriani* dapat dilihat pada kode program 4.18.

```
public static String Stemming_NaziefAndriani(String term)
throws Exception {
        if (Stemming.KataDasar(term)) return term;
        else {
            String temp = Stemming.InflectionSuffixes(term);
            temp = Stemming.DerivationSuffixes(temp, false);
            if (!Stemming.AwalanDgnAkhiranYgAda(temp)) {
                temp = Stemming.DerivationPreffixes
                        (temp, temp, "", 0);
            }
            return temp;
        }
    }
```

Kode Program 4. 18 Kode program Proses StemmingNazief- Andriani.

4.2.4 Implementasi PreProcessing

Pada kelas `PreProcessing` terdapat beberapa fungsi yang dijelaskan pada tabel 4.5.

Tabel 4. 5 Fungsi – fungsi yang terdapat pada kelas PreProcessing.

No	Fungsi	Keterangan
a)	`String CaseFolding (String _doc)`	Memfilter document inputan dengan merubah ukuran karakter (*to lowercase*), dan menghilangkan beberapa karakter tertentu dengan merubahnya menjadi spasi.
b)	`PreProcessing (String Document, int DokKe)`	Proses awal pembacaan suatu document dengan melakukan parsing, tokenizing, filtering stopword, dan stemming kata.

a) *Case Folding.*

Pada proses ini terjadi pemfilteran document inputan dengan merubah ukuran karakter (*to lowercase*), dan menghilangkan beberapa karakter tertentu dengan merubahnya menjadi spasi. Prosesnya dapat dilihat pada kode program 4.19.

```
private static String CaseFolding(String _doc)
throws Exception {
        String result;
        //1. merubah string menjadi lowercase.
        result = _doc.toLowerCase();
        //2. mengganti "." "," dan "-" menjadi spasi
        result = result.replace(".", " ");
        result = result.replace(",", " ");
        result = result.replace("-", " ");
        //3. Hapus semua special karakter
        result = result.replace("\\|", "");
        result = result.replace("\\n", "");
        result = result.replace("\"", "");
        result = result.replace("\'", "");
        result = result.replace("\\", "");
        //4. Hapus semua tanda baca
        result = result.replace("<", "");
        result = result.replace(">", "");
        result = result.replace("{", "");
        result = result.replace("}", "");
        result = result.replace("[", "");
        result = result.replace("]", "");
        result = result.replace("(", "");
        result = result.replace(")", "");
        result = result.replace("!", "");
        result = result.replace("#", "");
```

```
        result = result.replace("^", "");
        result = result.replace(";", "");
        //5. Hapus semua simbol
        result = result.replace("&", "");
        result = result.replace("*", "");
        result = result.replace("/", "");
        result = result.replace("+", "");
        result = result.replace("-", "");
        result = result.replace("=", "");
        result = result.replace("?", "");
        result = result.replace(":", "");
        result = result.replace("@", "");
        result = result.replace("%", "");

        return result;
    }
```

Kode Program 4. 19 Kode program memfilter document inputan.

b) *Preprocessing*

Proses ini merupakan proses awal yaitu pembacaan suatu dokumen dengan melakukan parsing, tokenizing, filtering stopword, dan stemming kata. Prosesnya dapat dilihat pada kode program 4.20.

```
public static void PreProcessing(String Document, int DokKe)
throws Exception {
        //1. Case Folding
        Document = PreProcessing.CaseFolding(Document);
        //2. Beggin Parsing
        String term[] = Document.split(" ");
        Global.Doc.add(new Tokenizing());
        if (Global.Doc.size() <= 1) {
            Global.Doc.add(new Tokenizing()); }
        Global.Doc.get(DokKe).SetLength(term.length);
        //3. Beggin Filtering
        String tempterm;
        for (int i=0; i<term.length; i++) {
            //3.1 Removing StopWord
            if (!StopWord.ItsStopWord(term[i])) {
                //3.2 Stemming KataTempter
                 Stemming.Stemming_NaziefAndriani(term[i]);
                if (!tempterm.trim().isEmpty()) {
                Global.Doc.get(DokKe).InsertToken(tempterm);
                    if (!Global.DT.contains(tempterm)) {
                        if (Global.DT.size() <= 1) {
                  Global.DT.add(new String()); }
                        Global.DT.add(tempterm);
                    }}}} }
```

Kode Program 4. 20 Kode program proses awal pembacaan suatu dokumen.

4.2.5 Implementasi `Pembobotan`.

Pada kelas `Pembobotan` terdapat beberapa fungsi yang dijelaskan pada tabel 4.6.

Tabel 4. 6 Fungsi – fungsi yang terdapat pada kelas Pembobotan.

No	Fungsi	Keterangan
a)	`int GetDFi(String term)`	Untuk memperoleh nilai *Document Frequence (DFi)* pada kumpulan dokumen lihat keterangan rumus 2.3
b)	`double GetNormalisasiTF (int DocKe, String term)`	Untuk memperoleh nilai *Term Frequent* (*TF)* ternormalisasi dari suatu dokumen.
c)	`double GetIDF (String term)`	Untuk memperoleh nilai *Inverse Document Frequency (IDF)* dari suatu dokumen lihat rumus 2.3
d)	`void BegginPembobotan( )`	Proses untuk memperoleh nilai bobot suatu term dari suatu dokumen terhadap kumpulan dokumen yang ada. Lihat rumus 2.2 atau 2.4
e)	`int GetCDFi (String term)`	Untuk memperoleh nilai *Document Frequence (DFi)* pada kumpulan *cluster*. Lihat keterangan rumus 2.3
f)	`double GetNormalisasiCTF (Cluster C, String term)`	Untuk memperoleh nilai *Term Frequent* (*TF)* ternormalisasi dari suatu *cluster*.
g)	`double GetCIDF(String term)`	Untuk memperoleh nilai *Inverse Document Frequency (IDF)* dari suatu Cluster lihat rumus 2.3
h)	`double GetPembobotanCluster(Cluster C, String term)`	Proses untuk memperoleh nilai bobot suatu *term* dari suatu *cluster* terhadap kumpulan *cluster* yang ada. Lihat rumus 2.2 atau 2.4

a) Memperoleh nilai *Document Frequence (DFi).*

Pada proses ini terjadi perolehan nilai *Document Frequence (DFi)* pada kumpulan dokumen dapat dilihat pada kode program 4.21.

```
private static int GetDFi(String term) throws Exception {
        int result = 0;
        for (int i=1; i<=Global.TotalDoc; i++) {
            if (Global.Doc.get(i).Index(term) > -1) {
          result += 1; }}
        return result;}
```

Kode Program 4. 21 Kode program nilai Document Frequence (DFi).

b) Memperoleh nilai *Term Frequent* (*TF)* ternormalisasi

Untuk memperoleh nilai *Term Frequent* (*TF)* ternormalisasi pada dokumen dapat dilihat pada kode program 4.22.

```
private static double GetNormalisasiTF(int DocKe, String
term) throws Exception {
return
(Global.Doc.get(DocKe).GetFrequensi(term) * 1.0) /
Global.Doc.get(DocKe).GetMTF();}
```

Kode Program 4. 22 Kode program memperoleh nilai Term Frequent (TF) ternormalisasi.

c) Memperoleh nilai *Inverse Document Frequency (IDF)*

Pada proses ini terjadi perolehan nilai *Inverse Document Frequency (IDF)* pada dokumen dapat dilihat pada kode program 4.23.

```
// log basis 2 = (log(x) / log(2))
private static double GetIDF(String term) throws Exception {
int DFi = Pembobotan.GetDFi(term);
if (DFi > 0) {
return Math.log((Global.TotalDoc * 1.0) / DFi) /
Math.log(2);
        } else { return 0; }
    }
```

Kode Program 4. 23 Kode program memperoleh nilai Inverse Document Frequency (IDF).

d) Memperoleh nilai bobot suatu term dari suatu dokumen terhadap kumpulan dokumen.

Pada proses ini terjadi perolehan nilai bobot suatu term dari suatu dokumen terhadap kumpulan dokumen prosesnya dapat dilihat pada kode program 4.24.

```
public static void BegginPembobotan() throws Exception {
        String term;
        double IDF;
        double NormalisasiTF;
        //virtual matriks term.
        for (int j=1; j<=Global.TotalDoc; j++) {
            //set maks frekuensi tiap dokumen.
            Global.Doc.get(j).SetMTF();
```

```
            for (int i=1; i<Global.DT.size(); i++){
                term = Global.DT.get(i);
                //set bobot term tiap dokumen.
                IDF = Pembobotan.GetIDF(term);
                if (IDF > 0) {
                   {
                    Global.Doc.get(j).SetBobot(term,
((Pembobotan.GetNormalisasiTF(j, term) * 1.0) * IDF));
                }
            else {
                 Global.Doc.get(j).SetBobot(term,   0); }
                }
               }
             }
```

Kode Program 4. 24 Kode program memperoleh nilai bobot suatu term dari suatu dokumen.

e) Memperoleh nilai *Document Frequence (DFi)* pada kumpulan *cluster*.

Pada proses ini terjadi perolehan nilai *Document Frequence (DFi)* pada kumpulan *cluster* prosesnya dapat dilihat pada kode program 4.25.

```
private static int GetCDFi(String term) throws Exception {
        int result = 0;
        for (int i=0; i<Global.Clusters.size(); i++) {
             int j=0;
            boolean stopku = false;
            while
           ((j < Global.Clusters.get(i).GetLength())
            && !stopku) {
           if
            (Global.Doc.get(Global.Clusters.get(i).
            GetElement(j)).Index(term) > -1) {
                 result += 1;
                  stopku = true; }
               j++;} }return result;
}
```

Kode Program 4. 25 Kode program memperoleh nilai Document Frequence (DFi) pada kumpulan cluster.

f) Memperoleh nilai *Term Frequent* (*TF)* ternormalisasi dari suatu *cluster*.

Pada proses ini terjadi perolehan nilai *Term Frequent* (*TF)* ternormalisasi dari suatu *cluster* prosesnya dapat dilihat pada kode program 4.26.

```
private static double GetNormalisasiCTF(Cluster C, String
term) throws Exception {
return (C.GetTF(term) * 1.0) / C.GetMTF();
    }
```

Kode Program 4. 26 Kode program untuk memperoleh nilai Term Frequent (TF) ternormalisasi dari suatu cluster.

g) Memperoleh nilai *Inverse Document Frequency (IDF)* dari suatu *cluster*.

