TUGAS AKHIR – SM141501

IDENTIFIKASI PENGGUNA MEDIA SOSIAL YANG BERPENGARUH BERDASARKAN *GRAPH* DENGAN METODE *SOCIAL NETWORK ANALYSIS*

SITI NUR DIANA
NRP 1213 100 057

Dosen Pembimbing
Alvida Mustika Rukmi, S.Si, M.Si

DEPARTEMEN MATEMATIKA
Fakultas Matematika dan Ilmu Pengetahuan Alam
Institut Teknologi Sepuluh Nopember
Surabaya 2017

“Halaman ini sengaja dikosongkan”

FINAL PROJECT – SM141501

IDENTIFICATION OF INFLUENTIAL SOCIAL MEDIA USERS BASED ON GRAPH USING SOCIAL NETWORK ANALYSIS METHOD

SITI NUR DIANA
NRP 1213 100 057

Supervisor
Alvida Mustika Rukmi, S.Si, M.Si

DEPARTMENT OF MATHEMATICS
Fakultas Matematika dan Ilmu Pengetahuan Alam
Institut Teknologi Sepuluh Nopember
Surabaya 2017

“Halaman ini sengaja dikosongkan”

LEMBAR PENGESAHAN

IDENTIFIKASI PENGGUNA MEDIA SOSIAL YANG BERPENGARUH BERDASARKAN *GRAPH* DENGAN METODE *SOCIAL NETWORK ANALYSIS*

IDENTIFICATION OF INFLUENTIAL SOCIAL MEDIA USERS BASED ON GRAPH USING SOCIAL NETWORK ANALYSIS METHOD

Diajukan Untuk Memenuhi Salah Satu Syarat
Untuk Memperoleh Gelar Sarjana Sains
Pada Bidang Studi Ilmu Komputer
Program Studi S-1 Departemen Matematika
Fakultas Matematika dan Ilmu Pengetahuan Alam
Institut Teknologi Sepuluh Nopember Surabaya

Oleh :
Siti Nur Diana
NRP. 1213 100 057

Menyetujui,

Dosen Pembimbing,

Alvida Mustika Rukmi, S.Si, M.Si
NIP. 19720715 199802 2 001

Mengetahui,
Kepala Departemen Matematika
FMIPA ITS

Dr. Imam Mukhlash, S.Si, MT
NIP. 19700831 199403 1 003

Surabaya, Juli 2017

“Halaman ini sengaja dikosongkan”

IDENTIFIKASI PENGGUNA MEDIA SOSIAL YANG BERPENGARUH BERDASARKAN *GRAPH* DENGAN METODE *SOCIAL NETWORK ANALYSIS*

Nama Mahasiswa	: Siti Nur Diana
NRP	: 1213 100 057
Departemen	: Matematika
Dosen Pembimbing	: Alvida Mustika Rukmi, S.Si, M.Si

Abstrak

Peningkatan penggunaan internet berdampak pada maraknya media sosial. Data percakapan sebuah topik tertentu pada media sosial dapat digunakan untuk mengetahui pola interaksi pengguna media sosial. Salah satu media sosial yang digunakan dan mengalami peningkatan yang pesat adalah *twitter*. Relasi antar pengguna *twitter* dapat direpresentasikan berupa *graph. Graph* yang terbentuk dianalisis dengan metode *social network analysis* (SNA) untuk mengidentifikasi pengguna yang berpengaruh. Dalam penelitian ini dihasilkan pengguna yang berpengaruh berdasarkan pengukuran pada SNA. Pengguna dengan *normalized degree centrality* yang tinggi adalah pengguna yang memiliki kolaborasi paling banyak dengan pengguna yang lain, dengan nilai 0.9907407407407407 yaitu $qu4d. Pengguna dengan *normalized closeness centrality* paling tinggi adalah pengguna yang memiliki kedekatan dengan pengguna yang lain, dengan nilai 1 yaitu $qu4d. Pengguna dengan *normalized betweeness centrality* yang paling tinggi adalah pengguna yang memiliki penghubung antar pengguna satu dengan pengguna yang lain, dengan nilai 0.06487318077207006 yaitu $qu4d. Pengguna yang berpengaruh adalah pengguna yang mempunyai *normalized degree centrality, normalized closeness centrality* dan *normalized betweeness centrality* yang tinggi dibanding dengan pengguna yang lain adalah $qu4d.

Kata Kunci : Media Sosial, Graph, Social Network Analysis.

“Halaman ini sengaja dikosongkan”

IDENTIFICATION OF INFLUENCIAL SOCIAL MEDIA USERS BASED ON GRAPH USING SOCIAL NETWORK ANALYSIS METHOD

Name of Student	: Siti Nur Diana
NRP	: 1213 100 057
Department	: Mathematics
Supervisor	: Alvida Mustika Rukmi, S.Si, M.Si

Abstract

The enhancement of the internet's usage has impact on the rise of social media. Conversation data of a particular topic in social media can be used to find out the pattern of social media user interaction. One of the social media which usually used and has rapid improvement is twitter. The relationship between twitter users can be represented in the form of graph. The graph that has formed analyzed by social network analysis (SNA) method to identify influential users. This study result an influential users based on the values of centrality. Users with the highest normalized degree of centrality is a user who has the most numerous collaborations with other users, with a value of 0.9907407407407407 i.e. $qu4d. Users with the highest normalized closeness of centrality is a user who has a high affinity with other users, with a value of 1 i.e. $qu4d. Users with the highest normalized betweeness of centrality is a user who has a liaison between users with other users, with a value of 0.06487318077207006 i.e. $qu4d. Influential users are who have the normalized degree centrality, normalized closeness centrality and normalized betweeness centrality higher than other users is $qu4d.

Keywords : Media Social, Graph, Social Network Analysis, Centrality.

“Halaman ini sengaja dikosongkan”

KATA PENGANTAR

Segala puji syukur penulis panjatkan ke hadirat Allah SWT, karena dengan ridlo-Nya penulis dapat menyelesaikan Tugas Akhir yang berjudul "Identifikasi Pengguna Media Sosial yang Berpengaruh Berdasarkan *Graph* dengan Metode *Social Network Analysis*" yang merupakan salah satu persyaratan akademis dalam menyelesaikan Program Sarjana Departemen Matematika, Fakultas Matematika dan Ilmu Pengetahuan Alam, Institut Teknologi Sepuluh Nopember Surabaya.

Tugas Akhir ini dapat diselesaikan dengan baik berkat kerja sama, bantuan, dan dukungan dari banyak pihak. Sehubungan dengan hal tersebut, penulis ingin mengucapkan terima kasih kepada :

1. Makruf selaku Ayah dan Siti Masfufah selaku ibu dari penulis yang telah memberikan doa, motivasi, semangat, tenaga, jasa yang tak terkira selama penulis mengerjakan Tugas Akhir.
2. Siti Masrifah selaku Kakak Perempuan dan Mukhammad Sofwan selaku Kakak Laki-laki dari penulis yang telah memberikan doa, motivasi, semangat, tenaga, jasa yang tak terkira selama penulis mengerjakan Tugas Akhir.
3. Dr. Imam Mukhlash, S.Si, MT selaku Kepala Departemen Matematika ITS.
4. Drs. Sadjidon, M.Si selaku Dosen Wali dan Penguji yang telah memberikan arahan akademik selama penulis menempuh pendidikan di Departemen Matematika ITS.
5. Alvida Mustika Rukmi, S.Si, M.Si selaku Dosen Pembimbing yang telah memberikan bimbingan dan motivasi kepada penulis dalam mengerjakan Tugas Akhir ini sehingga dapat terselesaikan dengan baik.
6. Dr. Didik Khusnul Arif, S.Si, M.Si selaku Kepala Program Studi S1 Departemen Matematika ITS.

7. Drs. Iis Herisman, M.Si selaku Sekretaris Program Studi S1 Departemen Matematika ITS.
8. Prof. DR. Mohammad Isa Irawan, MT, Dr. Chairul Imron, MI. Komp, Dian Winda Setyawati, S.Si, M.Si selaku Dosen Penguji yang telah memberikan saran dan perbaikan dalam Tugas Akhir ini.
9. Seluruh jajaran dosen dan staf Departemen Matematika ITS.
10. Lisa, Neni dan Nurma selaku sahabat yang selalu mengingatkan penulis untuk makan, menemani dalam suka dan duka selama mengerjakan Tugas Akhir.
11. Niken, Melynda, Eries, Mega, Ayur, Jessica, Ayu, Yenny, Ina, Wawan, Ivan, Gery selaku sahabat grup S.Si yang selalu menemani, memberikan motivasi selama mengerjakan Tugas Akhir.
12. Teman-teman Jurusan Matematika ITS angkatan 2013 yang saling mendukung dan memotivasi.
13. Semua pihak yang tak bisa penulis sebutkan satu-persatu, terima kasih telah membantu sampai terselesaikannya Tugas Akhir ini.

Penulis menyadari bahwa Tugas Akhir ini masih jauh dari kesempurnaan. Oleh karena itu, penulis mengharapkan kritik dan saran dari pembaca. Akhir kata, semoga Tugas Akhir ini dapat bermanfaat bagi semua pihak yang berkepentingan.

Surabaya, Juli 2017

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

“Halaman ini sengaja dikosongkan”

BAB I
PENDAHULUAN

Pada bab ini dibahas mengenai latar belakang yang mendasari penulisan Tugas Akhir ini. Di dalamnya mencakup identifikasi permasalahan pada topik Tugas Akhir kemudian dirumuskan menjadi permasalahan yang diberikan batasan-batasan dalam pembahasan pada Tugas Akhir ini.

1.1 Latar Belakang

Perkembangan teknologi komunikasi dan informasi dari tahun ke tahun semakin mengalami peningkatan. Hal tersebut tidak terlepas dengan adanya pertumbuhan internet secara global. Internet adalah kumpulan atau jaringan dari komputer yang ada diseluruh dunia. Menurut data [1] pada Agustus 2015 pengguna internet aktif di seluruh dunia mencapai 3,175 miliar dengan populasi penduduk dunia mencapai 7,357 miliar. Dilihat dari perkembangannya, jumlah pengguna internet bertumbuh hingga 7,6 persen. Pertumbuhan internet ini juga berpengaruh terhadap pertumbuhan pengguna media sosial dan *mobile*. Pengguna media sosial aktif mencapai 2,206 miliar, sedangkan pengguna *mobile* mencapai 3,734 miliar. Pertumbuhan yang paling signifikan ditunjukkan oleh pengguna yang mengakses media sosial melalui *platform mobile*. Pengguna jenis ini bertumbuh hingga 23,3 persen. Salah satu media sosial yang sering diakses adalah *twitter*.

Twitter merupakan aplikasi media sosial yang bersifat terbuka dibandingkan dengan aplikasi media sosial lainnya seperti *facebook* yang bersifat tertutup [2]. Berdasarkan [3] pengguna *twitter* mencapai 500 juta pengguna global. *Twitter* dimanfaatkan oleh aparat penegak hukum di Amerika Serikat untuk memberi informasi tambahan dalam melakukan penyelidikan. Bertambahnya *volume* pengguna *twitter* menyebabkan pertukaran informasi yang saling terhubung

menjadi sebuah peluang bagi bidang *social network analysis* untuk menangkap banyaknya informasi yang tersedia.

Social network analysis dapat dideskripsikan sebagai studi yang mempelajari tentang hubungan manusia dengan memanfaatkan teori *graph* [4]. *Graph* memiliki dua elemen utama yaitu *node* dan *edge*. *Node* merupakan representasi pengguna *twitter*, sedangkan *edge* merupakan representasi relasi antar pengguna *twitter*. *Graph* yang terbentuk akan dilakukan analisis dengan pengukuran nilai *normalized degree centrality*, *normalized closeness centrality*, dan *normalized betweeness centrality* dari masing-masing pengguna sehingga dapat diketahui pengguna media sosial yang berpengaruh. *Degree* digunakan untuk mengukur jumlah *edge* yang terhubung ke *node*. Pengguna dengan *normalized degree centrality* yang tinggi adalah pengguna yang memiliki kolaborasi paling banyak. Dengan menggunakan pengukuran tersebut, dapat di identifikasi pengguna yang paling aktif. *Normalized closeness centrality* didasarkan pada total jarak antar suatu *node* dengan *node* yang lain. Pengguna dianggap memiliki *normalized closeness centrality* tinggi jika memiliki kedekatan dengan pengguna yang lainnya. *Normalized betweeness centrality* disebut dengan penghubung, karena memiliki kemampuan untuk menyalurkan informasi dari satu pengguna kepada pengguna lainnya.

Berdasarkan latar belakang tersebut, pada Tugas Akhir ini diperoleh penelitian untuk mengukur kekuatan, pengaruh, atau karakteristik individu lain dari seseorang (berdasarkan pola hubungan). *Social network analysis* adalah sebuah cara untuk memahami perubahan sosial. Metode penelitian ini digunakan untuk identifikasi pengguna media sosial yang berpengaruh berdasarkan hasil ketiga *centrality* pengukuran dari *graph* yang terbentuk oleh *social network analysis.*

1.2 Rumusan Masalah

Berdasarkan latar belakang tersebut, dapat dirumuskan permasalahan dalam Tugas Akhir ini adalah sebagai berikut :

1. Bagaimana membentuk *graph* relasi pengguna *twitter* yang terlibat dalam pembahasan topik tertentu.
1. Bagaimana mengidentifikasi pengguna *twitter* yang paling aktif, pengguna *twitter* yang mempunyai kedekatan dengan pengguna yang lain, dan pengguna *twitter* yang bertindak sebagai penghubung berdasarkan analisis *graph* dengan metode *social analysis network*.

1.3 Batasan Masalah

Pada penelitian ini, penulis membuat batasan masalah sebagai berikut :

2. Media sosial yang digunakan adalah *twitter* yang terlibat dalam pembahasan topik tertentu.
3. Data yang digunakan sejumlah 17140 data *twitter* yang terlibat dalam pembahasan topik tertentu yang diambil pada bulan Januari 2015 sampai dengan bulan Mei 2016 pada situs Kaggle.
4. Hasil *social network analysis* diambil nama pengguna *twitter* dengan hasil pengukuran *centrality*.
5. Aplikasi yang digunakan adalah bahasa pemrograman Java.
6. Software basis data yang digunakan adalah MySQL

1.4 Tujuan

Berdasarkan permasalahan yang telah dirumuskan sebelumnya, tujuan penelitian Tugas Akhir ini adalah :

1. Membentuk representasi *graph* relasi pengguna *twitter* yang terlibat dalam pembahasan topik tertentu.
2. Menganalisa hasil dari representasi *graph* yang terbentuk menggunakan *social network analysis* untuk mengidentifikasi pengguna *twitter* yang berpengaruh berdasarkan topik tertentu.

1.5 Manfaat

Manfaat dari Tugas Akhir ini adalah sebagai informasi mengenai pengguna yang berpengaruh pada media sosial berdasarkan data *twitter* yang terlibat dalam pembahasan topik tertentu.

1.6 Sistematika Penulisan Tugas Akhir

Sistematika dari penulisan Tugas Akhir ini adalah sebagai berikut :

1. BAB I PENDAHULUAN
   Bab ini menjelaskan tentang gambaran umum dari penulisan Tugas Akhir ini yang meliputi latar belakang masalah, perumusan masalah, batasan masalah, tujuan, manfaat penelitian, dan sistematika penulisan.
2. BAB II TINJAUAN PUSTAKA
   Bab ini berisi tentang materi-materi yang mendukung Tugas Akhir ini, antara lain penelitian terdahulu, media sosial, Metode *Rapid Automatic Keyphrase Extraction* (RAKE) dan *Social Network Analysis*.
3. BAB III METODOLOGI PENELITIAN
   Bab ini menjelaskan metode penelitian yang dilakukan dalam menyelesaikan Tugas Akhir.
4. BAB IV PERANCANGAN DAN IMPLEMENTASI SISTEM
   Bab ini membahas analisa dan perancangan sistem perangkat lunak yang akan dibangun untuk identifikasi pengguna media sosial yang berpengaruh berdasarkan *graph* dengan metode *social network analysis*.
5. BAB V HASIL DAN PEMBAHASAN
   Bab ini menjelaskan mengenai hasil dari pembentukan *graph* relasi pengguna *twitter*. Setelah itu, dilakukan analisis terhadap *graph* yang terbentuk untuk mengidentifikasi pengguna *twitter* yang berpengaruh berdasarkan hasil pengukuran *degree centrality*,

closeness centrality dan *betweeness centrality* dari masing-masing pengguna.

6. BAB VI PENUTUP
   Bab ini berisi kesimpulan yang diperoleh dari pembahasan masalah sebelumnya serta saran yang diberikan untuk pengembangan selanjutnya.

“Halaman ini sengaja dikosongkan”

BAB II
TINJAUAN PUSTAKA

Pada bab ini dibahas mengenai dasar teori yang digunakan dalam penyusunan Tugas Akhir ini. Dasar teori yang dijelaskan dibagi menjadi beberapa subbab yaitu penelitian terdahulu, media sosial, Metode *Rapid Automatic Keyphrase Extraction* (RAKE) dan *Social Network Analysis*.

2.1 Penelitian Terdahulu

Pada penelitian sebelumnya, Rio Oktora dan Andry Alamsyah [5] melakukan penelitian mengenai pola interaksi dan aktor yang paling berperan pada event JGTC 2013 melalui media sosial *twitter* (studi menggunakan metode *social network analysis).* Sampel dari penelitian ini adalah *tweet* yang berupa interaksi (terdapat mention, baik berupa *reply* maupun *quote retweet*) yang memuat kata 'JGTC' dan '#JGTC36' pada 1 Desember 2013. Hasil penelitian pada *event* JGTC 2013 terdapat 7624 *node* (akun) yang terlibat dengan 7445 *edge* (interaksi) yang terjadi di *network* tersebut. Aktor (*node*) yang paling berpengaruh dalam *network* JGTC 2013 secara keseluruhan adalah raisa6690 yang merupakan bintang tamu pengisi acara pada *event* JGTC 2013.

Rizal Himmawan Sucipto [6] melakukan penelitian mengenai analisis jaringan teks berdasarkan *social network analysis* dan *text mining* untuk mengetahui persepsi kualitas merek pada konten percakapan di media sosial *twitter* dengan menggunakan studi kasus pada PT. Indosat dan PT. Telkomsel. Data yang digunakan dengan melakukan *crawling twitter* selama 7 hari dan didapatkan data *tweet* percakapan dengan kata kunci "indosat" sebanyak 14.253 dan kata kunci "telkomsel" sebanyak 46.911. Metode yang digunakan adalah metode *association rule* dan *community detection* untuk menemukan asosiasi kata-kata dan kelompok kata untuk dianalisis agar mendapatkan persepsi kualitas masing-masing

merek. Hasil analisis berupa persepsi kualitas, ditunjukkan dari hubungan kata-kata dominan dalam *graph* sesuai dengan *Branding Mention Merek* "Indosat" dan "Telkomsel". Dalam jaringan kualitas merek Indosat terdapat 107 *node* dan 264 *edges* dan pada jaringan kualitas merek Telkomsel terdapat 65 *node* dan 1133 *edges*.

Qunwei Li, dkk [7] melakukan penelitian mengenai prediksi node yang berpengaruh dengan menggunakan model *Multi-task Gausian Copula*. Data yang digunakan adalah data *tweets* dari *twitter* pada situs Kaggle sebanyak 17410 *tweets* dan 112 *users* yang diambil pada Januari 2015 sampai dengan Mei 2016. Qunwei Li, dkk telah melakukan *praprocessing* data, ekstraksi kata kunci, dan deteksi topik. Metode yang digunakan untuk mendeteksi topik adalah metode *Non-negative Matrix Factorization (NMF)* dan dihasilkan 10 topik.

Alvida Mustika Rukmi, [8] melakukan penelitian mengenai pengklasteran peneliti berdasarkan data publikasi dosen menggunakan *Latent Semantic Analysis* dan *K-Means++*. Data yang digunakan adalah atribut peneliti yang diperoleh melalui data publikasi peneliti dan karakteristik peneliti pada jejaring sosial yang telah terbentuk dari relasi antar peneliti. Metode yang digunakan untuk ekstraksi kata kunci menggunakan metode *Rapid Automatic Keyphrase Extraction (RAKE)*, dan metode ekstraksi konsep menggunakan metode *Latent Semantic Analysis (LSA)*. Sedangkan metode yang digunakan untuk pengklasteran menggunakan metode *K-Means++*. Hasil penelitian berupa identifikasi komunitas peneliti. Para peneliti akan mengetahui relasi dengan peneliti lainnya berkenaan dengan kemiripan topik publikasi ilmiah dan disiplin ilmu berdasarkan identitas komunitas penelitiannya.

2.2 Media Sosial

Pada dasarnya media sosial merupakan perkembangan mutakhir dari teknologi-teknologi *web* baru berbasis internet, yang memudahkan semua orang untuk dapat berkomunikasi, berpartisipasi, saling berbagi dan membentuk sebuah jaringan secara *online*. Media sosial mempunyai banyak bentuk, diantaranya yang paling populer yaitu *microblogging* (*twitter*). *Twitter* adalah suatu situs web yang merupakan layanan dari *microblog*, yaitu suatu bentuk *blog* yang membatasi ukuran setiap *post*-nya, yang memberikan fasilitas bagi pengguna untuk dapat menuliskan pesan. *Twitter* merupakan salah satu jejaring sosial yang paling mudah digunakan, karena hanya memerlukan waktu yang singkat tetapi informasi yang disampaikan dapat langsung menyebar secara luas. *Twitter* pertama kali ada pada tanggal 21 Januari 2000 di San Fransisco, California. *Microblogging twitter* hanya mampu mengantar pesan pendek, dengan panjang maksimal 140 karakter untuk setiap pesan.

Twitter merupakan jejaring sosial besar yang fokus pada kecepatan komunikasi. Kecepatan dan kemudahan dalam hal publikasi pesan membuat *twitter* menjadi media komunikasi yang penting. [9] mengatakan bahwa terdapat sekitar 140 juta pengguna aktif yang membuat lebih dari 400 juta pesan setiap hari.

Twitter dapat digunakan oleh pengguna untuk mempublikasikan pesan ("*tweeting*") dengan sangat cepat dan mudah. Pengguna dapat terhubung dengan pengguna lain melalui fitur '*follow*', sehingga pengguna dapat mengikuti *tweet* terbaru dari pengguna yang di *follow*. Perlu diperhatikan bahwa mekanisme follow tidak mewajibkan pengguna lain untuk melakukan *follow* balik. Istilah lain yang cukup populer diantaranya adalah RT atau *retweet*, dengan simbol '@' yang diikuti nama pengguna. *Retweet* merupakan sarana membalas *tweet* dengan menyertakan isi *tweet* sumber, sehingga pengguna yang menerima *retweet* bisa memahami

konteks pesan yang diterima. Selain itu terdapat simbol '#' yang diikuti sebuah kata yang merepresentasikan sebuah *hastag*. Opsi ini penting untuk menandai konteks dari sebuah pesan *twitter*, namun *hastag* bukanlah syarat untuk publikasi *tweet*.

Terdapat beberapa alasan pesan *twitter* digunakan sebagai sumber penelitian ini [10]:

1. Frekuensi posting pesan yang sangat tinggi.
2. Pesan *twitter* tidak terlalu panjang hanya 140 karakter, sehingga lebih deskriptif dan mudah dimengerti
3. *Twitter* menyediakan *semi-structured meta-data* (kota, negara, jenis kelamin, dan umur).

2.3 Metode *Rapid Automatic Keyphrase Extraction* (RAKE)

Metode *Rapid Automatic Keyphrase Extraction* adalah metode yang menggunakan pendekatan berbasis dokumen individu yang mampu mengelompokan tanpa bergantung pada koleksi dokumen. Metode RAKE memperhatikan asosiasi kata dengan menghitung matriks kemunculan bersama satu dengan yang lain. Matriks tersebut digunakan untuk mengukur skor kandidat kata kunci untuk kemudian dilakukan perangkingan.[11]

Metode RAKE memiliki tahapan sebagai berikut [12]:

1. Ekstraksi Kandidat
   Ekstraksi kandidat kata kunci dimulai dengan memisahkan teks menggunakan *stopword* dan tanda baca. Misalkan terdapat d_i adalah dokumen ke-i dari data *tweet* yang ada. Setelah *tweet* diekstraksi kata kuncinya, maka didapatkan kandidat kata kunci $Td_i = \{t_1, t_2, t_3, ...,t_n\}$, dengan $t_1, t_2, t_3, ...,t_n$ dalam bentuk kata, maupun frase kandidat kata kunci.
2. Menghitung matriks *co-ocurrence*
   Setelah kandidat kata kunci didapatkan, langkah selanjutnya adalah menghitung matriks *co-ocurrence*.

Matriks *co-ocurrence* memetakan frekuensi kemunculan suatu kata dan frase kata kunci

3. Menghitung rasio
   Nilai rasio merupakan perbandingan antara derajat kata dengan frekuensi kata. Derajat kata adalah jumlah kemunculan kata pada dokumen ditambah jumlah frase yang mengandung kata tersebut. Derajat kata pada matriks *co-ocurrence* didapat dari penjumlahan satu kolom atau satu baris. Frekuensi kata adalah jumlah kemunculan kata dalam teks. Nilai frekuensi bisa didapatkan pada nilai diagonal pada matriks *co-ocurrence*. Rasio dapat dihitung dengan menggunakan Persamaan :

$$rasio = \frac{deg(w)}{freq(w)} \quad (2.1)$$

   dimana : $deg(w)$ = derajat kata
   freq(w) = frekuensi kata.

4. Menghitung nilai fitur dasar
   Nilai fitur dasar merupakan nilai penjumlahan rasio kata yang ada pada kandidat kata kunci. Nilai tersebut kemudian diurutkan berdasarkan nilai tertinggi sampai terendah. Nilai fitur dasar dihitung dengan menggunakan Persamaan 2.

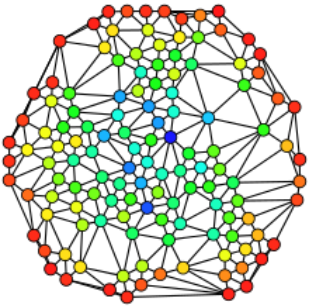
$$Nfd = rasio\,(t) + rasio\,(ft) \quad (2.2)$$

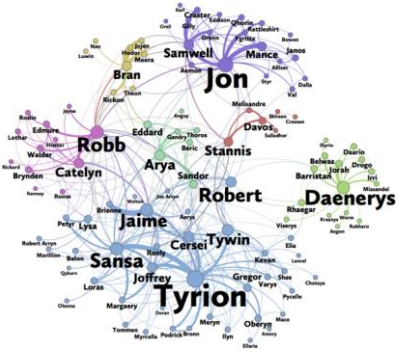
   dimana : *Nfd* = nilai fitur dasar,
   rasio (t) = rasio kata kunci t
   rasio (ft) = rasio frase yang mengandung kata kunci t.

5. Pemilihan kandidat frase kunci dengan skor tertinggi.
   Setelah kandidat kata kunci diberi nilai, selanjutnya dipilih sejumlah k

2.4 *Social Network Analysis*

Analisis jejaring sosial merupakan salah satu representasi teori *graph*. Implementasi dengan jejaring sosial mampu memeriksa struktur dari hubungan sosial dalam suatu kelompok serta mengungkapkan hubungan informal antar individu. Teori Jejaring Sosial memandang hubungan sosial dalam kaitan dengan *node* dan *ties*. Node adalah para aktor yang individu di dalam jejaring, dan *ties* adalah hubungan antar aktor. Dalam format yang paling sederhana, suatu jejaring sosial adalah suatu peta dari semua relevan *ties* antar *nodes* yang sedang dipelajari.





Gambar 2. 1 Visualisasi *graph social network analysis*

Terdapat tiga pengukuran sentralitas secara umum, yaitu [13]:

2.4.1 *Degree Centrality*

Nilai *degree centrality* suatu aktor merupakan banyaknya relasi langsung yang dimiliki oleh aktor tersebut. Semakin tinggi nilai *degree centrality* suatu aktor, semakin sentral posisinya dalam keseluruhan jaringan. Aktor yang menjadi pusat dalam jaringan biasanya mudah mempengaruhi dan dipengaruhi oleh individu-individu di sekitarnya [14].

Secara formal, *degree centrality* suatu aktor $\left(d(n_i)\right)$ di dalam *sociometrix X* dinyatakan pada Persamaan 2.3.

$$d(n_i) = \sum_{j=1}^{N} x_{ij} = \sum_{j=1}^{N} x_{ji} \quad (2.3)$$

dimana,

$d(n_i)$ = *degree* dari aktor (*node*) *i*

N = jumlah aktor

x_{ij} = isi sel *sociometrix x* baris ke *i* kolom ke *j; sociometrix* tersebut merupakan matriks boolean dengan nilai 0 menyatakan tidak adanya relasi dan nilai 1 menyatakan adanya relasi

Persamaan 2.3 harus dinormalisasikan dengan cara membandingkan Persamaan 2.3 dari setiap aktor dengan maksimum jumlah relasi yang mungkin dimiliki oleh seorang aktor. Jika banyak aktor dalam komunitas adalah *N,* maka maksimum jumlah relasi yang mungkin dimiliki oleh seorang aktor adalah *N-1*. Dengan demikian, nilai *normalized degree centrality* akan berkisar antara 0 dan 1. *Normalized degree centrality* dapat dinyatakan pada Persamaan 2.4.

$$C_D(n_i) = \frac{d(ni)}{N-1} \quad (2.4)$$

dimana,

$C_D(n_i)$ = *normalized degree centrality* aktor (*node*) *i*

$d(n_i)$ = *degree* dari aktor (*node*) *i*

N = jumlah aktor

2.4.2 *Closeness Centrality*

Closeness centrality bertujuan untuk mengetahui seberapa dekat seorang aktor dengan semua aktor lainnya dalam jaringan. Nilai *closeness centrality* suatu aktor merupakan total *geodesic distance* yang menghubungkan aktor tersebut dengan setiap aktor dalam jaringan. Aktor yang

memiliki nilai *closeness* yang tinggi merupakan aktor yang paling cepat mengetahui informasi atau isu yang sedang berkembang di komunitasnya. Selain itu, aktor yang memiliki nilai *closeness centrality* yang tinggi umunya memegang perananan sebagai pemimpin komunitas [14]. *Closeness* merupakan *invers* dari *degree centrality* sehingga secara formal, *closeness* suatu aktor $\left(c_c(n_i)\right)$ dalam *sociomatrix* dinyatakan pada Persamaan 2.5.

$$c_c(n_i) = \left[\sum_{j=1}^{N} d\left(n_i, n_j\right)\right]^{-1} \quad (2.5)$$

dimana,

$c_c(n_i)$ = *closeness centrality* aktor (*node*) *i*
$d\left(n_i, n_j\right)$ = jarak aktor *i* ke *j*
n_i = aktor (*node*) *i*, $i \neq j$
N = jumlah aktor

Normalisasi terhadap Persamaan 2.5 dilakukan dengan mengalikan Persamaan 2.5 suatu aktor *n* dengan jumlah maksimal relasi sosial yang mungkin dimiliki oleh aktor *n* tersebut. Jika banyak aktor dalam komunitas adalah *N*, maka jumlah maksimal relasi sosial yang mungkin dimiliki oleh aktor *n* adalah *N-1*. Dengan demikian, nilai *normalized closeness centrality* akan berkisar antara 0 dan 1. *Normalized Closeness centrality* dapat dinyatakan pada Persamaan 2.6.

$$C_c(n_i) = \left(c_c(n_i)\right)(N-1) \quad (2.6)$$

dimana,

$C_c(n_i)$ = *normalized centrality closeness* aktor (*node*) *i*
$c_c(n_i)$ = *closeness centrality* aktor (*node*) *i*
N = jumlah aktor

2.4.3 *Betweeness Centrality*

Betweeness centrality bertujuan untuk mengetahui seberapa besar kemungkinan seorang aktor menjadi perantara dalam hubungan setiap pasangan aktor dalam jaringan. Nilai *betweeness centrality* suatu aktor merupakan banyaknya kehadiran aktor tersebut dalam lintasan terpendek setiap pasangan aktor tersebut dalam jaringan. Berbeda dengan *degree centrality*, *betweeness centrality* merujuk pada banyaknya relasi tidak langsung yang dimiliki oleh sebuah aktor. Aktor dengan nilai *betweeness centrality* tertinggi merupakan aktor yang sering bertindak sebagai perantara dalam jaringan. Secara formal, *betweeness centrality* suatu aktor $c_b(n_i)$ dinyatakan pada Persamaan 2.7 [15].

$$c_b(n_i) = \sum_{j=1}^{n} \sum_{k=1}^{j-1} \frac{g_{jk}(ni)}{g_{jk}} \tag{2.7}$$

dimana,

$c_b(n_i)$ = *betweeness centrality* aktor (*node*) *i*

$g_{jk}(n_i)$ = jumlah *geodesic* dari aktor *j* ke aktor *k* yang mengandung aktor *i*

g_{jk} = jumlah *geodesic* dari aktor *j* ke aktor *k*

n_i = aktor (*node*) *i*, $i \neq j$, $i \neq k$

Normalisasi terhadap Persamaan 2.7 dilakukan dengan cara membandingkan Persamaan 2.7 suatu aktor *n* dengan jumlah pasangan aktor dalam komunitas tanpa kehadiran aktor *n* tersebut. jika banyak aktor dalam komunitas adalah *N*, maka banyaknya pasangan aktor yang mungkin tanpa kehadiran aktor *n* adalah $C_{(N-1,2)}$. Dengan demikian, nilai *normalized betweeness centrality* akan berkisar antara 0 dan 1. *Normalized betweeness centrality* dapat dinyatakan pada persamaan 2.8.

$$C_B(n_i) = \frac{c_b(n_i)}{\left[\frac{(N-1)(N-2)}{2}\right]} \quad (2.8)$$

dimana,

$C_B(n_i)$ = *normalized betweeness centrality* aktor (*node*) *i*

N = jumlah aktor

$c_b(n_i)$ = *betweeness centrality* aktor (*node*) *i*

BAB III
METODE PENELITIAN

Pada bab ini dijelaskan langkah-langkah yang digunakan dalam penyusunan Tugas Akhir. Disamping itu, dijelaskan pula prosedur dan proses pelaksanaan tiap-tiap langkah yang dilakukan dalam menyelesaikan Tugas Akhir.