Pada proses ini terjadi perolehan nilai *Inverse Document Frequency (IDF)* dari suatu *cluster* prosesnya dapat dilihat pada kode program 4.27.

```
private static double GetCIDF(String term) throws Exception
{
int DFi = Pembobotan.GetCDFi(term);
if (DFi > 0)
{
return Math.log((Global.Clusters.size() * 1.0) / DFi) /
Math.log(2);}
 else { return 0; }}
```

Kode Program 4. 27 Kode program memperoleh nilai Inverse Document Frequency (IDF) dari suatu cluster.

h) Memperoleh nilai bobot suatu *term* dari suatu *cluster* terhadap kumpulan *cluster* yang ada.

Pada proses ini terjadi perolehan nilai bobot suatu *term* dari suatu *cluster* terhadap kumpulan *cluster* yang ada prosesnya dapat dilihat pada kode program 4.28.

```
public static double GetPembobotanCluster(Cluster C, String
term) throws Exception {
return (Pembobotan.GetNormalisasiCTF(C, term) * 1.0) *
Pembobotan.GetCIDF(term);
    }
```

Kode Program 4. 28 Kode program memperoleh nilai bobot suatu term dari suatu cluster terhadap kumpulan cluster.

4.2.6 Implementasi `CosineSim`.

Pada kelas `CosineSim` terdapat beberapa fungsi yang dijelaskan pada tabel 4.7.

Tabel 4. 7 Fungsi – fungsi yang terdapat pada kelas CosineSim.

No	Fungsi	Keterangan
a)	`Void SetDataCS(double data, int x, int y)`	Melakukan *set* data pada *array matrix* data*CosSim* pada *index* (x,y) dengan data inputan.
b)	`Double GetDataCS(int x, int y)`	Memperoleh data *CosSim* pada *array matrix*.
c)	`Void HapusDataCS(int y)`	Menghapus / *trim arraylist*pada *index* ke-y.
d)	`Void HapusDataCS(int x, int y)`	Menghapus / *trim arraylist* pada *index* x dengan terlebih dahulu *mentrimarray* pada *index* y.
e)	`Void resizeDataCS()`	*Mentrim* ukuran *arraylist* sesuai dengan data yang telah dihapus.
f)	`double GetDotProductDD (int doc1, int doc2)`	Menghitung *dot product* dokumen ke satu dengan dokumen ke 2
g)	`double TotalPowerBobot DD(int docke)`	Menghitung total kuadrat bobot dari dokumen inputan.
h)	`double GetVectorDD(int doc1, int doc2)`	Menghitung jarak *cross product* antara dua dokumen inputan.
i)	`double GetCosineSimila rityDD(int doc1, int doc2)`	Menghitung *CosSim* antara dua dokumen berdasar rumus 2.6

a) Melakukan *set* data pada *array matrix* data*CosSim* pada *index* (x,y) dengan data inputan.

Pada proses ini dilakukan *set* data pada *array matrix* data*CosSim* pada *index* (x,y) dengan data inputan prosesnya dapat dilihat pada kode program 4.29.

```
public static void SetDataCS(double data, int x, int y)
throws Exception {
        if (CosineSim.CSim.isEmpty()) {
            for (int i=0; i<Global.TotalDoc;
```

```
i++) {
                    CosineSim.CSim.add(i, new CosSim());
                    for (int j=0; j<Global.TotalDoc; j++) {
                        CosineSim.CSim.get(i).dataCS.add(j, new
Double(0.0));
                    }

}
            }
            CosineSim.CSim.get(x).dataCS.set(y, data);
        }
```

Kode Program 4. 29 Kode program melakukan set data pada array matrix dataCosSim.

b) Memperoleh data *CosSim* pada *array matrix*.

Pada proses ini terjadi perolehan data *CosSim* pada *array matrix* dapat dilihat pada kode program 4.30.

```
public static double GetDataCS(int x, int y) throws
Exception {
return CosineSim.CSim.get(x).dataCS.get(y).doubleValue();
    }
```

Kode Program 4. 30 Kode program memperoleh dataCosSim pada array matrix.

c) Menghapus / *trim arraylist* pada *index* ke-y.

Pada proses ini terjadi penghapusan / *trim arraylist* pada *index* ke-y dapat dilihat pada kode program 4.31.

```
public static void HapusDataCS(int y) throws Exception {
        CosineSim.CSim.remove(y);
CosineSim.CSim.trimToSize();
    }
```

Kode Program 4. 31 Kode program menghapus / trim arraylist pada index ke-y

d) Menghapus / *trim arraylist* pada *index* x.

Pada proses ini terjadi penghapusan / *trim arraylist* pada *index* x dapat dilihat pada kode program 4.32.

```
public static void HapusDataCS(int x, int y) throws
Exception {
        CosineSim.CSim.get(x).dataCS.remove(y);
CosineSim.ReSizeDataCS();
    }
```

Kode Program 4. 32 Kode program menghapus / trim arraylist pada

e) Menghitung *dot product* dokumen ke satu dengan dokumen ke 2.

Pada proses ini terjadi perhitungan *dot product* dokumen ke satu dengan dokumen kedua dapat dilihat pada kode program 4.33.

```
    //Dot Product = Total V1.V2 (bobot tiap term antara dua
dokumen terpilih dikalikan)
protected static double GetDotProductDD(int doc1, int doc2)
throws Exception {
double result = 0;
for (int i=1; i<Global.DT.size(); i++){
result += (Global.Doc.get(doc1).GetBobot(Global.DT.get(i)) *
Global.Doc.get(doc2).GetBobot(Global.DT.get(i))
* 1.0);
        }
return result;

    }
```

Kode Program 4. 33 Kode program menghitung dot product dokumen ke satu dengan dokumen kedua

f) Menghitung total kuadrat bobot dari dokumen inputan.

Pada proses ini terjadi perhitungan total kuadrat bobot dari dokumen inputan dapat dilihat pada kode program 4.34.

```
//Dot Product = Total C1.C2 (bobot tiap term antara dua
cluster terpilih dikalikan)
    protected static double GetDotProductCC(int c1, int c2)
throws Exception {
        double result = 0;
        for (int i=1; i<Global.DT.size(); i++){
            result                                          +=
(Global.Clusters.get(c1).GetBobot(Global.DT.get(i))          *
```

```
Global.Clusters.get(c2).GetBobot(Global.DT.get(i)) * 1.0);
        } return result;
    }
```

Kode Program 4. 34 Kode program menghitung total kuadrat bobot dari dokumen inputan

g) Menghitung total kuadrat bobot dari dokumen inputan.

Pada proses ini terjadi perhitungan total kuadrat bobot dari dokumen inputandapat dilihat pada kode program 4.35.

```
//Total Power Bobot = Bobot tiap term pada dokumen terpilih
di kuadratkan kemudian ditotal.

private static double TotalPowerBobotD(int docke) throws
Exception {
double result = 0;
for (int i=1; i<Global.DT.size(); i++){
result                                                    +=
Math.pow(Global.Doc.get(docke).GetBobot(Global.DT.get(i)),
2);
        } return result * 1.0;
    }
```

Kode Program 4. 35 Kode program menghitung jarak cross product antara dua dokumen inputan

h) Menghitung jarak *cross product* antara dua dokumen inputan.

Pada proses ini terjadi perhitungan jarak *cross product* antara dua dokumen inputan dapat dilihat pada kode program 4.36.

```
protected static double GetVectorDD(int doc1, int doc2)
     throws Exception {
 return Math.sqrt(CosineSim.TotalPowerBobotD(doc1) * 1.0)
* Math.sqrt(CosineSim.TotalPowerBobotD(doc2) * 1.0) }
```

Kode Program 4. 36 Kode program menghitung jarak cross product antara dua dokumen inputan.

i) Menghitung *CosSim* antara dua dokumen berdasar rumus 2.6.

Pada proses ini terjadi perhitungan *CosSim* antara dua dokumen berdasar rumus 2.6 dapat dilihat pada kode program 4.37.

```
//CosineSim(V1,V2) = DotProduct(V1,V2) / Vektor(V1,V2);
public static double GetCosineSimilarityDD(int doc1, int
doc2) throws Exception {
if (doc1 == doc2) { return 1; }
double tempVector = (CosineSim.GetVectorDD(doc1, doc2) *
1.0);
if (tempVector != 0) {
return (CosineSim.GetDotProductDD(doc1, doc2) * 0.1) /
tempVector;
        } else return 0;
    }
```

Kode Program 4. 37 Kode program Menghitung CosSim antara dua dokumen.

4.2.7 Implementasi `Cluster`.

Pada kelas `Cluster` terdapat beberapa fungsi yang dijelaskan pada tabel 4.8.

Tabel 4. 8 Fungsi – fungsi yang terdapat pada kelas Cluster.

No	Fungsi	Keterangan
a)	`void SetElement int dokke)`	Melakukan *set* id dokumen yang termasuk dalam *cluster* ini.
b)	`int GetElement (int index)`	Memperoleh id dokumen pada *cluster* tersebut pada *index* yang diinginkan
c)	`int GetLength()`	Memperoleh jumlah dokumen yang terdapat pada *cluster* tersebut.
d)	`boolean isElementInThis Cluster(int data)`	Melakukan pengecekan apakah dokumen inputan telah terdapat pada *cluster* tersebut.
e)	`int GetTF(String term)`	Proses memperoleh *Term Frequent (TF)* suatu *term* pada *cluster* tersebut
f)	`double GetBobot (String term)`	Proses memperoleh bobot suatu *term* pada *cluster*

		tersebut
g)	`Void SetMTF()`	Proses melakukan *set* data maksimum *Term Frequent (TF)* pada *cluster* yang dibentuk dan mengambil term dgn *Term Frequent (TF)* terbesar sebagai topik.
h)	`int GetMTF()`	Proses memperoleh *Max Term Frequence (MTF)*pada *cluster* tersebut.
i)	`String GetTopik()`	Proses memperoleh topik pada *cluster* tersebut

a) Melakukan *set* id dokumen yang termasuk dalam *cluster* ini.