3.1 Studi Literatur

Pada tahap ini dilakukan berbagai pengumpulan informasi pendukung yang dapat menunjang pengerjaan Tugas Akhir meliputi:

a. Pengolahan *Text Mining* antara lain tahap preproses, ekstraksi kata kunci suatu dokumen menggunakan metode *Rapid Automatic Keyphrase Extraxction* (RAKE).
b. Metode *Social Network Analysis* yang digunakan untuk mengidentifikasi profil pengguna *twitter*.

3.2 Pengumpulan Data

Pada tahap ini dilakukan pengumpulan data yang diperoleh dari situs Kaggle sejumlah 17140 data *tweet* yang diambil pada bulan Januari 2015 sampai dengan bulan Mei 2016. Atribut yang terdapat dalam data *tweet* tersebut meliputi id data, *tweet*, nama pengguna dan *username*.

3.3 Perancangan Perangkat Lunak

Pada tahap ini, perancangan perangkat lunak meliputi beberapa perancangan utama yaitu perancangan data, perancangan proses, dan perancangan antarmuka aplikasi.

a. Tahap perancangan database meliputi perancangan data masukan. Data masukan aplikasi ini adalah data *twitter* yang terlibat dalam pembahasan topik tertentu. Data-data tersebut telah tersimpan di dalam database. Data lain yang digunakan adalah daftar *stopword*. *Stopword* merupakan kata-kata yang dianggap tidak penting dalam

ekstraksi kata kunci. *Stopword* tersebut juga telah tersimpan di database. Selain itu data yang digunakan adalah data topik yang telah didapatkan dari penelitian sebelumnya.

b. Tahap perancangan proses merupakan tahap perancangan fungsi-fungsi dan algoritma yang terdapat dalam aplikasi. Aplikasi dibangun menggunakan bahasa pemrograman Java. Secara garis besar terdapat 4 proses utama yaitu preproses, ekstraksi kata kunci, penarikan pengguna ke dalam *term* topik serta pembentukan *graph sosial network analysis.*
c. Tahap perancangan antarmuka aplikasi menggunakan GUI pada Java.

3.4 Pembuatan Perangkat Lunak

Pada tahap pembuatan perangkat lunak, rancangan data, proses, dan antarmuka diimplementasikan dengan bahasa pemrograman Java. Aplikasi utama dibangun menggunakan Netbeans IDE 7.4 untuk mempermudah proses coding dan perancangan antarmuka. Selain itu, pilihan software basisdata yang digunakan adalah MySQL karena waktu pemrosesannya cepat. Berikut proses implementasi program meliputi :

a. Proses *Text Mining*

Proses ini terdiri dari tiga tahapan yaitu ekstraksi kata kunci menggunakan metode RAKE dan penarikan pengguna *twitter* ke dalam *term* topik yang terbentuk pada penelitian sebelumnya [9].

b. Pembentukan *Graph Sosial Network Analysis*

Pada proses ini akan dilakukan pembentukan *graph* dari hasil penarikan pengguna *twitter* ke dalam *term* topik. Kemudian melakukan analisis terhadap *graph* yang terbentuk untuk menghitung *centrality* yang meliputi *degree centrality*, *closeness centrality* dan *betweeness centrality.*

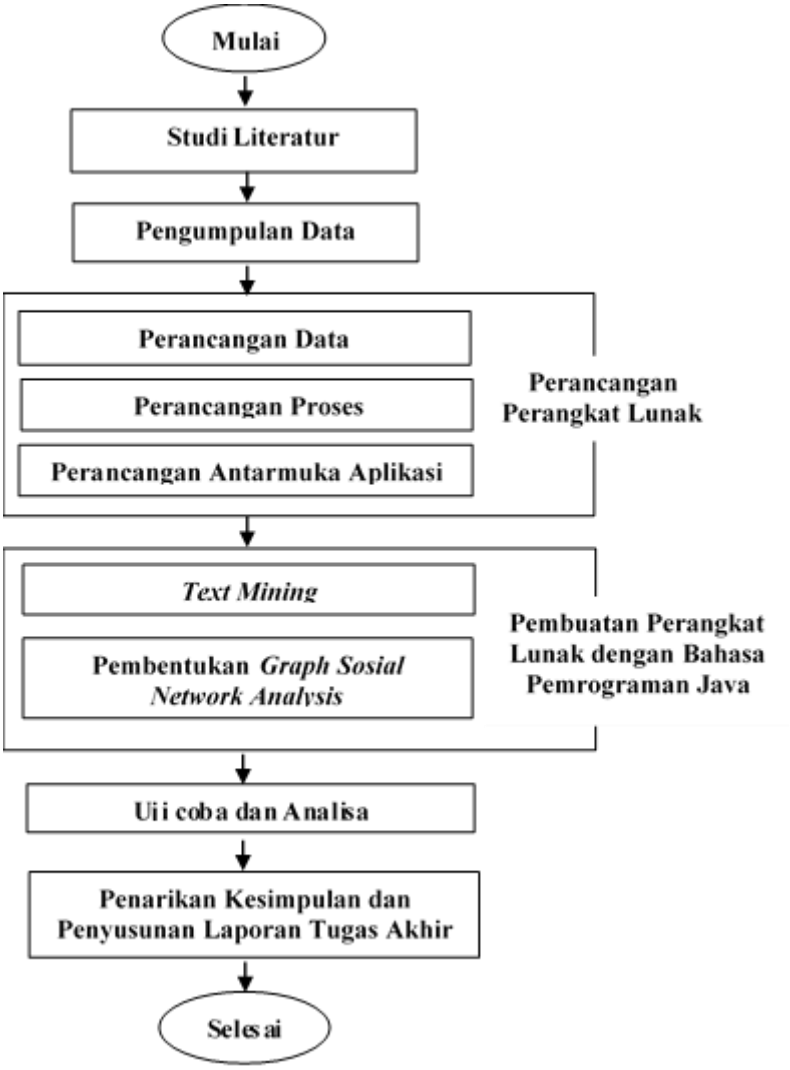
3.5 Uji Coba dan Analisa

Pada tahap ini, dilakukan uji coba sistem terhadap pembentukkan *graph social network analysis* merupakan visualisasi relasi yang mengambarkan sekumpulan *node* yang terkumpul dalam bentuk *graph*. Dimana *graph* jejaring sosial *twitter* yang terbentuk dari hasil penarikan pengguna ke dalam *term* topik yang telah terbentuk.

3.6 Penarikan Kesimpulan dan Penyusunan Laporan Tugas Akhir

Pada tahap akhir penelitian ini dilakukan penarikan kesimpulan dan penyusunan laporan Tugas Akhir dari identifikasi pengguna *twitter* yang berpengaruh berdasarkan *graph* dengan menggunakan *social network analysis.*

Berikut adalah diagram alir pengerjaan Tugas Akhir ini yang ditunjukkan pada Gambar 3:



Gambar 3. 1 Diagram alir pengerjaan Tugas Akhir

BAB IV
PERANCANGAN DAN IMPLEMENTASI SISTEM

Bab ini membahas analisa dan perancangan sistem perangkat lunak yang akan dibangun untuk identifikasi pengguna media sosial yang berpengaruh berdasarkan *graph* dengan metode *social network analysis*. Analisa sistem terdiri dari analisa kebutuhan sistem berupa deskripsi umum perangkat lunak dan lingkungan perancangan perangkat lunak.

Pada perancangan sistem akan membahas tentang sistem dari perangkat lunak yang akan dibangun, perancangan data yang meliputi data masukan dan data keluaran, serta perancangan algoritma proses-proses yang ada dalam tugas akhir ini. Selanjutnya membahas implementasi sistem yang meliputi hasil implementasi antarmuka dan keseluruhan proses di dalam sistem.

4.1 Analisis Sistem

4.1.1 Deskripsi Umum Perangkat Lunak

Perangkat lunak ini mampu melakukan ekstraksi kata kunci menggunakan data *twitter* dengan *Rapid Automatic Keyphrase Extraction* (RAKE) dan penarikan pengguna ke dalam *term* topik yang telah terbentuk. Untuk memenuhi kebutuhan tersebut maka perangkat lunak ini dibagi menjadi dua bagian utama:

1. Fitur proses awal terdiri dari dua bagian yaitu preproses dan ekstraksi kata kunci. Preproses data berfungsi untuk menghilamgkan karakter angka, menghilangkan tanda baca dan juga menghilangkan *stopword.* Sedangkan ekstraksi kata kunci berfungsi untuk mengekstraksi kata kunci pada seluruh data yang ada. Proses yang dilakukan pada fitur ini adalah preproses data menggunkan *case folding* dan *split* berdasarkan *stopword,* serta ekstraksi kata kunci menggunakan metode RAKE.

2. Fitur *social network analysis* terdiri dari penarikan pengguna ke dalam *term* topik berfungsi untuk menarik pengguna yang menulis *tweet* tertentu ke dalam *term* topik yang sudah dibentuk oleh penelitian sebelumnya. Kemudian dibentuk *graph* berdasarkan relasi yang ada antara pengguna satu dan pengguna yang lainnya. Visualisasi *graph* menggunakan library JUNG.

4.1.2 Lingkungan Perancangan Perangkat Lunak

Dalam proses perancangan perangkat lunak project ini, dibutuhkan lingkungan perancangan yang sesuai. Spesifikasi perangkat keras dan perangkat lunak yang digunakan dalam pengerjaan Project dijelaskan sebagai berikut:

Perangkat Keras:
1. Processor: AMD A8 – 6410 APU CPU@ 2,0 GHz
2. Memori (RAM) : 6,00 GB

Perangkat Lunak:
1. Sistem Operasi: Microsoft Windows 7
2. Perangkat Pengembang : Netbeans IDE 8,2, XAMPP versi 3.2.1

4.2 Perancangan Data

Data-data yang digunakan dalam perangkat lunak ini dapat dibedakan menjadi dua jenis, yaitu data masukan, dan data keluaran.

4.2.1 Data Masukan

Data masukkan adalah data-data yang digunakan sebagai masukan dari aplikasi. Masukan-masukan ini yang kemudian diolah oleh aplikasi melalui tahap-tahap tertentu sehingga menghasilkan keluaran yang diinginkan. Data masukan yang digunakan pada aplikasi ini adalah:

1. Data pengguna *twitter* yang terlibat dalam pembahasan topik tertentu yang telah disimpan di dalam *database*.
2. Data topik yang dihasilkan oleh penelitian sebelumnya.
3. Data kumpulan *stopword* yang telah tersimpan di dalam *database*.

Databse pengguna *twitter* memiliki beberapa atribut yaitu:

1. Id_data merupakan *id twitter*
2. *Tweet* merupakan status pengguna
3. Nama merupakan nama pengguna

4.2.2 Data Keluaran

Data keluaran merupakan data yang dihasilkan oleh aplikasi setelah proses-proses tertentu selesai dilakukan. Terdapat beberapa data keluaran pada aplikasi ini, yaitu:

1. Data hasil ekstraksi kata kunci menggunakan metode *Rapid Automatic Keyphrase Extraction* (RAKE).
2. Data hasil penarikan pengguna ke dalam *term* topik
3. Data hasil Analisis Jaringan Sosial berupa pengguna dan nilai *degree centrality*, *closeness centrality*, dan *betweeness centrality.*

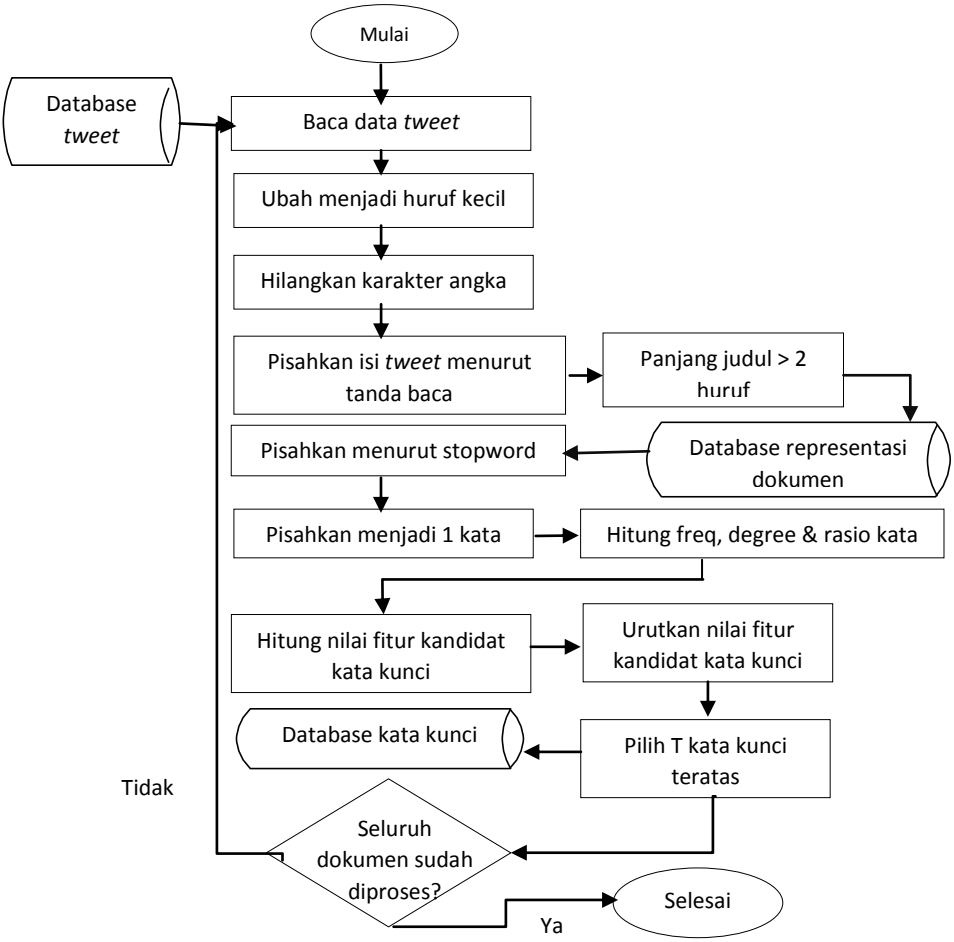
4.3 Perancangan Proses

Terdapat tiga proses utama yang dapat dilakukan oleh pengguna dalam perangkat lunak ini. Proses-proses tersebut antara lain:

4.3.1 Proses Awal

Proses awal dimulai dari ekstraksi kata kunci menggunakan metode *Rapid Automatic Keyphrase Extraction (RAKE)*. Kata kunci merupakan kata-kata yang dianggap penting apabila termasuk dalam peringkat 10 – 20 kata yang memiliki nilai terbesar dari sebuah data. Ekstraksi kata kunci ini memilki 5 tahapan utama, yaitu ekstraksi kandidat kata kunci, menghitung matriks *co-ocurrence*, menghitung nilai rasio, menghitung nilai fitur dasar dan memilih kata dengan nilai fitur tertinggi. Ekstraksi kandidat kata kunci sendiri meliputi 2 tahapan utama, yaitu *case folding* dan *split* berdasarkan *stopword. Case folding* adalah mengubah semua huruf menjadi huruf kecil, menghilangkan karakter angka dan menghapus tanda baca. Langkah-langkah ekstraksi kata kunci sebagai berikut:

- Input:
  - Data *Tweet*
- Output:
  - Kata kunci
- Langkah-langkah:
  1. Pengguna memasukkan kata kunci yang digunakan.
  2. Baca data *tweet* dari database
  3. Ubah seluruh huruf pada data *tweet* menjadi huruf kecil
  4. Hilangkan seluruh karakter angka
  5. Pisahkan isi *tweet* menurut tanda baca
  6. Pisahkan kata atau frase menurut *stopword*
  7. Simpan seluruh kata atau frase yang memiliki panjang lebih dari 2 huruf sebagai representasi dokumen
  8. Simpan seluruh kata dan frase sebagai kandidat kata.
  9. Pisahkan seluruh kandidat kata yang berupa frase menjadi satu kata
  10. Hitung frekuensi kemunculan setiap kata di dalam dokumen *freq(w)*
  11. Hitung nilai degree setiap kata *deg(w)*
  12. Hitung nilai rasio kata, *rasio(w) = deg(w)/freq(w)* setiap kata
  13. Hitung nilai fitur setiap kandidat kata dengan cara menambahkan nilai rasio tiap kata yang ada pada kandidat kata
  14. Urutkan nilai fitur kandidat kata dari kecil ke besar
  15. Simpan kata kunci sebanyak input jumlah kata yang dimasukkan oleh pengguna ke dalam database.
  16. Ulangi untuk setiap data *tweet* di dalam database

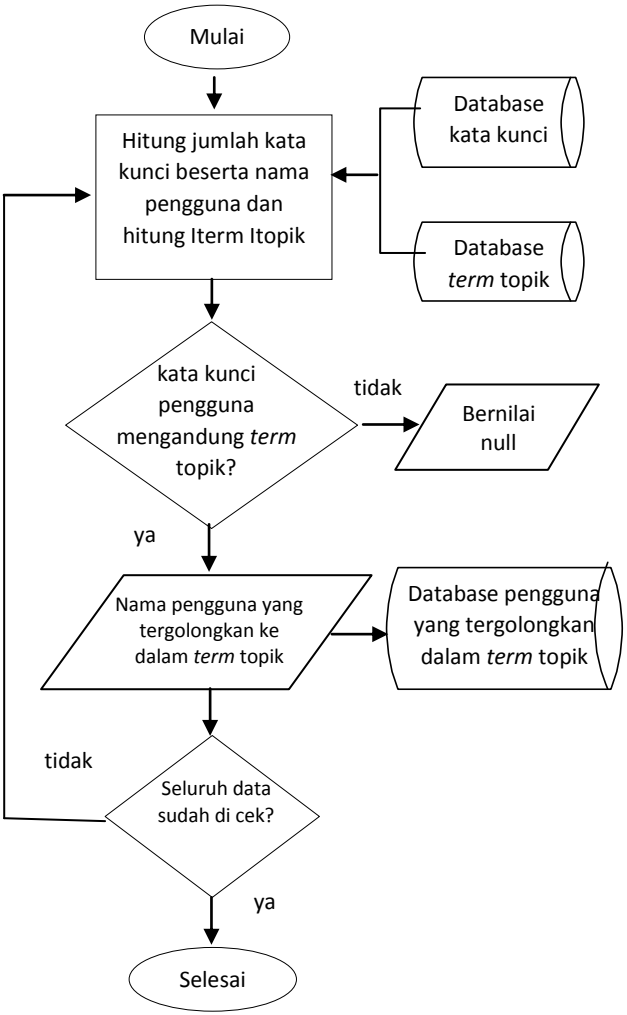


Gambar 4. 1 Ekstraksi kata kunci menggunakan RAKE

4.3.2 Penarikan Pengguna ke dalam Topik

Proses penarikan pengguna ke dalam *term* topik berfungsi untuk menarik pengguna yang menulis *tweet* tertentu ke dalam *term* topik yang sudah ada sebelumnya. Langkah-langkah penarikan pengguna ke dalam *term* topik adalah sebagai berikut:

- Input:
  - Kata kunci
  - *Term t*opik
- Output:
  - Pengguna yang sudah tergolongkan ke dalam *term* topik
- Langkah-langkah:
  1. Hitung kata kunci beserta nama pengguna yang telah tersimpan dalam *database*
  2. Hitung *term* topik dari *database*
  3. Kata kunci setiap pengguna akan dicek jika terdapat kata kunci yang mengandung *term* topik maka nama pengguna akan digolongkan kedalam *term* topik tersebut.
  4. Jika kata kunci tersebut tidak mengandung *term* topik maka bernilai *null.*
  5. Simpan semua *term* topik beserta nama pengguna yang sudah tergolongkan ke dalam *database.*
  6. Ulangi langkah 3, 4 dan 5 sebanyak panjangnya setiap konsep pengguna.

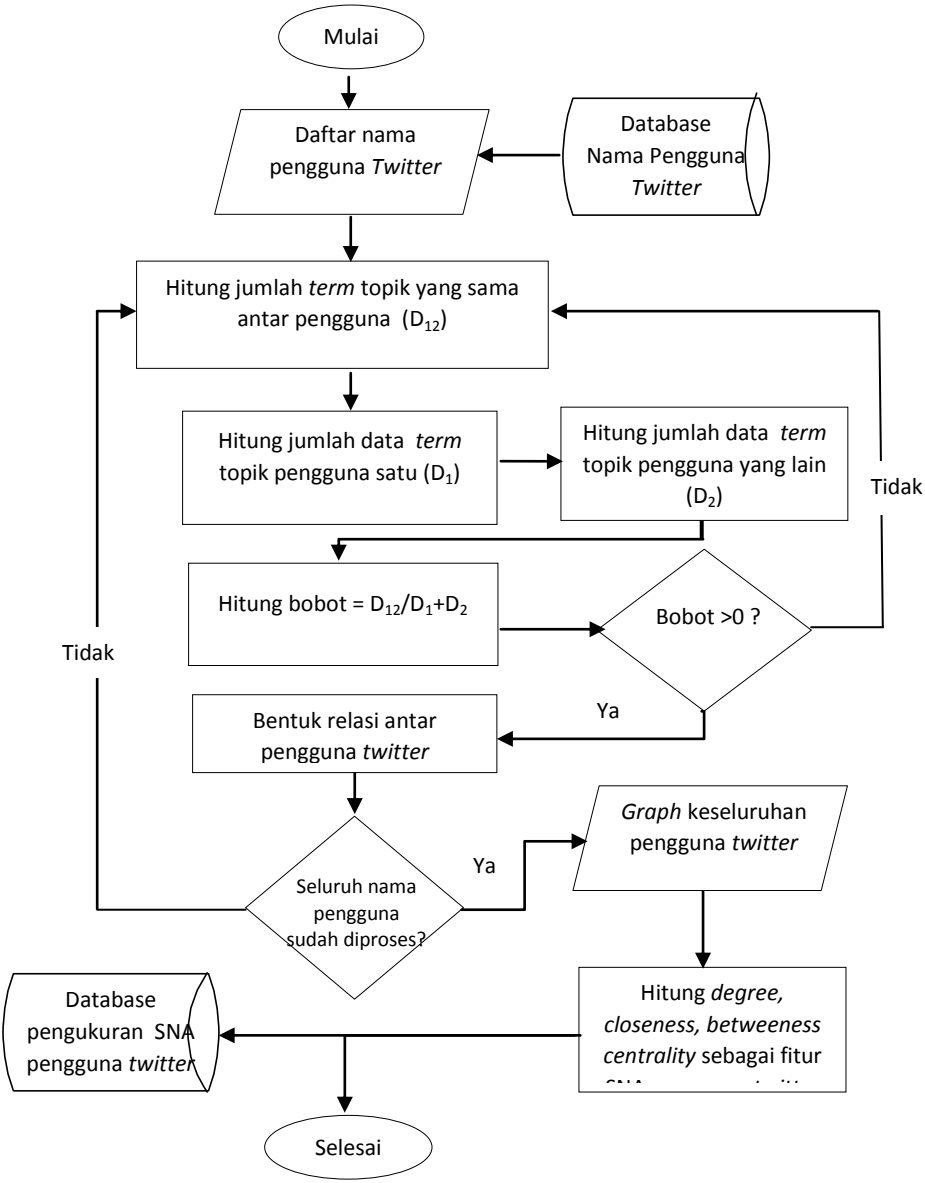


Gambar 4. 2 Penarikan nama pengguna ke dalam *term* topik

4.3.3 Pembentukan *Graph* Pengguna

Pembentukan *graph* berguna untuk merelasikan pegguna dengan pengguna yang lainnya. Jaringan sosial penguna terdiri dari *node* yang merepresentasikan pengguna serta e*dge* yang merepresentasikan hubungan antara pengguna. Hubungan antar pengguna didapatkan dari kesamaan *term* topik yang sudah dilakukan pada tahap penarikan pengguna ke dalam *term* topik. Berikut tahap pembentukan relasi awal pengguna:

- Input:
  - Pengguna yang sudah tergolongkan ke dalam *term* topik
- Ouput:
  - Fitur pengukuran *social network analysis* pengguna
- Langkah-langkah:
  1. Simpan seluruh nama pengguna yang ada pada *database*
  2. Hitung jumlah data *term* topik pengguna satu dengan pengguna yang lain
  3. Hitung jumlah data *term* topik pengguna tersebut
  4. Hitung jumlah data *term* topik pengguna yang lain
  5. Hitung bobot relasi dengan membagi jumlah *term* topik kedua pengguna dengan jumlah *term* topik pengguna pertama ditambah jumlah *term* topik pengguna kedua
  6. Jika nilai bobot lebih dari 0 maka bentuk relasi antar kedua pengguna tersebut
  7. Ulangi langkah 2, 3, 4, dan 5 untuk seluruh pengguna. Setelah didapatkan seluruh relasi pengguna maka akan terbentuk jaringan sosial pengguna berupa *graph.*
  8. Hitung nilai *degree centrality*, *closeness centrallity*, *dan betweeness centrality* setiap *node*, kemudian simpan di dalam *database*



Gambar 4. 3 Proses Pembentukan relasi keseluruhan

4.4 Implementasi Sistem

Setelah perancangan selesai, tahap selanjutnya adalah implementasi. Tahap ini bertujuan agar pengguna dapat menggunakan program yang telah dirancang.

4.4.1 Implementasi Antarmuka Aplikasi

Pada sub bab ini dijelaskan tentang kegunaan fungsi-fungsi yang ada di dalam aplikasi beserta tampilan desain. Antarmuka dibagi menjadi 3 tab, yaitu tab data *tweet*, proses awal dan *social network analysis*.

4.4.1.1 Tab Data *Tweet*

Tab ini hanya untuk memperlihatkan data *tweet* yang telah tersimpan pada *database*. Implementasi tab data *tweet* dapat dilihat pada Gambar 4.4.

Gambar 4. 4 Tab Data Tweet

4.4.1.2 Tab Proses Awal

Tab ini berguna untuk melakukan preproses data dan ekstraksi kata kunci. Pada tab ini terdapat tombol preproses, ekstraksi dan hasil preproses. Sebelum memulai ekstraksi, pengguna harus memasukkan jumlah kata kunci sebarang. Pada tab ini juga terdapat tabel yang digunkaan untuk menampilkan kata kunci yang berhasil diekstraksi dari data. Implementasi tab proses awal dapat dilihat pada Gambar 4.5

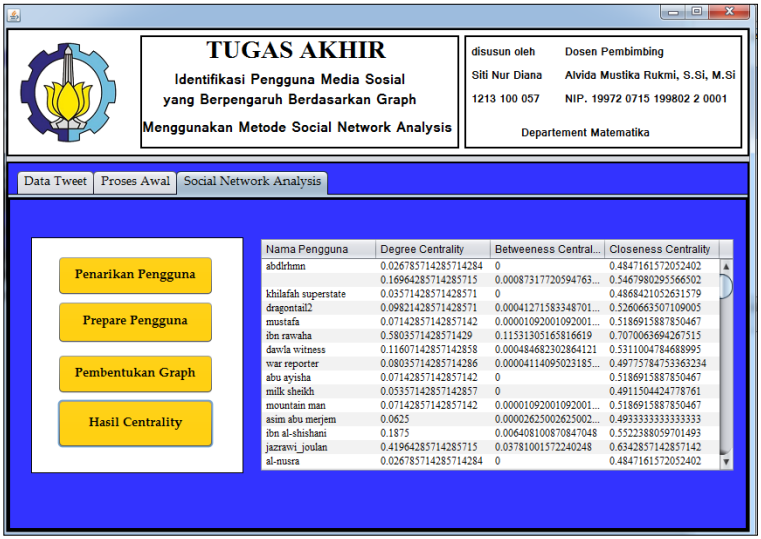
Gambar 4. 5 Tab Proses Awal

Tabel 4. 1 Tombol dan kegunaan paa tab proses awal

Tombol	Kegunaan
Preproses	Melakukan pemrosesan data sebelum digunakan meliputi *case folding* dan *split* berdasarkan *stopword*
Ekstraksi	Memulai proses ekstraksi kata kunci
Hasil Preproses	Data penggabungan kata kunci (*string* builder) berdasarkan nama pengguna

4.4.1.3 Tab *Social Network Analysis*

Tab berikutnya adalah tab *social network analysis.* Tab ini berguna untuk pembentukan *graph* pengguna *twitter*. Pada tab ini terdapat 4 tombol yaitu penarikan pengguna, *prepare* pengguna, pembentukan *graph* dan hasil *centrality.*



Gambar 4. 6 Tab *Social Network Analysis*

Tabel 4. 2 Tombol dan kegunaan pada tab Social Network Analysis

Tombol	Kegunaan
Penarikan Pengguna	Penarikan pengguna berdasarkan *term* topik
Prepare Pengguna	Membentuk vektor pengguna yang digunkan sebagai nilai pengukuran jarak antar pengguna
Pembentukan *Graph*	Memulai proses pembentukan relasi antara pengguna *twitter*

Hasil *Centrality*	Menampilkan hasil perhitungan *centrality* dan ditampilkan pada tabel

4.4.2 Implementasi Program

Perangkat lunak dalam Project ini dibuat menggunakan bahasa pemrograman Java pada platform Java Development Kit 7 dan Netbeans IDE 7.4 dengan library tambahan adalah JUNG. Database server yang digunakan adalah MySQL.

4.4.2.1 Implementasi Data *Tweet*

Implementasi data *tweet* merupakan tampilan antarmuka yang digunakan untuk menampilkan data dari *database*. Data-data tersebut yang akan digunakan sebagai inputan ke tahap selanjutnya.

```
private void UpdateTabel() {
        Database db = new Database();
        try {
         db.connectFirst();
         ResultSet rs = db.executeSelect("SELECT *
FROM data_tweets;");
          DefaultTableModel          dtm          =
(DefaultTableModel) tabeldata.getModel();
          dtm.setRowCount (0);
          String [] data = new String[4];
          int i = 1;
          while (rs.next()){
            data[0] = rs.getString ("id_data");
            data[1] = rs.getString ("tweets");
            data[2] = rs.getString ("nama");
            data[3] = rs.getString ("username");
            dtm.addRow (data);
            i++;
            }
          }
          catch (Exception e) {
JOptionPane.showMessageDialog(null,e);
     }
}
```

Gambar 4. 7 Implementasi data *tweet*

4.4.2.2 Implementasi Proses Awal

Proses awal terdiri dari preproses data menggunakan *case folding* dan *split* berdasarkan *stopword* serta ekstraksi kata kunci menggunakan RAKE. Proses ini dilakukan pada kelas RAKE.

4.4.2.2.1 Implementasi Tombol Preproses

Tombol preproses ini berfungsi untuk melakukan *case folding* dan *split* berdasarkan *stopword*. Metode *case folding* sendiri terdiri dari proses *lowercase* dan *split* berdasarkan tanda baca.

```
listKata = new ArrayList<>();
     listKataKunci = new ArrayList<>();
     Database db = new Database();
     db.connectFirst();
     String Dok = dokumen.getJudul();
     ArrayList<String>      tempKata      =      new
ArrayList<>();
     String hasil = prosesLowerCase(Dok);
     /*Split berdasarkan tanda baca*/
     ArrayList<String>         kandidatKK         =
prosesSplitTB(hasil);
     /*Split berdasarkan stopword*/
     kandidatKK = prosesSplitStopword(kandidatKK);
```

Gambar 4. 8 Implementasi preproses

Pada kode di atas terdapat beberapa *method* pada kelas RAKE yang digunakan yaitu:

1. prosesLowerCase()

   Method ini berguna untuk mengubah seluruh huruf menjadi huruf kecil.

```
Private String prosesLowerCase(String abstraksiDok)
{       abstraksiDok = abstraksiDok.toLowerCase();
            return abstraksiDok;   }
```

Gambar 4. 9 Implementasi proses *lowercase*

2. prosesSplitTB()

Method ini berguna untuk memisahkan *tweet* berdasarkan tanda baca.

```
private ArrayList<String> prosesSplitTB(String
abstraksiDok){

        ArrayList<String> hasil = new
ArrayList<String>();
        String URL1 = "((http|https)://|www[.])[-A-
Za-z0-9+&amp;@#/%?=~_()|!:,.;]*[-A-Za-z0-
9+&amp;@#/%=~_()|]";
        String URL2 =
"((www\\[\\s]+)|https?://[^\\s])";
        String LAIN = "@([^\\s]+)";
        String retweet = "(rt)";
        String URL3 = "(amp)";
        abstraksiDok =
abstraksiDok.replaceAll(URL1,"");
        abstraksiDok =
abstraksiDok.replaceAll(URL2,"");
        abstraksiDok =
abstraksiDok.replaceAll(LAIN,"");
        abstraksiDok =
abstraksiDok.replaceAll(retweet,"");
        abstraksiDok =
abstraksiDok.replaceAll(URL3,"");
        abstraksiDok = abstraksiDok.replaceAll("[0-
9]", "");
        abstraksiDok =
abstraksiDok.replaceAll("(\\r|\\n)","");

        String[] tempHasil =
abstraksiDok.split("\\p{Punct}");
        for(int i = 0; i < tempHasil.length; i++){
        if(tempHasil[i].length() > 2){
            StringBuilder sb = new StringBuilder();
            String[] buildString =
tempHasil[i].split(" ");
            for(int j = 0; j < buildString.length;
j++){
                if(buildString[j].length() > 1){
                    if(j < buildString.length-1){
                        sb.append(buildString[j]);
```

```
                        sb.append(" ");
                    }
                    else{
                        sb.append(buildString[j]);
                    }
                }
            }
            hasil.add(sb.toString());
        }

     }
        return hasil;
    }
```

Gambar 4. 10 Implementasi proses pemisahan berdasarkan tanda baca

3. prosesSplitStopword()

Method ini befungsi untuk memisahkan kandidat kata *term* menurut *stopword.*

```
private    ArrayList<String>    prosesSplitStopword
(ArrayList<String> splittedWord){

        ArrayList<String>  kandidatKataKunci  =  new
ArrayList<>();
        for(String s: splittedWord){
        ArrayList<String> hasil = isStopword(s);
            for(String st: hasil){
                st = st.replaceAll("(\\r|\\n)","");
                if(st.length() > 1){
                kandidatKataKunci.add(st);
            }
          }
        }
        return kandidatKataKunci;
    }
```

Gambar 4. 11 Implementasi proses pemisahan berdasarkan *stopword*

4.4.2.2.2 Implementasi Tombol Ekstraksi

Setelah pemisahan data *tweet* berdasarkan tanda baca dan *stopword* selesai dilakukan maka tahap berikutnya adalah menghitung nilai *frekuensi*, *degree*, dan *rasio* tiap kata. Tahap ini dimulai dengan memisahkan setiap kandidat kata kunci berdasarkan karakter spasi (“ “) sehingga menjadi hanya satu kata. Seluruh kata tersebut disimpan ke dalam variabel daftar kata. Nilai *frekuensi* dihitung berdasarkan kemunculan sebuah kata pada daftar kata. Nilai *degree* dihitung berdasarkan kemunculan sebuah kata pada kandidat kata kunci ditambah kemunculan kata tersebut pada daftar kata. Nilai *rasio* dihitung dengan cara membagi nilai *degree* dengan *frekuensi* kata tersebut.

```
/*Menghitung frekuensi kata*/
for(String s: tempKata){
    if(listKata.size() == 0){
        Kata katabaru = new Kata(s, 1, 0, 0);
        listKata.add(katabaru);
            }
    else{
        if(isAda(s)){
listKata.get(findIndex(s)).setFrek(listKata.get(fin
dIndex(s)).getFrek()+1);
                }
        else{
            Kata katabaru = new Kata(s, 1, 0, 0);
            listKata.add(katabaru);
            }
        }
}
/*Menghitung Degree*/
for(String s: kandidatKK){
    int jumlah = 0;
    String[] split = s.split(" ");
    jumlah = split.length;
        if(jumlah>1){
            for(Kata k: listKata){
```

```
                if(s.contains(k.getKata())){
                    k.setDeg(k.getDeg()+1);
                }
            }
        }
}

/*Menghitung Rasio*/
for(Kata k: listKata){
    k.setDeg(k.getDeg()+k.getFrek());
    k.setRasio(k.getDeg()/k.getFrek());
}
```

Gambar 4. 12 Implementasi tahap perhitungan *degree*, *frekuensi* dan *rasio*

Tahap berikutnya adalah mengukur nilai fitur kandidat kata kunci. nilai fitur kandidat kata kunci dihitung dengan menambahkan nilai rasio kata yang terdapat pada kandidat kata kunci tersebut. Nilai-nilai tersebut kemudian diurutkan dari besar ke kecil. Setelah itu diambil kata kunci dan disimpan di dalam *database*.

```
/*Menghitung Skor*/
    for(String s: kandidatKK){
        double skor = 0;
        for(Kata k:listKata){
            if(s.contains(k.getKata())){
                skor += k.getRasio() ;
            }
        }
        KataKunci kk = new KataKunci(s, skor);
        listKataKunci.add(kk);
    }
```

Gambar 4. 13 Implementasi proses perhitungan nilai fitur kandidat kata kunci

4.4.2.2.3 Implementasi Tombol Hasil Proses Awal

Tahap berikutnya adalah menggabungkan kata kunci berdasarkan nama pengguna dan menyimpan ke dalam *database.*

```
        while(dokIdRS.next()){
            StringBuilder sb = new StringBuilder();
            StringBuilder hasilprepos = new
StringBuilder();
            ArrayList<String> listRepDok = new
ArrayList<>();
            String idNama = dokIdRS.getString(1);
            ResultSet temp =
koneksi.executeSelect("SELECT `kata_kunci` FROM
`list_kata_kunci_rake` "
                    + "WHERE `nama` = \""+ idNama
+"\"" );
            while(temp.next()){

listRepDok.add(temp.getString("kata_kunci"));
            }
            for (int a=0; a<listRepDok.size();a++){
hasilprepos.append(listRepDok.get(a));
            hasilprepos.append(" ");
            }
            koneksi.executeUpdate("INSERT INTO
`hasil_preproses`(`teks`, `nama`) "
                                    + "VALUES
(\""+hasilprepos+"\",\""+idNama+"\")");
        }
      koneksi.destroyConnection();
    }
}
```

Gambar 4. 14 Implementasi tombol hasil proses awal

4.4.2.3 Implementasi *Social Network Analysis*

Pada tahap implementasi *social network analysis* berfungsi untuk membentuk *graph* pengguna dan menghitung *centrality.* Tahap ini terdiri dari beberapa tombol, yaitu tombol

penarikan pengguna, *prepare* pengguna, pembentukan *graph* dan hasil *centrality*.

4.4.2.3.1 Implementasi Tombol Penarikan Pengguna

Tombol penarikan pengguna berfungsi untuk melakukan proses penarikan pengguna ke dalam *term* topik. Dimana *term* topik sudah terbentuk oleh penelitian sebelumnya.

```
int count3[]=new int[daftarteks.length];
        int banyak=0;
        int max=0;
        for(int i=0;i<daftarteks.length;i++){
            for(int j=0;j<daftarstop.length;j++){
       if(daftarteks[i].contains(daftarstop[j]))
                    banyak++;
            }
            if(max<=banyak){
                max=banyak;
            }
            count3[i]=banyak;
            banyak=0;
        }
        System.out.println("Hasil deteksi topik");
System.out.println(Arrays.toString(count3));
        String word[][]=new
String[daftarteks.length][max];
        String posword[][]=new
String[daftarteks.length][max];
        int bn=0;
        int pos=0;
        for(int i=0;i<daftarteks.length;i++){
            bn=count3[i];
            for(int j=0;j<daftarstop.length;j++){
if(daftarteks[i].contains(daftarstop[j])){
                    word[i][pos]=daftarstop[j];
posword[i][pos]=daftarstopid[j];
                    pos++;
                    bn--;
                }
                if(bn==0){
                    break;
                }
            }
```

```
            pos=0;
        }
System.out.println(Arrays.deepToString(word));
System.out.println(Arrays.deepToString(posword)+"\n
");
        for (int i = 0; i < word.length; i++) {
        //System.out.println("points in cluster " +
(i + 1) + "\n");
        for (int j = 0; j < word[i].length; j++) {
        DokumenTA dta = new DokumenTA();
        LinkedList<DokumenTA> ldta =
dta.readPartToProses();
        LinkedList<String> user1 = new
LinkedList<>();
        LinkedList<String> listTopikMilikUser = new
LinkedList<>();
            for(DokumenTA ta : ldta){
if(user1.isEmpty()||!user1.contains(ta.getPengguna(
))){
      user1.add(ta.getPengguna());
      String temp=ta.getStatusPengguna();
      String[] tempsplit = temp.split(" ");
          for (String s : user1){
if(ta.getPengguna().equals(s)){
             for (String ss : tempsplit){
                if(word[i][j].equals(ss)){
    }
  } } } }
```

Gambar 4. 15 Implementasi proses penarikan pengguna ke dalam *term* topik

4.4.2.3.2 Implementasi Tombol *Prepare* Pengguna

Setelah dilakukan proses penarikan pengguna ke dalam *term* topik maka tahap berikutnya adalah mempersiapkan nama-nama pengguna tersebut sebagai bahan untuk membentuk *graph.*

```
public void prepareUsername() throws SQLException{
  Database koneksi = new Database();
  koneksi.connectFirst();
 koneksi.executeUpdate("DELETE FROM listpengguna");
```

```
        Pengguna pengguna = new Pengguna();
        DokumenTA dta = new DokumenTA();
    LinkedList<DokumenTA> ldta =
dta.readPartToInput();
    LinkedList<String> USERNAME = new
LinkedList<>();
        USERNAME = pengguna.listPenggunaInput();
        for(String s : USERNAME){
            LinkedList<String> listTAMilikPengguna
= new LinkedList<>();
            for(DokumenTA ta : ldta){
if(ta.getPengguna1().toLowerCase().equals(s)||
ta.getPengguna2().toLowerCase().equals(s)||
ta.getPengguna3().toLowerCase().equals(s)||
.
.dst
ta.getPengguna92().toLowerCase().equals(s)||
```

Gambar 4. 16 Implementasi proses *prepare* pengguna

4.4.2.3.3 Implementasi Tombol Pembentukan *Graph*

Tahap ini dilakukan untuk proses perhitungan bobot relasi setiap pengguna. Setelah didapatkan bobot relasi maka akan dibentuk *graph* dan perhitungan *centrality.*

```
LinkedList<DokumenTA> listDok =
dok.readPartToInput();
        LinkedList<String> listPengguna =
dosen.listPenggunaInput();
        int startidx = 0;
        Integer edgelbl = 0;
        for(String pengguna1 : listPengguna){
            pengguna1 = pengguna1.toLowerCase();
            for(int i = startidx; i <
listPengguna.size(); i++){
                String pengguna2 =
listPengguna.get(i).toLowerCase();
                Double d1uniond2 = 0.0;
                Double jumd1 = 0.0;
                Double jumd2 = 0.0;
                Double bobot = 0.0;
                if(!pengguna1.equals(pengguna2)){
```

```
                    for(DokumenTA dk : listDok){
                        if(
(dk.getPengguna1().toLowerCase().equals(pengguna1)&
&dk.getPengguna2().toLowerCase().equals(pengguna2))
||
.
.dst
(dk.getPengguna92().toLowerCase().equals(pengguna1)
&&dk.getPenggun91().toLowerCase().equals(pengguna2)
)||

if(dk.getPengguna1().toLowerCase().equals(pengguna1
)||
.
.dst
dk.getPengguna92().toLowerCase().equals(pengguna1)|
|
    jumd1 = jumd1 + 1;

if(dk.getPengguna1().toLowerCase().equals(pengguna2
)||
.
.dst
if(dk.getPengguna92().toLowerCase().equals(pengguna
2)||

    jumd2 = jumd2 + 1;
    bobot = d1uniond2 / (jumd1 + jumd2);
       if(d1uniond2 != 0.0){
    snaAddVertex(pengguna1);
snaAddVertex(pengguna2);
snaAddEdge(edgelbl.toString(), pengguna1,
pengguna2);
     edgelbl += 1;
}
```

Gambar 4. 17 Implementasi proses perhitungan bobot relasi dan pembentukan *graph*

Setelah perhitungan bobot relasi dan pembentukan *graph* maka akan dilakukan proses perhitungan *centrality* setiap pengguna. .

```
public void countDegreeCentrality() throws
SQLException{
        /*Menghitung degree centrality tiap
vertex*/
        System.out.println("Degree Centrality");
        Database koneksi = new Database();
        koneksi.connectFirst();
        koneksi.executeUpdate("DELETE FROM
`nodesnafitur`");
        int n = g.getVertexCount();
        DegreeScorer degCent = new DegreeScorer(g);
        double[] degValues = new double[n];
        int i = 0;
        for(String vertex : g.getVertices() ){
            degValues[i++] =
(double)degCent.getVertexScore(vertex)/(double)n;
            koneksi.executeUpdate("INSERT INTO
`nodesnafitur`(`id_node`, `degCent`, `betCent`,
`closeCent`) "
                    + "VALUES
(\""+vertex+"\","+((double)degCent.getVertexScore(v
ertex)/(double)n)+","+0.0+","+0.0+")");
        }
        koneksi.destroyConnection();

    }
```

Gambar 4. 18 Implementasi perhitungan *degree centrality*

```
 public void countClosenessCentrality() throws
SQLException{
        /*Menghitung closeness tiap node*/
        System.out.println("Closeness Centrality");
        Database koneksi = new Database();
        koneksi.connectFirst();
        int n = g.getVertexCount();
        System.out.println(n);
        double norm = n - 1;
        ClosenessCentrality closeCent = new
ClosenessCentrality(g);
        double[] closeValues = new double[n];


        int i = 0;
```

```
        for(String vertex : g.getVertices()){
            closeValues[i++] =
closeCent.getVertexScore(vertex);
            koneksi.executeUpdate("UPDATE
`nodesnafitur` "
                    + "SET
`closeCent`="+closeCent.getVertexScore(vertex)+" "
                    + "WHERE `id_node` =
\""+vertex+"\"");
        }
        koneksi.destroyConnection();
}
```

Gambar 4. 19 Implementasi perhitungan *closeness centrality*

```
    public void countBetweenessCentrality() throws
SQLException{
        /*Menghitung betweenes tiap node*/
        System.out.println("Betweeness
Centrality");
        Database koneksi = new Database();
        koneksi.connectFirst();
        BetweennessCentrality betweenCent = new
BetweennessCentrality(g);

betweenCent.setRemoveRankScoresOnFinalize(false);
        betweenCent.evaluate();
        double normVal = ((g.getVertexCount()-
1)*(g.getVertexCount()-2))/2;
        for(String vertex : g.getVertices()){
            koneksi.executeUpdate("UPDATE
`nodesnafitur` "
                    + "SET
`betCent`="+betweenCent.getVertexRankScore(vertex)/
normVal+" "
                    + "WHERE `id_node` =
\""+vertex+"\"");
//            System.out.println(vertex + " = " +
(betweenCent.getVertexRankScore(vertex)/normVal));
        }
        koneksi.destroyConnection();
    }
```

Gambar 4. 20 Implementasi perhitungan *betweeness centrality*

4.4.2.3.4 Implementasi Tombol Hasil *Centrality*

Proses ini hanya digunakan untuk menampilkan hasil *centrality* ke dalam tabel.

```
Database db = new Database();
        try {
            db.connectFirst();
            ResultSet rs = db.executeSelect("SELECT
* FROM nodesnafitur;");
            DefaultTableModel dtm =
(DefaultTableModel) tabelcentrality.getModel();
            dtm.setRowCount (0);
            String [] data = new String[4];
            int i = 1;
            while (rs.next()){
                data[0] = rs.getString ("id_node");
                data[1] = rs.getString ("degCent");
                data[2] = rs.getString ("betCent");
                data[3] = rs.getString
("closeCent");
                dtm.addRow (data);
                i++;
            }
        }
        catch (Exception e) {
            JOptionPane.showMessageDialog(null,e);
        }
```

Gambar 4. 21 Implementasi hasil *centrality*

BAB V
HASIL DAN PEMBAHASAN

Pada bab ini akan dilakukan pembahasan mengenai pengujian yang telah dilakukan perangkat lunak. Dalam hal ini menjelaskan mengenai proses pengujian yang dilakukan terhadap sistem *graph social network analysis*. Hasil pengujian kemudian dibahas untuk mengetahui kerja sistem secara keseluruhan dalam mengidentifikasi pengguna media sosial yang berpengaruh berdasarkan *graph*.