Pada proses ini dilakukan *set* id dokumen yang termasuk dalam *cluster* ini dapat dilihat pada kode program 4.38.

```
public void SetElement(int dokke) throws Exception {
      this.Element.add(dokke);
    }
```

Kode Program 4. 38 Kode program melakukan set id dokumen yang termasuk dalam cluster

b) Memperoleh id dokumen pada *cluster* tersebut pada *index* yang diinginkan.

Pada proses ini diperoleh id dokumen pada *cluster* tersebut pada *index* yang diinginkan dapat dilihat pada kode program 4.39.

```
public int GetElement(int index) throws Exception {
      return this.Element.get(index);
    }
```

Kode Program 4. 39 Kode program memperoleh id dokumen pada cluster.

c) Memperoleh jumlah dokumen yang terdapat pada *cluster* tersebut.

Pada proses ini diperoleh jumlah dokumen yang terdapat pada *cluster* tersebut dapat dilihat pada kode program 4.40.

```
public int GetLength() throws Exception {
return this.Element.size();
```

Kode Program 4. 40 Kode program memperoleh jumlah dokumen yang terdapat pada cluster.

d) Melakukan pengecekan apakah dokumen inputan telah terdapat pada *cluster* tersebut.

Pada proses ini dilakukan pengecekan apakah dokumen inputan telah terdapat pada *cluster* tersebut dapat dilihat pada kode program 4.41.

```
public boolean isElementInThisCluster(int data) throws
Exception {
        boolean result = false;
        for (int i=0; i<this.GetLength(); i++) {
            if (this.GetElement(i) == data) {
                result = true;
                break; } }
        return result;
    }
```

Kode Program 4. 41 Kode program melakukan pengecekan apakah dokumen inputan telah terdapat pada cluster.

e) Proses memperoleh *Term Frequent (TF)* suatu *term* pada *cluster* tersebut.

Proses memperoleh *Term Frequent (TF)* suatu *term* pada *cluster* tersebut dapat dilihat pada kode program 4.42.

```
public int GetTF(String term) throws Exception {
int result = 0;
for (int i=0; i<this.GetLength(); i++) {
     result +=Global.Doc.get(this.GetElement(i)).
     GetFrequensi(term); }
return result; }
```

Kode Program 4. 42 Kode program memperoleh Term Frequent (TF) suatu term pada cluster.

f) Proses memperoleh bobot suatu *term* pada *cluster* tersebut.

Proses memperoleh bobot suatu *term* pada *cluster* tersebut dapat dilihat pada kode program 4.43.

```
    public double GetBobot(String term) {
        double result = 0;
        for (int i=0; i<this.GetLength(); i++) {
            result                                              +=
Global.Doc.get(this.GetElement(i)).GetBobot(term);
        }
        return (result * 1.0)/ this.GetLength();
    }
public double GetBobot(String term) throws Exception {
        return Pembobotan.GetPembobotanCluster(this, term);
    }
```

Kode Program 4. 43 Kode program memperoleh bobot suatu term pada cluster.

g) Proses melakukan *set* data maksimum *Term Frequent (TF)* pada *cluster* yang dibentuk dan mengambil term dengan *Term Frequent (TF)* terbesar sebagai topik.

Proses melakukan *set* data maksimum *Term Frequent (TF)* pada *cluster* yang dibentuk dan mengambil term dengan *Term Frequent (TF)* terbesar sebagai topik dapat dilihat pada kode program 4.44.

```
public void SetMTF() throws Exception {
        for (int i=1; i<Global.DT.size(); i++) {
            int temp = 0;
            for (int j=0; j<this.GetLength(); j++) {
                temp                                            +=
Global.Doc.get(this.GetElement(j)).GetFrequensi(Global.DT.ge
t(i));
            }

            if (this.MTF.MTF <= temp) {
                if (this.MTF.MTF == temp) {
                    this.MTF.Term.concat(Global.DT.get(i)   +
", ");
                } else { this.MTF.Term = Global.DT.get(i); }
                this.MTF.MTF = temp;
            }}}
```

Kode Program 4. 44 Kode program melakukan set data maksimum Term Frequent (TF) pada cluster.

h) Proses memperoleh topik pada *cluster* tersebut.

Proses memperoleh topik pada *cluster* tersebut dapat dilihat pada kode program 4.45.

```
public String GetTopik() throws Exception {
            return this.MTF.Term;     }
```

Kode Program 4. 45 . Kode program memperoleh topik pada cluster

4.2.8 Implementasi `ERock`

Pada kelas `ERock` terdapat beberapa fungsi yang dijelaskan pada tabel 4.9.

Tabel 4. 9 Fungsi – fungsi yang terdapat pada kelas ERock.

No	Fungsi	Keterangan
a)	`Void setclusterCC(int c1, int c2)`	Proses untuk melebur suatu *cluster* (c1) dengan *cluster* lainnya (c2).
b)	`Double GoodNessMeasure( double CS, int ni,int nj)`	Proses menghitung nilai *goodness measure* sesuai dengan rumus 2.7.
c)	`Void loopGoodNessMeasure()`	Proses looping perhitungan *CosSim* dan *Goodness Measure* untuk diambil *cluster* dengan hubungan *Goodness Measure* terbesar dan disusun sebagai *cluster* baru.
d)	`Void ERock`	Menjalankan Algoritma *E-Rock (looping Goodness Measure).*

a) Melebur suatu *cluster* (c1) dengan *cluster* lainnya (c2).

Melebur suatu *cluster* (c1) dengan *cluster* lainnya (c2) prosenya dapat dilihat pada kode program 4.46.

```
private static void SetClusterCC(int c1, int c2) throws
Exception {
        if (Global.Clusters.size() < c1) {
            Global.Clusters.add(c1, new Cluster());
        }
        int E;
        for (int i=0;
            i<Global.Clusters.get(c2).GetLength(); i++) {
            E = Global.Clusters.get(c2).GetElement(i);
       If
       (!Global.Clusters.get(c1).
       isElementInThisCluster(E)) {
       Global.Clusters.get(c1).SetElement(E);
```

```
            }
        }
        Global.Clusters.remove(c2);
        Global.Clusters.trimToSize();
        Global.Clusters.get(c1).SetMTF();
    }
```

Kode Program 4. 46 Kode program melebur suatu cluster (c1) dengan cluster lainnya (c2).

b) Menghitung nilai *goodness measure* sesuai dengan rumus 2.7.

Proses menghitung nilai *goodness measure* sesuai dengan rumus 2.7.Prosesnya dapat dilihat pada kode program 4.47.

```
protected static double GoodnessMeasure(double CS, int ni,
      int nj) throws Exception {
      double NilaiBwh = (Math.pow(ni+nj, Global.fPower()))
                        - (Math.pow(ni, Global.fPower()))
                        -(Math.pow(nj, Global.fPower()));
      if (NilaiBwh > 0) {
          return (CS / NilaiBwh);
      } else return 0;}
```

Kode Program 4. 47 Kode program Menghitung nilai goodness measure.

c) *Looping Goodness Measure* untuk diambil *cluster* dengan hubungan *Goodness Measure* terbesar dan disusun sebagai *cluster* baru.

Proses melakukan *looping* perhitungan *Goodness Measure* untuk diambil *cluster* dengan hubungan *Goodness Measure* terbesar dan disusun sebagai *cluster* baru. Prosesnya dapat dilihat pada kode program 4.48.

```
    int i = 0;
          // set dok to cluster type method
          for (int j=0; j<Global.TotalDoc; j++) {
              Global.Clusters.add(j, new Cluster());
              Global.Clusters.get(j).SetElement(j+1);
              Global.Clusters.get(j).SetMTF();
    }
    while (Global.Clusters.size() > Global.jumCluster) {
          ArrayList<TGM> TempDT = new ArrayList<TGM>();
          double MaxGM = 0.0;
        int indexMax = 0;
    MainForm.txtCluster.append("Loop GoodNess Measure Ke "
    + Integer.toString(i) + "\n");
        for (int x=0; x<Global.Clusters.size()-1; x++) {
        for (int y=x+1;y<Global.Clusters.size(); y++) {
```

```
                        TempDT.add(new TGM());
                    TempDT.get(TempDT.size() -1).Ni = x;
                    TempDT.get(TempDT.size() -1).Nj = y;
 // hitung GM
                    TempDT.get(TempDT.size() -1).value
          = ERock.GoodnessMeasure(ERock.LoopCosineSim(x, y),

      Global.Clusters.get(x).GetLength(),

      Global.Clusters.get(y).GetLength());
// set / cari cluster dengan maksimum goodness measure
  if (MaxGM < TempDT.get(TempDT.size() -1).value) {
      MaxGM = TempDT.get(TempDT.size() -1).value;
      indexMax = TempDT.size() -1;
       }
// tamplian kesimpulan GM
      String tempXY = "";
      Global.LoopGM.add(i, new DrawPoint());
      for  (int  m=0;  m<Global.Clusters.get(x).GetLength();
      m++) {tempXY +=
      (Integer.toString(Global.Clusters.get(x).GetElement(m)
      ) + ",");

      Global.LoopGM.get(i).elementX.add(Global.Clusters.get(
      x).GetElement(m));
                        }
       tempXY += " n ";
        for  (int  m=0;  m<Global.Clusters.get(y).GetLength();
        m++) {
        tempXY +=
        (Integer.toString(Global.Clusters.get(y).
         GetElement(m)) + ",");

      Global.LoopGM.get(i).elementY.add(Global.Clusters.get(
      y).GetElement(m));}
      MainForm.txtCluster.append("("  +  tempXY.substring(0,
      tempXY.length()    -1)    +    ")    =    "    +
      Double.toString(TempDT.get(TempDT.size()  -1).value)  +
      "\n");
                    }
                   }
      // set cluster baru sesuai GM

      MainForm.txtCluster.append("========================
      ============ \n");
      ERock.SetClusterCC(TempDT.get(indexMax).Ni,
      TempDT.get(indexMax).Nj);
                  i++;
               }
    }
```

Kode Program 4. 48 Kode program *looping* perhitungan *Goodness Measure*.

d) Menjalankan Algoritma *E-ROCK.*

Proses menjalankan Algoritma *E-ROCK (looping Goodness Measure)* .Prosesnya dapat dilihat pada kode program 4.49.

```
public static void ProsesERock()
      throws Exception {
      ERock.LoopGoddnessMeasure();
    }
```

Kode Program 4. 49 Kode program menjalankan AlgoritmaE-Rock.