5.1 Data Uji Coba

Data yang digunakan untuk uji coba merupakan data *twitter* yang terlibat dalam pembahasan topik tertentu sejumlah 17410 yang diambil pada bulan Januari 2015 sampai dengan bulan Mei 2016 pada situs Kaggle. Data tersebut tersimpan di dalam *database*. Atribut yang digunakan adalah id, *tweet* dan nama pengguna.

5.2 Ekstraksi Kata Kunci

Pengujian terhadap ekstraksi kata kunci ini dilakukan dengan cara memasukkan jumlah kata kunci secara sebarang. Hal ini bertujuan untuk mengontrol jumlah kata kunci yang diinginkan serta untuk mendapatkan jumlah kata kunci yang konvergen. Sedangkan inputan dari ekstraksi kata kunci adalah hasil dari *case folding* dan *split* berdasarkan *stopword.* Dimana data yang digunakan sebanyak 17410 data *tweet* yang sudah tersimpan di dalam *database*, Hasil pengujian ini dapat dilihat pada Tabel 5.1.

Tabel 5. 1 Pengujian ekstraksi kata kunci

No	Parameter RAKE	Jumlah kata kunci
1.	1000	27366
2.	5000	36210
3	10000	37210
4.	15000	37210

Berdasarkan hasil pengujian diatas menunjukkan bahwa jumlah kata kunci yang konvergen ketika dipilih parameter RAKE >= 10000 jumlah kata kunci dengan jumlah data 17410 data *tweet.*

5.3.1 Analisa Ekstraksi Kata Kunci Menggunakan RAKE

Analisa ekstraksi kata kunci bertujuan untuk mengetahui bagaimana proses ekstraksi kata kunci menggunakan RAKE. Tahap yang dilakukan sebelum dilakukan ekstraksi kata kunci adalah preproses data. Pada tahap ini, dilakukan proses *case folding* yaitu pengubahan setiap huruf besar pada setiap kata dokumen menjadi huruf kecil juga menghilangkan tanda baca dan angka, proses *filtering* yaitu menghilangkan kata tidak penting pada database *stopword* dan proses *tokenizing* yaitu pemisahan setiap kata penyusun dokumen berdasarkan tanda baca dan *stopword.*

Berikut merupakan contoh ekstraksi kata kunci pada 5 data yang diolah dengan pengambilan 2 kata kunci tertinggi.

Tabel 5. 2 Contoh 5 data uji *tweet*

Id_Data	Data *Tweet*	Nama Pengguna
1	RT @alamriki_omar: Tired and Sick reading about Brussels attack, Do you know how many people have been killed in Iraq Since 2003? were was…	abutariq al mujahid
2	I'm not XYZ @support false report ??? https://t.co/QUTeZ3t89R	$QU4D

Tabel 5. 2 Contoh 5 data uji *tweet* (lanjutan)

Id_Data	Data *Tweet*	Nama Pengguna
3	RT @RamiAlLolah: Extremely fierce clashes between #XYZ & #Turkey army followed by multiple VBIEDs in #Bashiqa in #Iraq https://t.co/rxtPlU…	abdlrhmn
4	#AmaqAgency Scenes from Islamic State-Controlled #Palmyra City Watch: https://t.co/j06ozvUQM0 Download: https://t.co/E3qpIMjqft	Abu Musab Al-Amreeki
5	RT @RamiAlLolah: World: Crying? Boy: Yes; I'm hungry World: Not XYZ? Boy: No World: Sorry, can't do anything for you #Darayya #Syria https?…	9flames of haqq

Tabel 5. 3 Contoh hasil preproses data

Id_Data	Kandidat Kata Kunci	Nama Pengguna
1	tired	abutariq al mujahid
1	sick reading	abutariq al mujahid
1	brussels attack	abutariq al mujahid
1	people	abutariq al mujahid
1	killed	abutariq al mujahid
1	iraq since	abutariq al mujahid
1	was	abutariq al mujahid
2	xyz false repo	$QU4D
3	extremely fierce clashes between	Abdlrhmn
3	xyz	Abdlrhmn
3	turkey army	Abdlrhmn
3	multiple vbieds in	Abdlrhmn
3	bashiqa in	Abdlrhmn
3	iraq	Abdlrhmn
4	amaqagency scenes	Abu Musab Al-Amreeki
4	islamic state	Abu Musab Al-Amreeki
4	controlled	Abu Musab Al-Amreeki
4	palmyra citywatch	Abu Musab Al-Amreeki
4	download	Abu Musab Al-Amreeki
5	world	9flames of haqq
5	crying	9flames of haqq
5	boy	9flames of haqq
5	yes	9flames of haqq
5	hungryworld	9flames of haqq
5	xyz	9flames of haqq
5	boy	9flames of haqq
5	noworld	9flames of haqq
5	sorry	9flames of haqq
5	can	9flames of haqq
5	you	9flames of haqq
5	darayya	9flames of haqq
5	syria https	9flames of haqq

Setelah dilakukan preproses data untuk menghasilkan kandidat kata kunci, tahap selanjutnya adalah pembentukan ekstraksi kata kunci menggunakan RAKE.

Tabel 5. 4 Contoh hasil ekstraksi kata kunci dengan input 2 kata kunci

Id_Data	Kata Kunci	Nama Pengguna
1	sick reading	abutariq al mujahid
1	brussels attack	abutariq al mujahid
2	xyz false repo	$QU4D
3	extremely fierce clashes between	Abdlrhmn
3	multiple vbieds in	Abdlrhmn
4	amaqagency scenes	Abu Musab Al-Amreeki
4	islamic state	Abu Musab Al-Amreeki
5	hungryworld	9flames of haqq
5	syria https	9flames of haqq

Pada Tabel 5.4 merupakan hasil ekstraksi kata kunci menggunakan RAKE dengan diambil 2 hasil perhitungan nilai fitur dasar tertinggi tiap data.

5.3 Penarikan Pengguna ke dalam *Term* Topik

Penarikan pengguna ke dalam *term* topik merupakan proses untuk membentuk relasi antara pengguna satu dengan pengguna yang lainnya. Contoh proses ini menggunakan 5 data yang terdapat pada Tabel 5.2 yang sudah di lakukan preproses data dan ekstraksi kata kunci. Kemudian dilakukan *stringbuilder* untuk mengabungkan kata kunci menjadi kalimat berdasarkan nama pengguna.

Tabel 5. 5 Contoh data *tweet* hasil *stringbuilder*

Nama Pengguna	Data *Tweet*
$QU4D	xyz false repo
9flames of haqq	syria http hungryworld noworld crying boy yes sorry can you darayya
abdlrhmn	extremely fierce clashes between multiple vbieds in turkey army bashqia in xyz iraq
Abu Musab Al-Amreeki	amaqagency scenes islamic state palmyra citywatch controlled download
Abutariq al mujahid	sick reading brussels attack iraq since tired people killed was

Setelah proses *stringbuilder* maka akan dilakukan proses penarikan pengguna ke dalam *term* topik.

Tabel 5. 6 Contoh penarikan pengguna ke dalam *term* topik

Id_*Term*_Topik	*Term* Topik	Nama Pengguna
1	xyz	$QU4D
7	al	$QU4D
3	syria	9flames of haqq
4	http	9flames of haqq
7	http	9flames of haqq
1	xyz	Abdlrhmn
1	iraq	Abdlrhmn
3	turkey	Abdlrhmn
8	army	Abdlrhmn
8	iraq	Abdlrhmn
8	turkey	Abdlrhmn
4	state	Abu Musab Al-Amreeki
4	islamic	Abu Musab Al-Amreeki
4	control	Abu Musab Al-Amreeki
6	myra	Abu Musab Al-Amreeki
6	islam	Abu Musab Al-Amreeki

Tabel 5. 6 Contoh penarikan pengguna ke dalam *term* topik (lanjutan)

Id_*Term*_Topik	*Term* Topik	Nama Pengguna
7	Al	Abu Musab Al-Amreeki
7	islam	Abu Musab Al-Amreeki
9	islam	Abu Musab Al-Amreeki
10	amaqagency	Abu Musab Al-Amreeki
10	city	Abu Musab Al-Amreeki
1	iraq	abutariq al mujahid
1	attack	abutariq al mujahid
2	killed	abutariq al mujahid
2	attack	abutariq al mujahid
8	iraq	abutariq al mujahid
9	people	abutariq al mujahid

Dari hasil penarikan pengguna ke dalam *term* topik, kemudian akan dilakukan proses penggolongan pengguna satu dengan pengguna yang lain berdasarkan *term* topik yang sama.

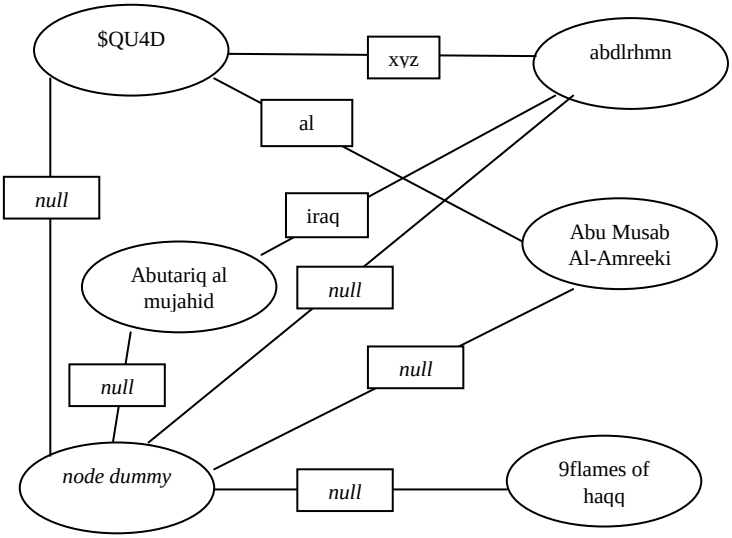
Tabel 5. 7 Contoh hasil proses penggolongan pengguna berdasarkan *term* topik yang sama

Id_*Term*_Topik	*Term* Topik	Nama Pengguna	
		1	2
1	xyz	$QU4D	abdlrhmn
1	Iraq	abdlrhmn	abutariq al mujahid
1	attack	abutariq al mujahid	
2	killed	abutariq al mujahid	
3	syria	9flames of haqq	
3	turkey	abdlrhmn	

Tabel 5. 7 Contoh hasil proses penggolongan pengguna berdasarkan *term* topik yang sama (lanjutan)

Id_*Term*_Topik	*Term* Topik	Nama Pengguna	
		1	2
4	state	Abu Musab Al-Amreeki	
4	islamic	abu musab al-amreeki	
4	http	9flames of haqq	
4	control	Abu Musab Al-Amreeki	
6	myra	Abu Musab Al-Amreeki	
6	islam	Abu Musab Al-Amreeki	
7	al	$QU4D	Abu Musab Al-Amreeki
7	islam	Abu Musab Al-Amreeki	
7	http	9flames of haqq	
8	iraq	abdlrhmn	abutariq al mujahid
8	turkey	abdlrhmn	
9	people	abutariq al mujahid	
9	islam	Abu Musab Al-Amreeki	
10	amaqagency	Abu Musab Al-Amreeki	
10	city	Abu Musab Al-Amreeki	

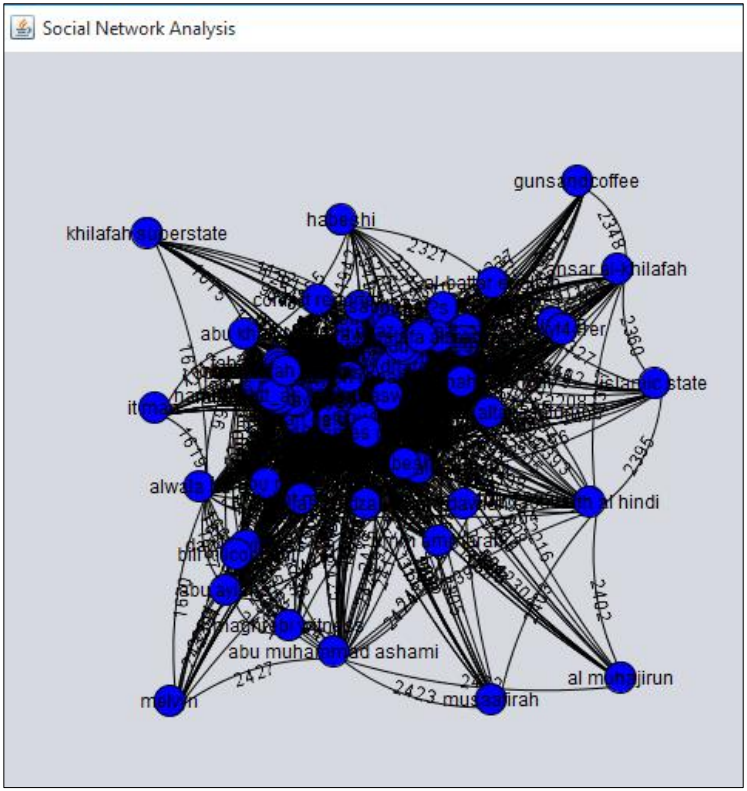
Berdasarkan Tabel 5.7 menunjukkan bahwa pengguna yang mempunyai relasi dapat dilihat pada Gambar 5.1.



Gambar 5. 1 Relasi pengguna berdasarkan *term* topik yang sama

5.4 Pembentukan *Graph Social Network Analysis*

Pembentukkan *graph social network analysis* merupakan visualisasi relasi yang mengambarkan sekumpulan *node* (aktor) yang terkumpul dalam bentuk *graph*. Berikut ini adalah *graph* jejaring sosial pengguna *tweet* yang terbentuk secara keseluruhan.



Gambar 5. 2 Visualisasi *graph* SNA secara keseluruhan

Perhitungan nilai *centrality* baik *degree, betweeness* maupun *closeness*, bertujuan untuk menganalisa jejaring sosial yang telah terbentuk pada setiap individu, serta mengetahui *centrality* dari struktur jaringan relasi yang dimiliki oleh masing-masing aktor.

Berikut ini menyatakan hasil perhitungan *degree*, *betweeness* maupun *closeness centrality* yang sudah dinormalisasikan terhadap setiap pengguna pada jejaring sosial yang terbentuk. Tabel-tabel yang disuguhkan,

mengambil 20 pengguna *twitter* yang mempunyai nilai tertinggi dalam masing-masing perhitungan.

5.5.1 Nilai *Normalized Degree Centrality*

Nilai *normalized degree centrality* digunakan untuk menentukan aktor yang paling berpengaruh berdasarkan banyaknya *edge* atau hubungan yang terjadi antara sebuah *node* dengan *node* yang lainnya. Nilai *normalized degree centrality* sebagai tolak ukur banyaknya interaksi pengguna *twitter* terhadap pengguna *twitter* lainnya. Semakin tinggi nilai *degree* seorang pengguna *twitter*, menunjukkan semakin banyak pengguna yang mempunyai keterkaitan topik *tweet* dengannya. Hasil dari pengukuran *normalized degree centrality* secara keseluruhan dapat dilihat pada Tabel 5.8.