4.2.9 Implementasi `ErrorRasio`

Pada kelas `ErrorRasio` terdapat beberapa fungsi yang dijelaskan pada tabel 4.10.

Tabel 4. 10 Fungsi – fungsi yang terdapat pada kelas ErrorRasio.

No	Fungsi	Keterangan
a)	`Private static boolean isInClass (String Add)`	Melakukan pengecekan Apakah dokumen tersebut termasuk di dalam sebuah kelas yang telah ditentukan sebelumnya.
b)	`private static void SetClass (int loop)`	Melakukan set kelas – kelas yang terbentuk berdasarkan label- label yang ditentukan sebelumnya.
c)	`protected static double GetErrorRasio ()`	Melakukan perhitungan *error ratio* berdasarkan rumus 2.8
d)	`public static double ErrorRasio()`	Menjalankan algoritma *error ratio* `(GetErrorRasio())`

a) Melakukan perhitungan *error ratio* berdasarkan rumus 2.8

Proses perhitungan *error ratio* dapat dilihat pada kode program 4.50.

```
protected static double GetErrorRasio() throws Exception {
        int temp = 0; int index,idE; String add;
        for (int i=0; i<Global.Clusters.size(); i++) {
            for                    (int                    j=0;
j<Global.Clusters.get(i).GetLength(); j++) {
                idE = Global.Clusters.get(i).GetElement(j) -
1;
                add                                          =
MainForm.DaftarFile.get(idE).Alamat.trim().replace(MainForm.
DaftarFile.get(idE).Nama, "");
                index = ErrorRasio.GetClass(add);
                if (i != index) {
                    temp += 1;
                }
            }
        }
        return (1.0 * temp) / Global.TotalDoc;
    }
```

Kode Program 4. 50 Kode program perhitungan error ratio.

b) Menjalankan algoritma *error ratio* .

Proses menjalankan algoritma *error ratio* dapat dilihat pada kode program 4.51.

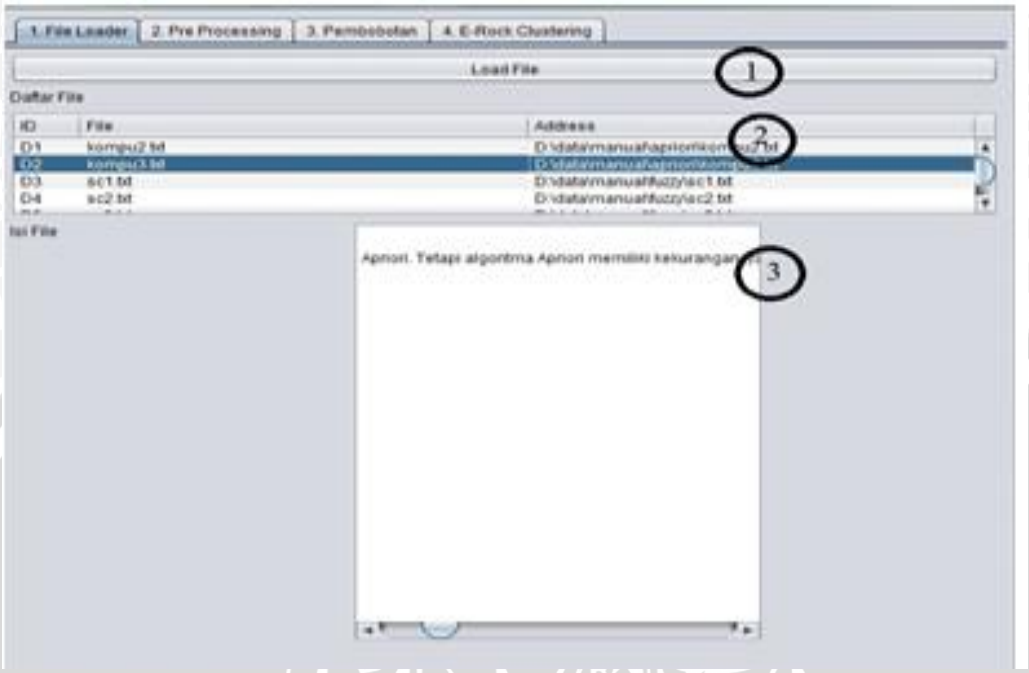
```
public static double ErrorRasio() throws Exception {
        double result = 1.0;
        double temp;
        for (int i=0; i<Global.jumCluster; i++) {
            ErrorRasio.SetClass(i);
            temp = ErrorRasio.GetErrorRasio();
            if (result > temp) { result = temp; }
        }
        return result * 100;
    }
```

Kode Program 4. 51 Kode program menjalankan algoritma error ratio.

4.3 Implementasi Antar Muka (*Interface*).

Implementasi antar muka (*interface*) terbagi menjadi 4 tab, yaitu:

a) *Tab File Loader*.

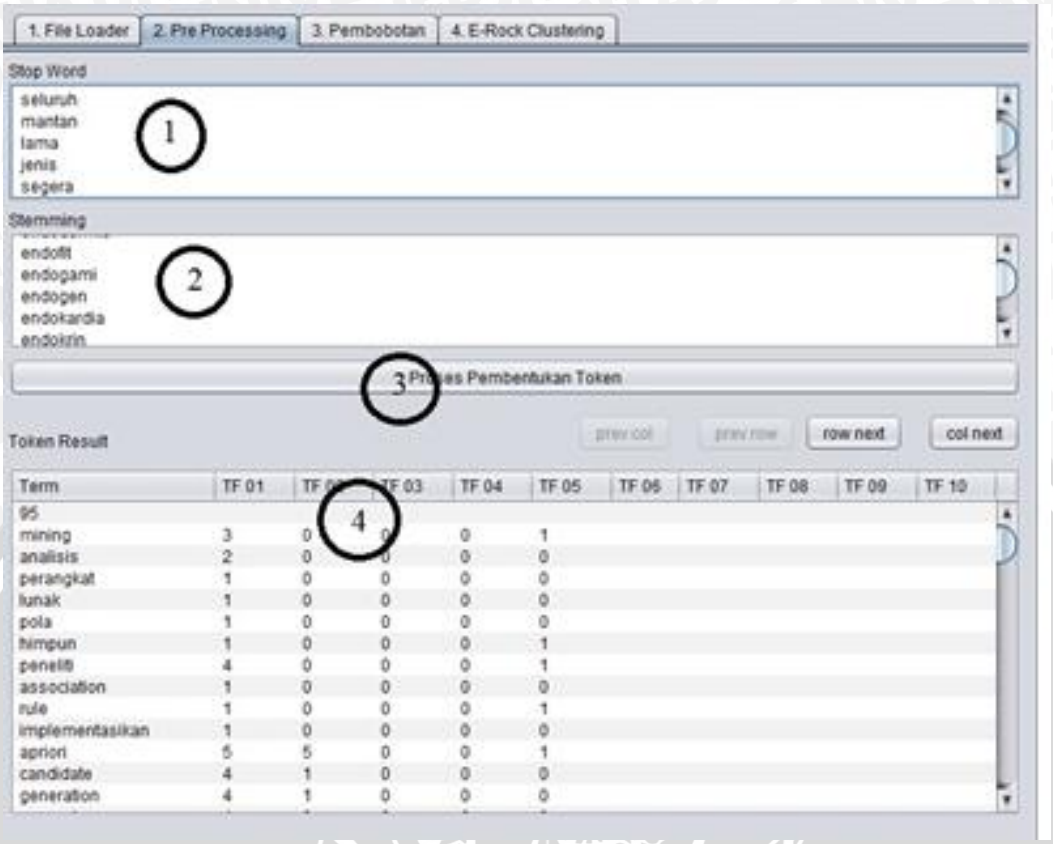


Gambar 4. 1 Tampilan *interface* tab *file loader*

Keterangan dari gambar 4.1 adalah sebagai berikut

1. *Button load file*. Merupakan *button* dimana memilih folder data latih disimpan.
2. Tabel Dokumen. Berisi nama – nama dokumen yang dipilih oleh *button load file*.

b) *Tab Pre Processing*.

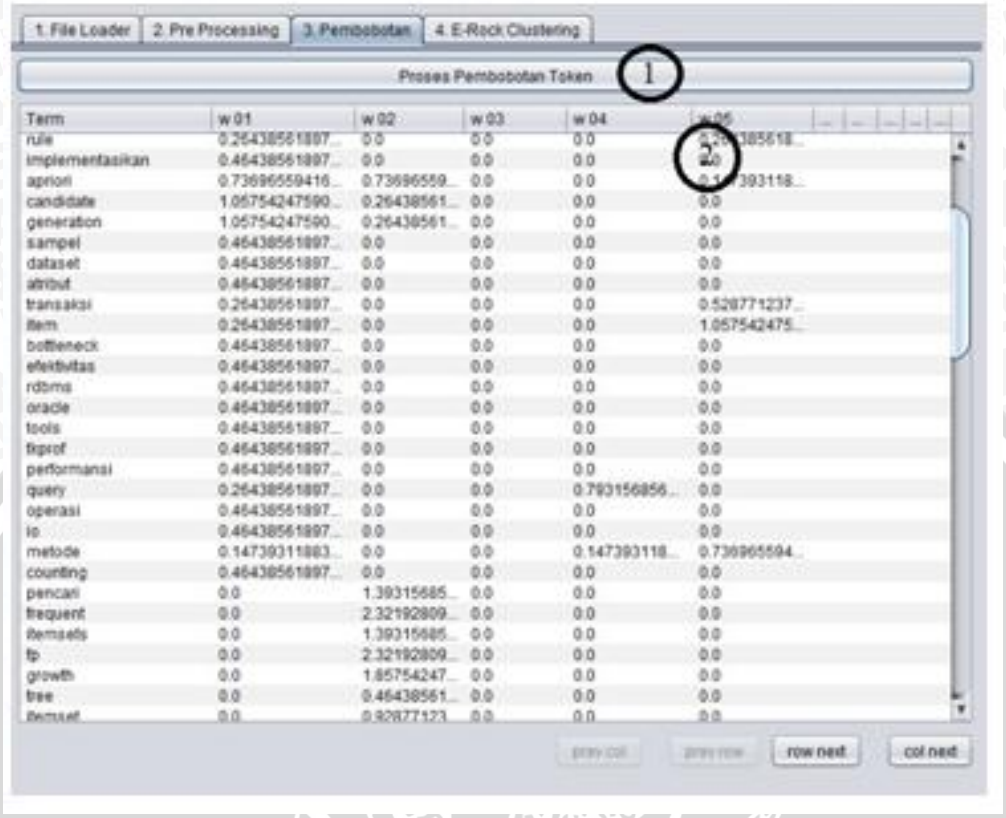


Gambar 4. 2 Tampilan *interface* tab pre processing.