Tabel 5. 8 Hasil perhitungan *normalized degree centrality*

No	Nama Pengguna	*Normalized Degree Centrality*
1	$qu4d	0.9907407407407407
2	Abdlrhmn	0.9814814814814815
3	abutariq al mujahid	0.9814814814814815
4	abu musab al-amreeki	0.9814814814814815
5	9flames of haqq	0.9814814814814815
6	10flames of haqq	0.9814814814814815
7	abu hanzalah	0.9722222222222222
8	abdus mujahid	0.9537037037037037
9	abu naseeha	0.9444444444444444
10	al-aswad	0.9444444444444444
11	abu ibn taha	0.9351851851851852
12	a. ibrahim	0.9074074074074074
13	abu dharda	0.9074074074074074
14	abu_azzam25	0.9074074074074074
15	abu rikaz alhijazi	0.8981481481481481
16	asim abu merjem	0.8703703703703703
17	abu khalid	0.8703703703703703

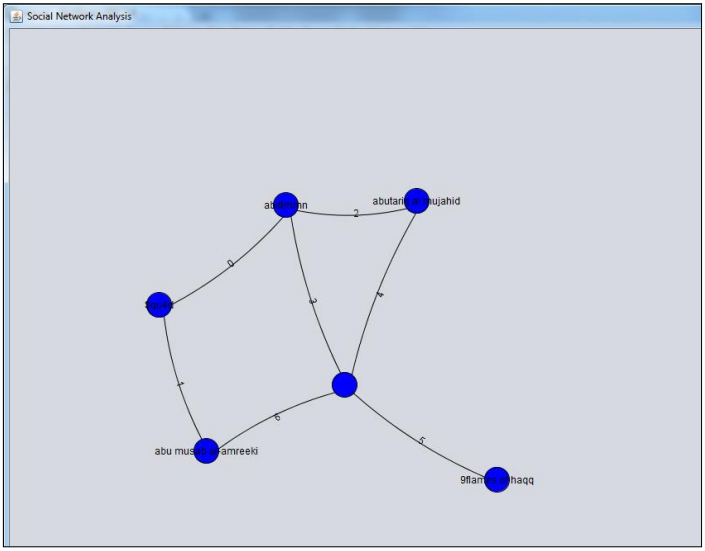
Tabel 5. 8 Hasil Perhitungan *normalized degree centrality* (lanjutan)

No	Nama Pengguna	*Normalized Degree Centrality*
18	catacat coconuty	0.7685185185185185
19	alwala bara	0.7592592592592593
20	conflict reporter	0.75

Berdasarkan Tabel 5.8 menunjukkan bahwa $qu4d memiliki nilai *degree centrality* tertinggi yaitu 0.9907407407407407. Hal ini dapat diartikan bahwa $qu4d memiliki hubungan atau interaksi yang berupa *mention, quote retweet* atau *reply* antar pengguna lain sebanyak 107 terkait pembahasan topik tertentu pada *twitter*.

5.5.1.1 Analisa Hasil *Normalized Degree Centrality*

Berdasarkan hasil nilai *normalized degree centrality* pada Tabel 5.8, dilakukan analisa pada perhitungan *normalized degree centrality* secara manual untuk mengetahui apakah hasil yang didapatkan sesuai. Berikut merupakan perhitungan *normalizd degree centrality* secara manual dapat dilihat pada penjelasan dibawah ini dengan menggunakan 5 data *tweet* pada tabel 5.2 dan *graph* yang terbentuk dapat dilihat pada Gambar 5.3.



Gambar 5. 3 Visualisasi *graph* SNA menggunakan 5 data *tweet* untuk perhitungan *normalized degree*, *closeness* dan *betweeness centrality*

Perhitungan *normalized degree centrality* secara manual dengan menggunkan Persamaan 2.4.

- $C_D(\ ) = \frac{4}{5} = 0{,}8$
- $C_D(\text{abdlrhmn}) = \frac{3}{5} = 0{,}6$
- $C_D(\$\text{qu4d}) = C_D(\text{abu musab al} - \text{amreeki})$
  $= C_D(\text{abut}ariq\ \text{al mujahid})$
  $= \frac{2}{5} = 0{,}4$
- $C_D(\text{9flames of haqq}) = \frac{1}{5} = 0{,}2$

Tabel 5. 9 Hasil perhitungan *normalized degree centrality* menggunakan 5 data

No	Nama Pengguna	*Normalized Degree Centrality*
1		0.8
2	Abdlrhmn	0.6
3	$qu4d	0.4
4	abutariq al mujahid	0.4
5	abu musab al-amreeki	0.4
6	9flames of haqq	0.2

Berdasarkan percobaan dengan 5 data *tweet*, diperoleh hasil perhitungan manual yang sama dengan Tabel 5.9.

5.5.2 Nilai *Normalized Closeness Centrality*

Nilai *normalized closeness centrality* dilihat dari *node-node* yang dapat menjangkau *node* lainnya dengan jalur yang lebih pendek. Semakin mendekati nilai 1 atau sama dengan 1, maka semakin dekat *node* tersebut dengan *node* yang lain atau semakin *closeness node* tersebut. Pengguna *twitter* yang memiliki nilai *normalized closeness centrality* tertinggi disebabkan karena *node* ini memiliki akses yang cepat ke pengguna *twitter* lainnya. Kedekatan jarak antar pengguna *twitter* menyatakan topik *tweet* memiliki keterkaitan yang erat. Hasil perhitungan *normalized closeness centrality* dapat dilihat pada Tabel 5.10.

Tabel 5. 10 Hasil perhitungan *normalized closeness centrality*

No	Nama Pengguna	*Normalized Closeness Centrality*
1	$qu4d	1
2	Abdlrhmn	0.9907407407407408
3	abutariq al mujahid	0.9907407407407408
4	abu musab al-amreeki	0.9907407407407408
5	9flames of haqq	0.9907407407407408
6	10flames of haqq	0.9907407407407408
7	abu hanzalah	0.981651376146789
8	abdus mujahid	0.963963963963964
9	abu naseeha	0.9553571428571429
10	al-aswad	0.9553571428571429
11	abu ibn taha	0.9469026548672567
12	a. ibrahim	0.9224137931034484
13	abu dharda	0.9224137931034484
14	abu_azzam25	0.9224137931034484
15	abu rikaz alhijazi	0.9145299145299145
16	asim abu merjem	0.8916666666666667
17	abu khalid	0.8916666666666667
18	catacat coconuty	0.816793893129771
19	alwala bara	0.8106060606060607
20	conflict reporter	0.8045112781954887

Berdasarkan Tabel 5.10 menjukkan bahwa nilai *normalized closeness centrality* dengan peringkat pertama adalah $qu4d, sedangkan nilai *normalized closeness centrality* dengan peringkat kedua terdapat 5 akun pengguna *twitter* yaitu abdlrhmn, abutariq al mujahid, abu musab al-amreeki, 9flames of haqq dan 10flames of haqq. Nilai *normalized closeness centrality* untuk peringkat pertama yaitu 1 dan untuk peringkat kedua yaitu 0.9907407407407408. Selisih nilai kedua *node* antara 0.0092592592. Hal ini mengakibatkan *node* dengan nilai *normalized closeness centrality* 1 lebih

cepat dan lebih mudah dalam berkomunikasi dengan *node* lain ketika melakukan *mention, quote retweet* atau *reply* tanpa melalui banyak perantara yang harus dilalui. Akan tetapi *node-node* dengan nilai *normalized closeness centrality* 0.9907407407407408 sama baiknya dalam hal kecepatan dan kemudahan berkomunikasi dengan *node* lain tanpa melalui banyak perantara yang harus dilalui.

5.5.2.1 Analisa Hasil *Normalized Closeness Centrality*

Berdasarkan hasil nilai *normalized closeness centrality* , perhitungan *normalized closeness centrality* secara manual untuk mengetahui apakah hasil yang didapatkan sesuai. Berikut merupakan perhitungan *normalized closeness centrality* secara manual dapat dilihat pada penjelasan dibawah ini dengan menggunakan 5 data dan *graph* yang terbentuk dapat dilihat pada gambar 5.3.

Perhitungan *normalized closeness centrality* secara manual dengan menggunkan Persamaan 2.6.

- $C_D(\ ) = \frac{5}{1+1+1+1+2} = 0{,}833333333333334$
- $C_D(\text{abdlrhmn}) = \frac{5}{1+1+1+2+2} = 0{,}71428571427143$
- $C_D(\text{abu musab al} - \text{amreeki}) = \frac{5}{1+1+2+2+2} = 0{,}625$
- $C_D(\text{abut}ari\text{q al mujahid}) = \frac{5}{1+1+2+2+2} = 0{,}625$
- $C_D(\$\text{qu4d}) = \frac{5}{1+1+2+2+3} = 0{,}55555555555556$
- $C_D(9\text{flames of haqq}) = \frac{5}{1+2+2+2+3} = 0{,}5$

Tabel 5. 11 Hasil perhitungan *normalized closeness centrality* menggunakan 5 data

No	Nama Pengguna	*Normalized Closeness Centrality*
1		0.8333333333333334
2	Abdlrhmn	0.7142857142857143
3	abu musab al-amreeki	0.625
4	abutariq al mujahid	0.625
5	$qu4d	0.555555555555556
6	9flames of haqq	0.5

Berdasarkan percobaan dengan 5 data diperoleh hasil perhitungan manual yang sama dengan Tabel 5.11.

5.5.3 Nilai *Normalized Betweeness Centrality*

Nilai *normalized betweeness centrality* digunakan untuk mengetahui posisi *node* dalam *network* dimana *node* tersebut tidak boleh hilang. Jika *node* tersebut hilang, maka akan terjadi gangguan komunikasi dalam *network*. Hal ini merupakan symbol "kekuatan pengaruh" seorang pengguna *twitter* berdasarkan topik pembicaraan *tweet* tertentu. Hasil perhitungan *normalized betweeness centrality* dapat dilihat pada Tabel 5.12.

Tabel 5. 12 Hasil perhitungan *normalized betweeness centrality*

No	Nama Pengguna	*Normalized Betweeness Centrality*
1	$qu4d	0.06487318077207006
4	abu musab al-amreeki	0.062399993897922494
5	9flames of haqq	0.062399993897922494
2	Abdlrhmn	0.06208747874902016
3	abutariq al mujahid	0.06208747874902016
6	10flames of haqq	0.06208747874902016
7	abu hanzalah	0.06069156723195252
8	abdus mujahid	0.05582898653139262
10	al-aswad	0.054124157715740466
9	abu naseeha	0.05286269369354635
11	abu ibn taha	0.05166110207196789
12	a. ibrahim	0.046696840016826176
13	abu dharda	0.046696840016826176
15	abu rikaz alhijazi	0.04477636098575192
14	abu_azzam25	0.04460944805624963
16	asim abu merjem	0.04134334898398169
17	abu khalid	0.040961532909363656
19	alwala bara	0.028921602222118852
18	catacat coconuty	0.026391679751998843
20	conflict reporter	0.02380362936511921

Berdasarkan Tabel 5.12 menunjukkan bahwa *node* dengan nama pengguna $qu4d memiliki nilai *normalized betweenes centrality* tertinggi dalam *network* dengan nilai *normalized betweenes centrality* 0.06487318077207006. Artinya $qu4d adalah aktor penghubung atau jembatan dari seluruh aliran informasi dalam percakapan mengenai pembahasan topik tertentu di *twitter*. Hal ini mengakibatkan banyak *node* lain yang bergantung pada *tweet* yang di-*posting* oleh $qu4d di *twitter* karena *node* tersebut sebuah jembatan bagi *node* lain.

5.5.3.1 Analisa Hasil *Normalized Betweeness Centrality*

Berdasarkan hasil niali *normalized betweeness centrality* pada Tabel 5.7, dilakukan analisa pada perhitungan *normalized betweeness centrality* secara manual untuk mengetahui apakah hasil yang didapatkan sesuai. Berikut merupakan perhitungan *normalized betweenes centrality* secara manual dapat dilihat pada penjelasan dibawah ini dengan menggunakan 5 data *tweet* dan *graph* yang terbentuk dapat dilihat pada Gambar 5.2.

Tabel 5. 13 Hubungan antar *node* dalam *network*

Source Node	*Target Node*	*Intermediate Nodes in the Path*	*Path*
A	B		AB
A	C		AC
A	D	B / E	ABD / AED
A	E		AE
A	F	E	AEF
B	C	A	BAC
B	D		BD
B	E	A / D	BAE / BDE
C	D	E	CED
C	E		CE
C	F	E	CEF
D	E		DE
D	F	E	DEF
E	F		EF
Total Path			18

Keterangan:

A = abdlrhmn
B = $qu4d
C = abutariq al mujahid
D = abu musab al-amreeki
E = *node dummy*
F = 9fames of haqq

Perhitungan *normalized betweeness centrality* secara manual dengan menggunkan persamaan 5.3.

- $C_D(\ ) = \frac{2 \text{ x } \frac{4+0,5}{18}}{5 \text{ x } 4} = 0{,}025$
- $C_D(\text{abdlrhmn}) = \frac{2 \text{ x } \frac{1+0,5}{18}}{5 \text{ x } 4} = 0{,}0083333333$
- $C_D(\text{abu musab al} - \text{amreeki}) = \frac{2 \text{ x } \frac{0,5}{18}}{5 \text{ x } 4} = 0{,}002777778$
- $C_D(\text{abut}ari\text{q al mujahid}) = \frac{2 \text{ x } \frac{0}{18}}{5 \text{ x } 4} = 0$
- $C_D(\$\text{qu4d}) = \frac{2 \text{ x } \frac{0,5}{18}}{5 \text{ x } 4} = 0{,}0083333333$
- $C_D(\text{9flames of haqq}) = \frac{2 \text{ x } \frac{0}{18}}{5 \text{ x } 4} = 0$

Tabel 5. 14 Hasil perhitungan normalized betweeness centrality menggunakan 5 data

No	Nama Pengguna	*Normalized Betweeness Centrality*
1		0.025
2	Abdlrhmn	0.0083333333
3	$qu4d	0.0083333333
4	abu musab al-amreeki	0.0027777778
5	abutariq al mujahid	0
6	9flames of haqq	0

Berdasarkan percobaan dengan 5 data *tweet*, diperoleh hasil perhitungan manual yang sama dengan Tabel 5.

BAB VI
PENUTUP

Pada bab ini berisi tentang beberapa kesimpulan yang dihasilkan berdasarkan penelitian yang telah dilaksanakan.dan saran yang dapat digunakan jika penelitian ini dikembangkan.

6.1 Kesimpulan

Berdasarkan analisis terhadap hasil pengujian program, maka dapat diambil kesimpulan sebagai berikut:

1. Metode *Social Network Analysis (*SNA) telah berhasil diterapkan untuk melakukan identifikasi pengguna yang berpengaruh berdasarkan *graph* yang terbentuk dan dihasilkan 108 *node* (pengguna) *twitter* yang terlibat dalam pembahasan topik tertentu pada *twitter* bulan Januari 2015 sampai bulan Mei 2016 dengan total 17410 *tweet* yang terdapat pada situs Kaggle.
2. Aktor (*node)* atau pengguna yang berpengaruh dalam *network twitter* pada bulan Januari 2015 sampai bulan Mei 2016 secara keseluruhan adalah $qu4d. Pengguna ini memiliki peringkat tertinggi pada *normalized degree centrality*, *normalized closeness centrality* dan *normalized betweeness centrality* dengan nilai 0.9907407407407407; 1; 0.06487318077207006. Hal ini menandakan bahwa nama pengguna $qu4d merupakan aktor yang berpengaruh, aktor yang menjadi jembatan komunikasi antar pengguna dan terkait dengan banyaknya pengguna yang memiliki keterkaitan yang tinggi pada *twitter* yang membahas tentang topik tertentu.

6.2 Saran

Ada beberapa hal yang penulis sarankan untuk pengembangan selanjutnya yaitu:

1. Data *tweet* yang digunakan terbatas hanya 17410 *tweet*. Untuk hasil yang lebih bagus dapat dilakukan dengan memperbanyak data yang diolah agar interaksi yang diamati lebih banyak.
2. Penulis berharap penelitian selanjutnya melakukan uji coba penelitian dengan menggunakan RAM komputer yang cukup tinggi untuk menghindari komputasi yang lama.

DAFTAR PUSTAKA

[1] Techinasia. "Statistik Pengguna Internet dan Media Sosial Terbaru 2015", https://id.techinasia.com/talk/statistik-pengguna-internet-dan-media-sosial-terbaru-2015, (diakses pada 8 Februari 2017). Situs Berita Techinasia.