Keterangan gambar 4.2 adalah sebagai berikut:

1. *Text area* stopilst. Berisi daftar- daftar stoplist yang disimpan dalam *stoplist.txt*
2. *Text area* kamus. Berisi kata dasar yang diperoleh dari Kamus Besar Bahasa Indonesia (KBBI) *offline* dan disimpan dalam *kamus.txt*.
3. *Button* proses pembentukan token. Merupakan button yang memproses kata-kata dalam dokumen menjadi terpisah- pisah.
4. Tabel *term frequence (tf)* dokumen. Berisi tentang kata- kata beserta frekuensi di tiap dokumen.

c) Tab Pembobotan.

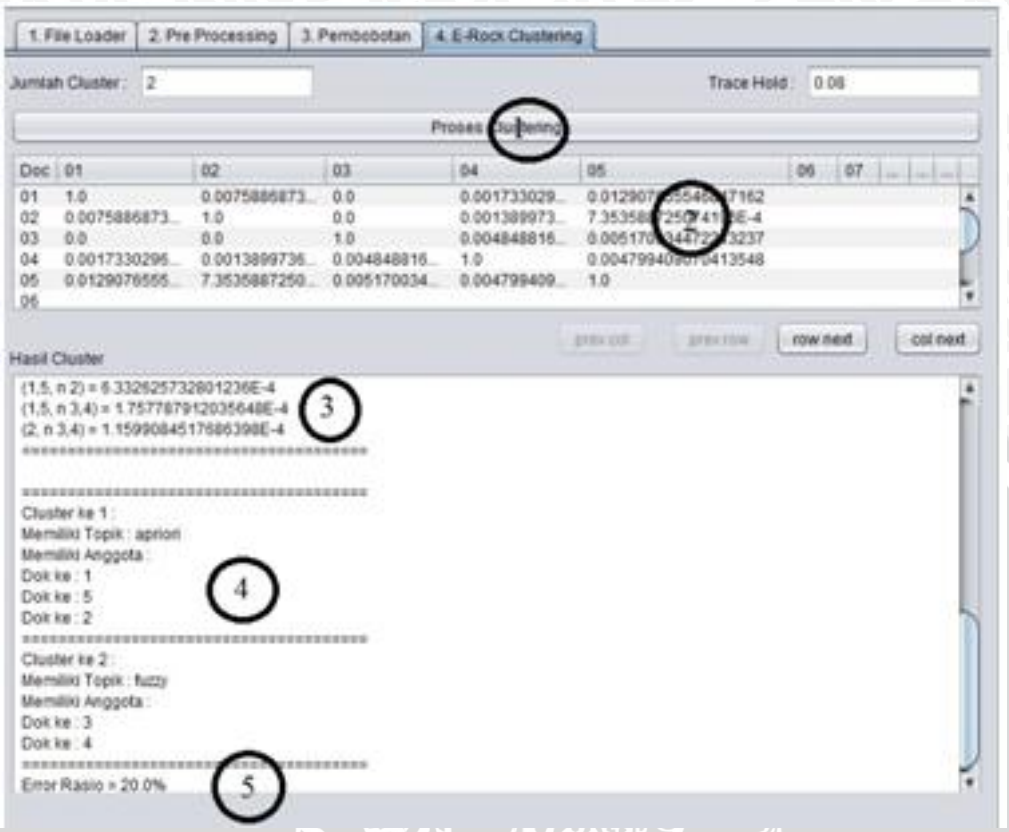


Gambar 4. 3 Tampilan *interface* tab pembobotan.

Keterangan gambar 4.3 adalah sebagai berikut :

1. *Button* proses pembentukan token. Merupakan button yang memproses perhitungan nilai bobot *(tf-idf)* tiap- tiap term dalam tiap- tiap dokumen.
2. Tabel Pembobotan. Berisi tentang kata- kata beserta nilai bobot *(tf- idf)* pada masing- masing dokumen.

d) *Tab E-Rock Clustering*.



Gambar 4. 4 Tampilan *interface* tab E-Rock clustering.

Keterangan gambar 4.4 adalah sebagai berikut :

1. *Button* proses clustering. Merupakan button untuk mengelompokkan dokumen menjadi cluster- cluster dan melakukan pembentukan topik.
2. Tabel *cosine similarity*.Berisi nilai *link* antar dokumen.
3. *Text area* tampilan perhitungan dan jumlah iterasi *goodness measure.*
4. *Text area* tampilan anggota tiap- tiap *cluster* beserta topik yang dibentuk.
5. *Text area* tampilan *error ratio* hasil *clustering*.

4.4 Implementasi Uji Coba

Dalam sub bab ini akan dibahas mengenai implementasi dari metode pengujian yang telah dilakukan oleh sistem dan hasil dari pengujian tersebut.

4.4.1 Skenario Evaluasi

Skenario evaluasi sistem pembentukan topik ini menggunakan 100 dokumen artikel jurnal ilmiah ilmu komputer. Dokumen–dokumen tersebut terbagi dalam 10 kategori topik yang telah melalui proses pengkategorian melalui ahli bahasa, yakni seperti yang terdapat pada tabel 4.11 sebagai berikut:

Tabel 4. 11 Tabel Kategori topik.

No	Kategori Topik	Anggota dokumen	Jumlah dokumen
1	Genetika	*1.txt – 10.txt*	10
2	Apriori	*11.txt – 20.txt*	10
3	Backpropagation	*21.txt – 30.txt*	10
4	DNS	*31.txt – 40.txt*	10
5	Firewall	*41.txt – 50.txt*	10
6	Kohonen	*51.txt – 60.txt*	10
7	SQL	*61.txt – 70.txt*	10
8	Fuzzy	*71.txt – 80.txt*	10
9	Kriptografi	*81.txt – 90.txt*	10
10	PHP	*91.txt – 999.txt*	10

Label dari tiap kategori topik disimpan dalam bentuk *file* teks, dimana setiap *file* melambangkan satu kategori topik dan berisi nama dokumen yang termasuk dalam kategori tersebut. *File* label dari tiap data latih disimpan dalam satu *folder*.

Pengujian pertama dilakukan untuk mengetahui pengaruh *threshold* terhadap *error ratio*yang dihasilkan pada jumlah cluster yang terbentuk. Sebelumnya telah ditentukan jumlah cluster yang diinginkan adalah sebanyak 10 cluster. Sedangkan rentang threshold yang digunakan adalah mulai 0.1 sampai dengan 0.95 dengan tiap-tiap kenaikan sebesar 0,05.

Pengujian kedua dilakukan untuk mengetahui pengaruh *threshold* terhadap *error ratio* dari topikyang dihasilkan. Sebelumnya telah ditentukan jumlah cluster yang diinginkan adalah sebanyak 10 *cluster*. Sedangkan rentang *threshold* yang digunakan adalah mulai 0.1 sampai dengan 0.95 dengan tiap-tiap kenaikan sebesar

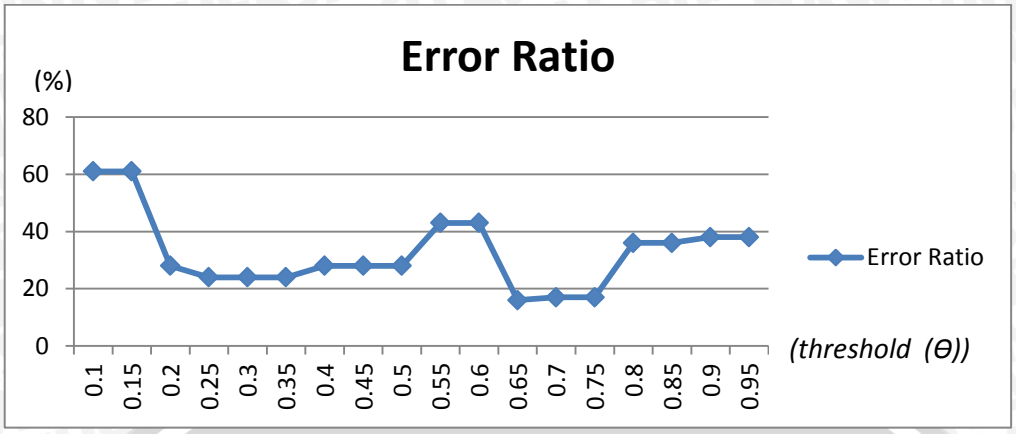
0,05.Nilai *threshold* di *set* kurang dari 1 dikarenakan pada nilai 1 nilai yang akan dihasilkan adalah statis atau sama. Rentang *threshold* yang dinaikkan setiap 0.05 dikarenakan setiap kenaikan 0.05 pembentukan *cluster* tiap- tiap dokumen yang terjadi berubah- ubah [ADI-10].

4.4.2 Hasil Evaluasi

Pengujian tahap pertama dilakukan untuk mengetahui pengaruh *threshold* terhadap *error ratio* yang dihasilkan pada jumlah cluster yang terbentuk. Sebelumnya telah ditentukan jumlah *cluster* yang diinginkan adalah sebanyak 10 *cluster*. Sedangkan rentang threshold yang digunakan adalah mulai 0.1 sampai dengan 0.95 dengan tiap-tiap kenaikan sebesar 0,05.Hasil pengujian ditampilkan dalam tabel 4.12.

Tabel 4. 12 Hasil pengujian tahap pertama.

Threshold	*Error Ratio (%)*	Jumlah iterasi
0.1	61	89
0.15	61	89
0.2	28	89
0.25	24	89
0.3	24	89
0.35	24	89
0.4	28	89
0.45	28	89
0.5	28	89
0.55	43	89
0.6	43	89
0.65	16	89
0.7	17	89
0.75	17	89
0.8	36	89
0.85	36	89
0.9	36	89
0.95	36	89
Average	32.778	89



Gambar 4. 5 Grafik hubungan threshold dan error ratio pada saat jumlah cluster sebanyak 10

Berdasarkan grafik yang terdapat pada gambar 4.5 diketahui nilai *error ratio* minimum terdapat pada *threshold* 0.65 dengan nilai sebesar 16. Sedangkan untuk nilai *error ratio* maksimum terdapat pada *threshold* 0.1 dan 0.15 dengan nilai sebesar 61%. Untuk jumlah iterasi dimulai dari rentang *threshold* 0.1 sampai dengan 0.95 bernilai sama, yaitu sebanyak 89 kali. Sehingga didapatkan *average error ratio* didapatkan nilai sebesar 32.778% dengan *average* jumlah iterasi sebanyak 89 kali.

Nilai *error ratio* yang berubah-ubah pada tiap *threshold* disebabkan oleh pembentukan *cluster* yang berisi dokumen-dokumen dengan masing- masing dokumen hanya memiliki satu kategori topik. Dapat dilihat pada gambar 4.4 bahwa tiap-tiap *cluster* yang terbentuk memiliki anggota yang terdiri dari dokumen-dokumen yang terdapat data latih. Dapat dilihat pula pada tabel 4.11 bahwa telah ditentukan sebelumnya jenis dokumen beserta topiknya.

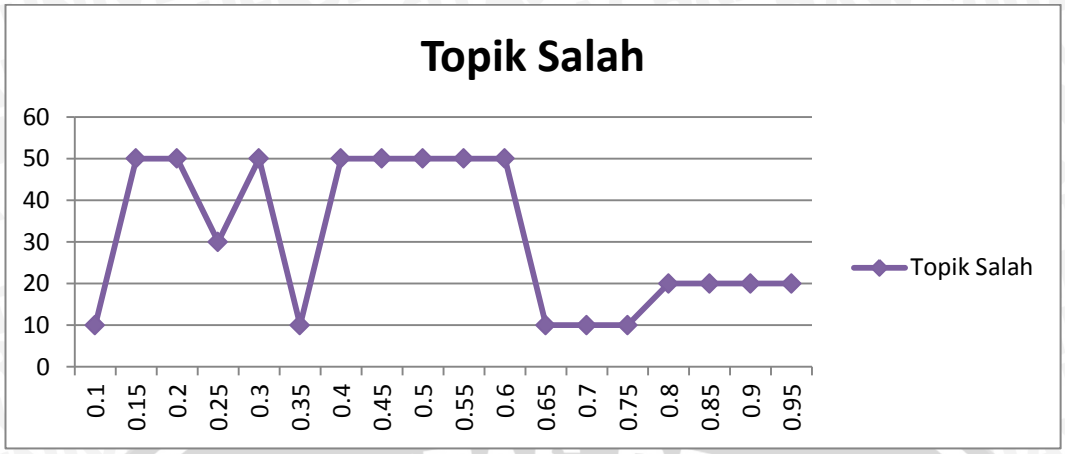
Pada proses pembentukan *cluster* dengan nilai *threshold* sebagai acuannya, dihasilkan *cluster-cluster* topik beserta dokumen-dokumen anggotanya. Akan tetapi permasalahan yang terjadi adalah kadang dokumen tidak pada tempat seharusnya, yaitu berdasarkan topik-topik yang telah ditentukan sebelumnya.Hal tersebut dikarenakan karena nilai kesamaan antar dokumen yang dihasilkan pada tiap-tiap pasangan dokumen mempengaruhi proses pembentukan *cluster*.Untuk lebih jelasnya dapat dilihat pada lampiran 4.

Sehingga dapat disimpulkan bahwa grafik hubungan *threshold* dan *error ratio* pada saat jumlah *cluster* sebanyak 10 memperlihatkan angka *threshold* mempengaruhi pembentukan nilai *error ratio* akan tetapi pola yang dihasilkan oleh grafik tidak mengalami kenaikan atau penurunan yang statis sehingga dapat disimpulkan hubungan *threshold* dan *error ratio* pada saat jumlah *cluster* sebanyak 10 menghasilkan grafik tidak berpola dan nilai *error ratio* yang terbentuk berdasarkan tingkat kebenaran penempatan dokumen yang terdapat pada tiap-tiap *cluster*, sedangkan nilai *threshold* mempengaruhi proses penempatan dokumen-dokumen tersebut pada tiap-tiap *cluster* yang terbentuk

Pengujian kedua dilakukan untuk mengetahui pengaruh *threshold*terhadap *error ratio* dari topikyang dihasilkan. Sebelumnya telah ditentukan jumlah cluster yang diinginkan adalah sebanyak 10 cluster. Sedangkan rentang threshold yang digunakan adalah mulai 0.1 sampai dengan 0.95 dengan tiap-tiap kenaikan sebesar 0,05. Hasil pengujian ditampilkan dalam tabel 4.13.

Tabel 4. 13 Hasil pengujian tahap kedua.

Threshold	*Error Ratio (%)*	Topik salah
0.1	61	0.1
0.15	61	0.5
0.2	28	0.5
0.25	24	0.3
0.3	24	0.5
0.35	24	0.1
0.4	28	0.5
0.45	28	0.5
0.5	28	0.5
0.55	43	0.5
0.6	43	0.5
0.65	16	0.1
0.7	17	0.1
0.75	17	0.1
0.8	36	0.2
0.85	36	0.2
0.9	36	0.2
0.95	36	0.2
Average		0.3



Gambar 4. 6 Grafik hubungan antara nilai threshold dan topik salah yang terbentuk pada saat cluster sebanyak 10.

Berdasarkan grafik pada gambar 4.6 diketahui bahwa kesalahan pembentukan topik minimum terjadi pada saat *threshold* ada pada kisaran 0.1; 0.35; 0.65; 0.7; 0.75 yaitu sebesar 0.1 atau 10%. Sedangkan kesalahan pembentukan topik maksimum terjadi pada saat *threshold* ada pada kisaran 0.15; 0.2; 0.3; 0.4; 0.45;0.5;0.55;dan 0.6 yaitu sebesar 0.1 atau 10%. Untuk average kesalahan pembentukan topik yang dibentuk oleh algoritma *E-ROCK* adalah sebesar 0.3 atau 30%.

Untuk hubungan antara *error ratio* dan topik salah yang terbentuk adalah dapat dilihat pada gambar 4.7

Gambar 4. 7 Grafik hubungan antara error ratio dengan topik salah yang terbentuk pada saat jumlah cluster sebanyak10.

Berdasarkan grafik pada gambar 4.7 dapat diketahui bahwa kesalahan pembentukan topik minimum terjadi sebesar 0.1 atau 10% pada *error ratio* minimum sebesar 16 %. Sedangkan untuk pembentukan kesalahan topik maksimum terjadi sebesar 0.5 atau 50% pada error ratio maksimum sebesar 61%. Berdasarkan grafik-grafik yang terdapat pada gambar 4.6 dan 4.7 diketahui hubungan antara nilai *threshold*, *error ratio*, dan kesalahan pembentukan kata pada topik yang dihasilkan oleh sistem adalah nilai *error ratio* yang terbentuk berdasarkan tingkat kebenaran penempatan dokumen yang terdapat pada tiap-tiap *cluster*, sedangkan nilai *threshold* mempengaruhi proses penempatan dokumen-dokumen tersebut pada tiap-tiap *cluster* yang terbentuk. Pada proses pembentukan *cluster* dengan nilai *threshold* sebagai acuannya, dihasilkan *cluster-cluster* topik beserta dokumen-dokumen anggotanya. Akan tetapi permasalahan yang terjadi adalah kadang dokumen tidak pada tempat seharusnya, yaitu berdasarkan topik-topik yang telah ditentukan sebelumnya.Hal tersebut dikarenakan karena nilai kesamaan antar dokumen yang dihasilkan pada tiap-tiap pasangan dokumen mempengaruhi proses pembentukan *cluster*.Untuk lebih jelasnya dapat dilihat pada lampiran. Selain itu frekuensi term terbesar yang dimiliki tiap-tiap *cluster* mempengaruhi proses pembentukan kata yang dijadikan topik pada tiap-tiap *cluster*. Frekuensi *term* terbesar diambil dari *max term frequence* yang dimiliki oleh anggota dokumen yang ada dalam tiap-tiap *cluster*.

BAB V
PENUTUP

5.1 Kesimpulan

Kesimpulan yang diperoleh dari penelitian ini adalah sebagai berikut:

1) Sistem pembentukan topik pada artikel jurnal ilmu komputer berbahasa Indonesia menggunakan algoritma *E-ROCK (Enhanched- RObust Clustering using linKs)* dilakukan melalui tahap utama, yaitu *preprocessing*, perhitungan bobot, pembentukan cluster menggunakan algoritma *E-ROCK (Enhanched - RObust Clustering using linKs)*, dan pembentukan topik berdasarkan *term frequence* (*tf*) terbesar pada tiap-tiap *cluster*.
2) Didapatkan hasil *error ratio* minimum yang sebesar 16% dengan tingkat kesalahan pembentukan topik minimum sebesar 0.1 atau 10% pada saat *threshold* sebesar 0.65 . Sedangkan *error ratio* maksimum yang dihasilkan sebesar 61% dengan tingkat kesalahan pembentukan topik paling besar sebesar 0.5 atau 50% pada saat *threshold* sebesar 0.15 . Nilai *average error ratio* yang dihasilkan oleh agoritma *E-ROCK (Enhanched - RObust Clustering using linKs)* adalah sebesar 32.778% Sedangkan nilai *average* kesalahan pembentukan kata yang dijadikan topik yang dihasilkan oleh agoritma *E-ROCK (Enhanched - RObust Clustering using linKs)* adalah sebesar 0.3 atau 30%. Secara umum hubungan *threshold* dan *error ratio* pada saat jumlah *cluster* sebanyak 10 memperlihatkan angka *threshold* mempengaruhi pembentukan nilai *error ratio* akan tetapi pola yang yang ditunjukkan oleh grafik tidak mengalami kenaikan atau penurunan yang statis sehingga dapat disimpulkan hubungan *threshold* dan *error ratio* menghasilkan grafik tidak berpola dan nilai *error ratio* yang terbentuk berdasarkan tingkat kebenaran penempatan dokumen yang terdapat pada tiap-tiap *cluster*, sedangkan nilai *threshold* mempengaruhi proses penempatan dokumen-dokumen tersebut pada tiap-tiap *cluster* yang terbentuk.

3) Pembentukan topik yang dihasilkan agoritma *E-ROCK (Enhanched-RObust Clustering using linKs)* didasarkan pada frekuensi *term* terbesar yang dimiliki oleh tiap- tiap *cluster*. Akan tetapi karena pembentukan topik dilakukan hanya berdasarkan frekuensi *term* terbesar yang dimiliki oleh tiap- tiap *cluster* saja, sehingga seringkali muncul kata- kata yang tidak seharusnya menjadi topik. Selain itu pula kadang dihasilkan pembentukan topik yang sama pada *cluster* berbeda.

5.2 Saran

Pada penelitian ini terdapat beberapa hal yang dapat dikembangkan dan diperbaiki untuk penelitian lebih lanjut :

1) Pengolahan kata sebaiknya tidak hanya memperlakukan kata sebagai kata tunggal saja, namun perlu juga adanya penanganan frasa.
2) Kadangkala kata yang dijadikan topik pada tiap-tiap *cluster* adalah kata yang sama. Sehingga yang perlu dibenahi pada algoritma *E-ROCK (Enhanched - RObust Clustering using linKs)* ini adalah topik yang terbentuk tidak hanya memperhatikan jumlah *term frequence (tf)* saja, melainkan kata- kata yang berhubungan dengan topik tersebut.

DAFTAR PUSTAKA

[ADI-10] Adi, Heru Darmawan,dkk. 2010. ***Rancang Bangun Aplikasi Search Engine Tafsir Al-qur'an Menggunakan Text Mining dengan Algoritma VSM(Vector Space Model).***Surabaya : STIKOM.

[AGR-90] Agresti, Alan.1990. ***Categorial Data Analysis***. New York : John Willey & Sons.

[ALG-02] Al-Ghifari,Abu.2002.***Kiat Menjadi Penulis Sukses***. Bandung: Mujahid Press.

[ASI-05] Asian,Jelita, Wiliams, Hugh E, dan Tahaghoghi S.M.M. .2005. ***Stemming Indonesian***. Australia : School of Computer Science and Information Technology.

[BAR-06] Barakbakh, A.R.2006. ***Clustering***.Workshop Data Mining Jurusan teknologi Informasi Politeknik Elektronika Negeri Surabaya.

[BAR-10] Barakbah, A.R dan Kiyoki Y. 2010.***A Fast Algorithm for K-Means Optimization using Pillar Algorithm.***The 2nd International Workshop with Mentors on Databases, Web and Information Management for Young Researchers.

[BUD-10] Budhi, Georgious. S, dkk. 2010. ***Hierarchical Clustering Untuk Aplikasi Automated Text Integration.*** Jawa Timur, Surabaya : Universitas Kristen Petra Jurusan Teknik Informatika.

[CYR-60] Cyril Cleverdon, 1960. ***Report on the Testing and Analysis of an Investigation into the Comparative Efficiency of Indexing Systems,*** The College of Aeronautics, Cranfield.

[ELF-12] Elfitri, Ayu Rahayu.2012. ***Bingung Menentukan Topik Esai?IniTipsnya***. http://edukasi.kompas.com/read/2012/03/19/09295215/Bingung.Menentukan.Topik.Esai.Ini.Tipsnya. Tanggal akses 30 April 2012.

[FEL-07] Feldman,R dan James S. 2007. ***The Text Mining Handbook***.England.Cambridge University Press.

[FRA-92] Frakes, W.1992. ***Stemming algorithms,in W.Frakes & R.Baeza-Yates, eds, 'Information Retrieval : Data Structures and Algorithms'***.Prentice-Hall.

[GAR-06] Garcia,E.2006. ***SVD and LSI Tutorial 3: Computing the Full SVD of a Matrix***. http://www.miislita.com/information-retrieval-tutorial/svd-lsi-tutorial-3-full-svd.html Tanggal akses 7 Mei 2011.

[GUH-00] Guha,S.,Rajeev Rastogi, dan Kyuseok Shim. 2000. ***ROCK : A Robust Clustering Algorithm For Categorical Attributes.***Jurnal information system Vol.25, No.5 pp 354-366. http://citiseer.ist.psu.edu/126378.html.Tanggal akses 14 April 2009.

[HAR-06] Harlian, Milkha.2006. ***Text Mining***.Referensi Raymond J. Mooney CS. Machine Learning Text Categorization.University of Texas at Austin.

[HAS-08] Hasymi, Ali. 2008. ***Statistika Dasar: Konsep- Konsep Dasar Penelitian(bagian4)***.Pontianak. http://omegahat.blogspot.com/2008/01/konsep-konsep-dasar-penelitian-bagian4-html. Tanggal akses 20 Maret 2012

[HAY-10] Hayuhardika, Widhy Nugraha Putra.2010.***Sistem Peringkas Dokumen Teks Otomatis Berbahasa Indonesia dengan Metode Lexrank:Graph-Based Summarization Algorithm***.Malang :Universitas Brawijaya.

[KAN-03] Kantardzic,Mehmed. 2003. ***Data Mining : Concepts, Models, Methods and Algoithm.***John Willey & Sons. New Jersey.

[MAN-09] Manning, Christopher D, dkk.2009.***An Introduction to Information Retrieval.***Inggris : Cambridge University Press.

[NUR-04] Nurjanah , Nunuy. 2004.***Menulis Bertujuan***.Terjemahan dari Buku Writing With a Purpose McCrimmon, James M..1983. Boston: Houghton Mifflin Company.Bandung: Universitas Pendidikan Indonesia.

[PRA-10] Prasetyawan, Dedy.2010.***Klasifikasi Really Simple Syndication dan Penentuan Trend Topik Menggunakan Algoritma Key Nearest Neighbor***.Malang : Universitas Brawijaya.

[RAH-04] Rahmawati, Tuti.2004. ***Perancangan dan Pembuatan Aplikasi Clustering Dokumen Web dengan Metode Hierarcial Agglomerative Clustering Pendekatan Complete Link.***Surabaya : Institut Sepuluh November Surabaya Teknik Informatika.

[RES-07] Resmini, Novi.2007. ***Karangan Ilmiah dan Teknik Penulisan Karangan Ilmiah***.Bandung .Universitas Pendidikan Indonesia.

[RIF-95] Rifa'i, Mien A. 1995, ***Buku Pegangan Gaya Penulisan, Penyunting dan Penerbitan Karya Ilmiah Pegangan Gaya Penulisan.***Yogyakarta : Universitas Gajah Mada.

[RIZ-10] Rizwan Ahmad, **Aasia Khanum,2010**.***Document Topic Generation in Text Mining by Using Cluster Analysis with EROCK***, International Journal of Computer Science & Security CSC Journals.Malaysia: Kuala Lumpur.

[SIL-11] Silabus FMIPA Brawijaya.2011.***Program Studi Ilmu Komputer UniversitasBrawijaya***.http://mipa.ub.ac.id/uploads/DraftKuikulumIlmuKomputer2011_Revisi_15Mei2011.pdf..Tanggal akses 30 April 2012.

[SOU-03] Soucy , P.& Mineau, G.2003.***Feature Selection Strategies For Text Categorization.*** Canada:Halifax,New York.Inc.

[TAL-03] Tala, F.Z .2003.***Astudy of Stemming Effects on Information Retrieval in Bahasa Indonesia***.Master of Logic Project, Institute for Logic, Language and Computation . University Van Amsterdam.The Netherlands.

[TRI-09] Triawati,C. 2009.***Metode Pembobotan Statistical Concept Based untuk Klastering dan Kategorisasi Dokumen Berbahasa Indonesia.***Bandung :Institut Teknologi Telkom Bandung.

[WID-08] Widjono, Hs.2008.***Bahasa Indonesia Mata Kuliah Pengembangan Kepribadian di Perguruan Tinggi***. Jakarta: Grasindo.

LAMPIRAN

Lampiran 1 Daftar Stopword.

yang	oleh	Dia	id	alami
Di	menjadi	kalau	lewat	perubahan
Dan	orang	terjadi	belakang	hukuman
Itu	ia	banyak	ikut	ketentuan
Dengan	telah	menurut	barang	diperkirakan
Untuk	adalah	anda	meningkatkan	mengarah
Tidak	seperti	hingga	kejadian	diharapkan
Ini	sebagai	tak	kehidupan	memakai
Dari	bahwa	baru	keterangan	antar
Dalam	dapat	beberapa	penggunaan	berapa
Akan	para	ketika	masing	keseluruhan
Pada	harus	saja	menghadapi	memperhatikan
Juga	namun	jalan	menyebabkan	mengusulkan
Saya	kita	sekitar	sumber	berupa
Ke	satu	secara	studi	dikenal
Karena	masih	dilakukan	kasus	disimpulkan
tersebut	hari	sementara	motif	ditentukan
Bisa	hanya	tapi	kerja	diperlukan
Ada	mengatakan	sangat	menghitung	dibutuhkan
mereka	kepada	hal	deng	menemukan
Lebih	kami	sehingga	memilih	pengetahuan
Kata	setelah	seorang	buah	guna
Tahun	melakukan	bagi	pembentukan	mendukung
Sudah	lalu	besar	dengan	pengambilan
Atau	belum	lagi	aplikasi	dibahas
saat	lain	selama	nilai	permasalahan
antara	agar	sempat	fungsi	perbaikan
waktu	semua	perlu	perkenalkan	ditekankan
sebuah	sedang	menggunakan	pisah	daripada
jika	kali	memberikan	holland	two
sampai	kemudian	sedangkan	de	dua
jadi	hasil	langsung	jong	id
terhadap	sejumlah	apakah	1975	way
serta	juta	pihak	teknik	group
pun	persen	melalui	prinsip	by
salah	sendiri	diri	dasar	tiga
merupakan	katanya	mencapai	teori	mengukur
atas	demikian	aku	kembangkan	membuktikan
sejak	masalah	berada	penyusun	penghitungan
membuat	mungkin	tinggi	jadwal	rules
baik	umum	ingin	urut	kekurangan

memiliki	setiap	sebelum	penilaian	membutuhkan
kembali	bulan	tengah	keandalan	mengatasi
selain	bagian	kini	penentuan	digunakanlah
tetapi	bila	tahu	tata	menentukan
pertama	lainnya	bersama	letak	memerlukan
memang	terus	depan	peralatanmesin	menyajikan
pernah	luar	begitu	pabrik	pembahasan
apa	cukup	merasa	distribusi	membahas
mulai	termasuk	berbagai	pergerakan	memecahkan
sama	sebelumnya	mengenai	perencanaan	pilihan
tentang	bahkan	maka	transportasi	menunjukkan
bukan	wib	jumlah	anggaran	diperhatikan
masuk	rasa	posisi	pemeliharaan	dibandingkan
katanya	maupun	asal	penerapan	pengembangan
mengalami	seluruh	sekali	pendahuluan	yudhistira
sering	mantan	sesuai	bersifat	dikembangkan
ujar	lama	sebesar	et	menangani
kondisi	jenis	berat	al	seringkali
akibat	segera	dirinya	1997	mengimplementasi
hubungan	misalnya	member	metoda	disebut
paling	mendapat	pagi	klasik	arti
mendapatkan	bawah	ternyata	memanfaatkan	tahani
selalu	jangan	mencari	maksimum	tugas
meminta	meski	sumber	generasi	membantu
melihat	terlihat	ruang	makalah	semangat
sekarang	akhirnya	menunujukkan	contoh	menjadi
mengaku	punya	biasanya	tahap	menganalisis
mau	yakni	nama	sinus	alami
kerja	Terakhir	Sebanyak	mempengaruhi	perubahan
acara	Kecil	Utara	ketelitian	hukuman
menyatakan	Panjang	Berlangsung	rinci	ketentuan
masa	Badan	Barat	rincian	diperkirakan
proses	Jelas	Kemungkinan	mengoperasikan	mengarah
tanpa	Jauh	Yaitu	solusi	diharapkan
selatan	Tentu	Berdasarkan	titik	memakai
sempat	Semakin	Sebenarnya	selanjutnya	antar
adanya	Tinggal	Cara	chromosome	berapa
hidup	Kurang	Utama	data	keseluruhan
datang	Mampu	Pekan	support	memperhatikan
terlalu	Khusus	Upaya	aturan	mengusulkan
membawa	Bentuk	Mengetahui	algoritma	berupa
kebutuhan	Ditemukan	Mempunyai	kendaraan	dikenal
suatu	Diduga	Berjalan	pengaturan	disimpulkan
menerima	Mana	Menjelaskan	jaringan	ditentukan
penting	Ya	Mengambil	dimaksudkan	diperlukan

tanggal	Kegiatan	Benar	mendiskusikan	dibutuhkan
bagaimana	Sebagian	Lewat	berulang	menemukan
terutama	Tampil	Belakang	ulang	pengetahuan
tingkat	Hamper	Ikut	berbeda	guna
awal	Bertemu	Barang	kemiringan	mendukung
sedikit	Usai	Meningkatkan	konvergensi	pengambilan
nanti	Berarti	Kejadian	diinginkan	dibahas
pasti	Keluar	Kehidupan	mengevaluasi	permasalahan
muncul	Pula	Keterangan	diarahkan	perbaikan
dekat	Digunakan	Penggunaan	arahan	ditekankan
Lanjut	Justru	Masing-masing	lembah	daripada
Ketiga	Padahal	menghadapi	bukit	two
Biasa	Menyebutkan	Menyebabkan	diperkirakan	dua
Dulu	Gedung	sumber	menuju	
Kesempatan	Apalagi	way	menghasilkan	
Ribu	Program	group	ekstrim	
Akhir	Milik	by	global	
Membantu	Teman	tiga	mempertahan	
Terkait	Menjalani	mengukur	ketetapan	
Sebab	Keputusan	membuktikan	ditetapkan	

Lampiran 2 Contoh Dokumen

- ✓ **Isi Dokumen 2 Berkategori Apriori.**

Algoritma yang umum digunakan dalam proses pencarian frequent itemsets (data yang paling sering muncul) adalah Apriori. Tetapi algoritma Apriori memiliki kekurangan yaitu membutuhkan waktu yang lama dalam proses pencarian frequent itemsets. Untuk mengatasi hal tersebut maka digunakanlah algoritma FP-Growth. Dalam makalah ini akan dibahas penerapan Apriori dan FP-Growth dalam proses pencarian frequent itemsets. Penggunaan FP-Tree yang digunakan bersamaan dengan algoritma FP-growth untuk menentukan frequent itemset dari sebuah database, berbeda dengan paradigma Apriori yang memerlukan langkah candidate generation, yaitu dengan melakukan scanning database secara berulang-ulang untuk menentukan frequent itemset. Makalah ini juga menyajikan pembahasan mengenai perbandingan kompleksitas waktu antara algoritma FP-growth dengan Apriori dan hasil dari perbandingan algoritma tersebut.

- ✓ **Isi Dokumen 3 Berkategori Fuzzy.**

Makalah ini membahas mengenai masalah pengambilan keputusan untuk memilih alternatif karyawan terbaik. Untuk memecahkan masalah ini digunakan Fuzzy Analytical Hierarchy Process, Pilihan karyawan terbaik dengan Fuzzy Analytical Hierarchy Process menunjukkan bahwa subjektifitas kriteria sangat diperhatikan dibandingkan dengan menggunakan Analytical Hierarchy Process konvensional. Konsep fuzzy yang dipakai dalam pengembangan AHP ini adalah model Fuzzy AHP dengan pembobotan nonadditive yang dikembangkan oleh Yudhistira.

- ✓ **Isi Dokumen 4 Berkategori Fuzzy.**

Sistem database yang ada sampai sekarang, hanya mampu menangani data yang bersifat pasti(crisp), deterministik dan presisi. Padahal, dalam kondisi nyata seringkali dibutuhkan data yang samar untuk proses pengambilan

keputusan. Untuk mengatasi pengambilan keputusan yang membutuhkan variabel- variabel yang memiliki nilai yang samar dapat menggunakan logika fuzzy. Proyek akhir ini akan mengimplementasikan logika fuzzy ke dalam query, yang disebut Fuzzy Query Database. Artinya, suatu query yang memiliki variabel variabel yang bernilai fuzzy. Untuk mendapatkan informasi tentang kinerja pegawai, maka dibutuhkan kriteria - kriteria absensi dan kinerja pegawai yang bernilai ambigu. Data kriteria berdasarkan pada nilai derajat keanggotaan tiap pegawai yang telah diproses dengan proses metode fuzzy tahani. Dengan adanya tugas akhir ini diharapkan dapat membantu mendapatkan informasi tentang kinerja pegawai dan bisa memberikan semangat bagi pegawai untuk menjadi lebih baik.

- ✓ **Isi Dokumen 5 Berkategori Fuzzy.**

Sebagai salah satu aplikasi data mining, market basket analysis umumnya dilakukan dengan memakai metode Apriori. Metode ini mencari asosiasi antar item dengan hanya menghitung berapa kali item-item tersebut muncul dalam keseluruhan transaksi tanpa memperhatikan quantitas item dalam transaksi. Oleh karena itu, peneliti mengusulkan metode Fuzzy c-Covering. Fuzzy c-Covering merupakan salah satu metode yang dipakai untuk mengklasifikasikan elemen elemen dari suatu himpunan universal menjadi partisi-partisi berupa fuzzy sets. Fuzzy c-Covering sendiri merupakan generalisasi dari metode fuzzy cpartition yang telah dikenal sebelumnya. Dari hasil pengujian, dapat disimpulkan bahwa semakin kecil minimum support dan confidence yang ditentukan, semakin banyak rule yang dapat dihasilkan dan waktu yang diperlukan semakin banyak. Selain itu, semakin tinggi jumlah kombinasi yang dicari, semakin sedikit waktu yang dibutuhkan.

Lampiran 3 Surat Pernyataan

Lampiran 4 Perhitungan pembentukan *cluster* menggunakan rumus *goodness measure* menggunakan *threshold.*

Berdasarkan rumus 2.7 yang terdapat pada bab 2, dilakukan perhitungan *goodness measure* dengan menggunakan rentang *threshold* 0.06 dan 0.08 untuk mengetahui pengaruh *threshold* terhadap pembentukan *cluster*, data yang digunakan adalah data manual pada perhitungan bab 3 dan didapatkan:

Tabel *goodness measure* iterasi 0 pada rentang *threshold* 0.06

Pair	*Goodness Measure*
D1,D2	0.00233
D1,D3	0,000179
D1,D4	0.000383
D1,D5	0.000885
D2,D3	0.000415
D2,D4	0.00059
D2,D5	0.00058
D3,D4	0.000835
D3,D5	0.000656
D4,D5	0.000888

Pembentukan *cluster* pada iterasi 0 adalah pada dokumen 1 dan 2.

Tabel *goodness measure* iterasi 0 pada rentang *threshold* 0.08

Pair	*Goodness Measure*
D1,D2	0.00168
D1,D3	0
D1,D4	0.000383
D1,D5	0.00285
D2,D3	0
D2,D4	0.000307
D2,D5	0.00016288
D3,D4	1.00107
D3,D5	0.00114515
D4,D5	0.001063

Pembentukan *cluster* pada iterasi 0 adalah pada dokumen 1 dan 5.

Dokumen 1 dan 2 a dalah dokumen dengan 1 kategori topik, yaitu apriori, sedangkan untuk dokumen 3, 4, 5 memiliki kategori topik fuzzy. Disini dapat

dilihat terdapat kesalahan pembentukan anggota cluster pada rentang *threshold* 0.08 pada iterasi 0.