[2] Aini, N.A., & Andry, A. (2015). "Analisis pada Peringkat Top Brand Menggunakan Jejaring Sosial Percakapan dengan *Social Network Analysis* (Studi Kasus pada *Smartphone* Samsung, Blackberry, Nokia, Iphone di Indonesia)". Tugas Akhir Manajemen Bisnis Telekomunikasi dan Informatika, Fakultas Ekonomi Bisnis, Universitas Telkom, Indonesia.

[3] Kominfo. "Pengguna Internet di Indonesia 63 Juta Orang", https://kominfo.go.id/index.php/content/detail/3415/Kominfo+%3A+Pengguna+Internet+di+Indonesia+63+Juta+Orang/0/berita_satker, (diakses pada 8 Februari 2017). Situs Berita Kominfo.

[4] Tsvetovat, Maksim., Kouznetsov, Alexander. (2011). "Social Network Analysis for Startups". USA:O'reilly Media.

[5] Alamsyah, Andry. Oktora Rio. (2014). "Pola Interaksi dan Aktor yang paling Berperan pada Event JGTC 2013 melalui Media Sosial Twitter (Studi Menggunakan Metode Social Network Analysis)". Volume 14. No. 3. Jurnal Manajemen Indonesia.

[6] Sucipto, R.H. (2015). "Analisis Jaringan Teks Berdasarkan *Social Network Analysis* dan *Text Mining* untuk Mengetahui Persepsi Kualitas Merek pada Konten Percakapan di Media Sosial Twitter (Studi Kasus pada PT. Indosat Tbk. dan PT. Telkomsel)". Tugas Akhir Manajemen Bisnis Telekomunikasi dan Informatika, Universitas Telkom, Indonesia.

[7] Li, Q., Bhavya, K., Jayaraman, J.T., Zhenliang, Z., Pramod, K.V. (2016). "Influential Node Detection in Implicit Sosial Network using Multi-task Gaussian Copula Models". Journal of Machine Learning Research. Lawrence Livermore National Laboratory, Syracuse University, Amerika.

[8] Rukmi, A.M. (2016). "Kajian Algoritma K-Means++ dan Graf pada Analisis Jejaring Sosial untuk Pengklasteran Peneliti". Riset Text Mining Departemen Matematika, Intitut Teknologi Sepuluh Nopember Surabaya, Indonesia.

[9] Kumar S., Morstatter F., dan Liu. (2013). "Twitter Data Analytics". SpringerBriefs in Computer Science.

[10] A. Culotta. (2010). "Detecting Influenza Outbreaks by Analyzing Twitter Messages". Jurnal Departement of Computer Science Southeastern Lousiana University, Hammond, LA 70402.

[11] Michael W.B, Jacob Kogan. (2010). "Text Mining : Applications and Theory.

[12] Mohamed Abubaker, Wesam Ashour. (2013). "Efficient Data Clustering Algorithms : Improvements over K-Means", I.J Intelligent Systems and Applicationa.

[13] Wasserman, S., Faust, K. (1994). " Social Network Analysis: Method and Applications". United States of America, Cambridge University Press.

[14] Borgatti, Steve. (2005). "Introduction to Social Network Analysis", http://www.analyttictecth.com/borgatti/papers/borgatti%20%20Special%20issue%20on%20blockmodels%20-%20Introduction.pdf, (diakses pada 18 Februari 2017).

[15] Lin, Yan., Chen, Yan., Li Jianfeng. (2010). "Research on Traffic Layout Based on Social Network Analysis". Management Science and Engineering, Dalian Maritime University.

LAMPIRAN A
Data Topik

Terdapat 10 topik yang digunakan dalam tugas akhir ini

id	Term Topik	id	Term Topik	Id	Term Topik
1	xyz	2	Killed	3	Syria
1	ramiallolah	2	Soldiers	3	Russia
1	Iraq	2	Today	3	ramiallolah
1	Attack	2	Airstrikes	3	turkey
1	Libya	2	Injured	3	ypg
1	warreporter1	2	Wounded	3	breakingnews
1	Saa	2	Civilians	3	usa
1	aamaq	2	Militants	3	group
1	Usa	2	Iraqi	3	saa
1	Abu	2	Attack	3	terror

id	Term Topik	id	Term Topik	Id	Term Topik
4	State	5	Aleppo	6	assad
4	Islamic	5	Nid	6	regime
4	fighters	5	Gazaui	6	myra
4	fighting	5	Rebels	6	forces
4	Group	5	North	6	rebels
4	Saudi	5	Today	6	fsa
4	new	5	northern	6	pro
4	http	5	Syrian	6	islam
4	wilaya	5	Ypg	6	syrian
4	control	5	turkish	6	jaysh

id	Term Topik	Id	Term Topik
7	al	**Id**	**topik**
7	qaeda	8	army
7	nusra	8	iraq
7	abu	8	near
7	sham	8	ramiallolah
7	ahrar	8	iraqi
7	islam	8	lujah
7	jabhat	8	turkey
7	http	8	ramadi
7	warreporter1	8	west

id	Term Topik	id	Term Topik
9	allah	10	breaking
9	people	10	islamicstate
9	muslims	10	forces
9	abu	10	amaqagency
9	accept	10	city
9	muslim	10	fighters
9	make	10	iraqi
9	know	10	near
9	don	10	area
9	islam	10	syrian

LAMPIRAN B

Source code

1. Preproses Data

```
package yakinbisa;

import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.LinkedList;

public class RAKE {

    Stopword stop = new Stopword();
    ArrayList<Kata> listKata;
    ArrayList<KataKunci> listKataKunci;
    LinkedList<String> listStopword =
stop.readStopword();

    public void ProsesRake(DokumenTweet dokumen, int
jum_kk) throws SQLException{
        listKata = new ArrayList<>();
        listKataKunci = new ArrayList<>();
        Database db = new Database();
        db.connectFirst();
        String Dok = dokumen.getStatusPengguna();
        ArrayList<String> tempKata = new ArrayList<>();
        /*Merubah string menjadi huruf kecil*/
        String hasil = prosesLowerCase(Dok);
        /*Split berdasarkan tanda baca*/
        ArrayList<String> kandidatKK =
prosesSplitTB(hasil);
        /*Split berdasarkan stopword*/
        kandidatKK = prosesSplitStopword(kandidatKK);

        /*huruf yg mempunyai panjang >2 akan disimpan
ke db*/
        for(String s : kandidatKK){
            if(s.length() > 2){
//                System.out.println(s);
                String query = "INSERT INTO
representasi_dok_rake VALUES('"+ dokumen.getId() +"','"
+ s + "')";
                db.executeUpdate(query);}}
```

```
        /*pisahkan seluruh kandidatkk menjadi satu
kata*/
        for(String s: kandidatKK){
            String[] tempsplit = s.split(" ");
            for(int i = 0; i< tempsplit.length; i++){
                tempKata.add(tempsplit[i]);}}

        /*Mencari frekuensi kata*/
        for(String s: tempKata){

            if(listKata.size() == 0){
                Kata katabaru = new Kata(s, 1, 0, 0);
                listKata.add(katabaru);
            }
            else{
                if(isAda(s)){

listKata.get(findIndex(s)).setFrek(listKata.get(findInd
ex(s)).getFrek()+1);
                }
                else{
                    Kata katabaru = new Kata(s, 1, 0,
0);
                    listKata.add(katabaru);}}}
        /*Mencari Degree*/
        for(String s: kandidatKK){
            int jumlah = 0;
            String[] split = s.split(" ");
            jumlah = split.length;
            if(jumlah>1){
                for(Kata k: listKata){
                    if(s.contains(k.getKata())){
                        k.setDeg(k.getDeg()+1);}}}}
        for(Kata k: listKata){
            k.setDeg(k.getDeg()+k.getFrek());
            k.setRasio(k.getDeg()/k.getFrek());}
        /*Mencari Skor*/
        for(String s: kandidatKK){
            double skor = 0;
            for(Kata k:listKata){
                if(s.contains(k.getKata())){
                    skor += k.getRasio() ;}}
            KataKunci kk = new KataKunci(s, skor);
            listKataKunci.add(kk);}

        /*Sorting kandidat kata kunci menurut skor*/
        Collections.sort(listKataKunci, new
Comparator<KataKunci>() {
```

```
        /*Sorting kandidat kata kunci menurut skor*/
        Collections.sort(listKataKunci, new
Comparator<KataKunci>() {

            @Override
            public int compare(KataKunci t, KataKunci
t1) {
                if(t.getSkor() > t1.getSkor()){
                    return 1;
                }
                else if (t.getSkor() < t1.getSkor()){
                    return -1;
                }
                else
                    return 0;}});

        int count = 0;
        jum_kk = listKata.size()/3;
         for(KataKunci k:listKataKunci){
            if(count < jum_kk){
                if(k.getKata().length() > 2){
                    ResultSet rs =
db.executeSelect("SELECT kata_kunci FROM
list_kata_kunci WHERE kata_kunci =
\""+k.getKata()+"\"");
                    if(!rs.first()){
                        String query = "INSERT INTO
list_kata_kunci VALUES('"+k.getKata().replaceAll("[^a-
zA-Z ]", "")+"')";
                        db.executeUpdate(query);
                        count++;}}}
            else{
                break;
            }
        }
        db.destroyConnection();
    }

    /*proses pembobotan*/
    public void prosesRAKE2(int jum_kk) throws
SQLException{

        Database db = new Database();
        db.connectFirst();

        db.executeUpdate("DELETE FROM
list_kata_kunci_norake");
```

```
String query = "SELECT DISTINCT id_dok FROM
representasi_dok_rake";
        ResultSet rsdok = db.executeSelect(query);
        ArrayList<String> listDokumen = new
ArrayList<>();

        while(rsdok.next()){
            listDokumen.add(rsdok.getString("id_dok"));
        }
        System.out.println(listDokumen.size());

        for(String str : listDokumen){
            listKata = new ArrayList<>();
            listKataKunci = new ArrayList<>();
            ArrayList<String> kandidatKK = new
ArrayList<>();
            ArrayList<String> tempKata = new
ArrayList<>();
            query = "SELECT frase_rep FROM
representasi_dok_rake WHERE id_dok = \""+str+"\"";
            rsdok = db.executeSelect(query);
            while(rsdok.next()){

kandidatKK.add(rsdok.getString("frase_rep"));
            }
          for(String s: kandidatKK){
           String[] tempsplit = s.split(" ");
            for(int i = 0; i< tempsplit.length; i++){
                    tempKata.add(tempsplit[i]);}}
            for(String s: kandidatKK){
                String[] tempsplit = s.split(" ");
            for(int i = 0; i< tempsplit.length; i++){
                    tempKata.add(tempsplit[i]);}}
                       return 0;
    //lower case
    private String prosesLowerCase(String
abstraksiDok){
        abstraksiDok = abstraksiDok.toLowerCase();
        return abstraksiDok;
    }

    //pemisahan menurut tanda baca, url, retweet
    private ArrayList<String> prosesSplitTB(String
abstraksiDok){

        ArrayList<String> hasil = new
ArrayList<String>();
        String URL1 = "((http|https)://|www[.])[-A-Za-
```

```
z0-9+&amp;amp;@#/%?=~_()|!:,.;]*[-A-Za-z0-
9+&amp;amp;@#/%=~_()|]";
        String URL2 =
"((www\\[\\s]+)|https?://[^\\s])";
        String LAIN = "@([^\\s]+)";
        String retweet = "(rt)";
        String URL3 = "(amp)";
        abstraksiDok =
abstraksiDok.replaceAll(URL1,"");
        abstraksiDok =
abstraksiDok.replaceAll(URL2,"");
        abstraksiDok =
abstraksiDok.replaceAll(LAIN,"");
        abstraksiDok =
abstraksiDok.replaceAll(retweet,"");
        abstraksiDok =
abstraksiDok.replaceAll(URL3,"");
        abstraksiDok = abstraksiDok.replaceAll("[0-9]",
"");
        abstraksiDok =
abstraksiDok.replaceAll("(\\r|\\n)","");

        String[] tempHasil =
abstraksiDok.split("\\p{Punct}");
        for(int i = 0; i < tempHasil.length; i++){
        if(tempHasil[i].length() > 2){
            StringBuilder sb = new StringBuilder();
            String[] buildString = tempHasil[i].split("
");
            for(int j = 0; j < buildString.length;
j++){
                if(buildString[j].length() > 1){
                    if(j < buildString.length-1){
                        sb.append(buildString[j]);
                        sb.append(" ");
                    }
                    else{
                        sb.append(buildString[j]);}}}
            hasil.add(sb.toString());}}
        return hasil;}
    //pemisahan menurut stopword
    private ArrayList<String>
prosesSplitStopword(ArrayList<String> splittedWord){

        ArrayList<String> kandidatKataKunci = new
ArrayList<>();
        for(String s: splittedWord){
            ArrayList<String> hasil = isStopword(s);
```

```
for(String st: hasil){
                st = st.replaceAll("(\\r|\\n)","");
                if(st.length() > 1){
                kandidatKataKunci.add(st);}}}
        return kandidatKataKunci;}
    private ArrayList<String> isStopword(String kata){
        ArrayList<String> splitKata2 = new
ArrayList<>();
    ArrayList<String> kataKunci = new ArrayList<>();
            String[] splitKata = kata.split(" ");
            for(int i = 0; i < splitKata.length; i++){
              if(i == splitKata.length - 1){
                    splitKata2.add(splitKata[i]);
             StringBuilder sb = new StringBuilder();
for(int j = 0; j < splitKata2.size(); j++){
                sb.append(splitKata2.get(j));
if(j != splitKata2.size() - 1 ){
                            sb.append(" ");}}
                    kataKunci.add(sb.toString());
                    splitKata2 = new ArrayList<>();
                }
                else
if(!listStopword.contains(splitKata[i])){
                    splitKata2.add(splitKata[i]);
                }
                else if
(listStopword.contains(splitKata[i])) {
              StringBuilder sb = new StringBuilder();
for(int j = 0; j < splitKata2.size(); j++){
                        sb.append(splitKata2.get(j));
                    if(j != splitKata2.size() - 1 ){
                            sb.append(" ");}}
                    kataKunci.add(sb.toString());
                    splitKata2 = new ArrayList<>();}}
          return kataKunci”    }
    private boolean isAda(String s) {

        for(Kata k:listKata){
            if(k.getKata().equals(s)){
                return true;}}
        return false;}
    private int findIndex(String s) {
        int index;
        for(Kata k:listKata){
            if(k.getKata().equals(s)){
                return listKata.indexOf(k);}}
        return 0;}}
```

2. Social Network Analysis

```
package yakinbisa;

public class SNA {
public ArrayList<String> formRelasiPengguna2() throws
FileNotFoundException, SQLException{

        ArrayList<String> netStrenght = new
ArrayList<>();
        DokumenTweet dok = new DokumenTweet();
        Pengguna pengguna = new Pengguna();

        LinkedList<DokumenTweet> listDok =
dok.readPartDokumen();
        LinkedList<String> listPengguna =
pengguna.listPenggunaInDokumen();
        int startidx = 0;
        Integer edgelbl = 0;
        for(String pengguna1 : listPengguna){
            pengguna1 = pengguna1.toLowerCase();
            for(int i = startidx; i <
listPengguna.size(); i++){
                String pengguna2 =
listPengguna.get(i).toLowerCase();
                Double d1uniond2 = 0.0;
                Double jumd1 = 0.0;
                Double jumd2 = 0.0;
                Double bobot = 0.0;
                if(!pengguna1.equals(pengguna2)){
                    for(DokumenTweet dk : listDok){
                        if(
(dk.getPengguna().toLowerCase().equals(pengguna1)&&dk.g
etPengguna1().toLowerCase().equals(pengguna2))||

(dk.getPengguna1().toLowerCase().equals(pengguna1)&&dk.
getPengguna().toLowerCase().equals(pengguna2))){
                            d1uniond2 = d1uniond2 + 1;}
                        }

if(dk.getPengguna().toLowerCase().equals(pengguna1)||

dk.getPengguna1().toLowerCase().equals(pengguna1)){
                            jumd1 = jumd1 + 1;
                        }

if(dk.getPengguna().toLowerCase().equals(pengguna2)||

dk.getPengguna1().toLowerCase().equals(pengguna2)){
                            jumd2 = jumd2 + 1;
```

```
if(dk.getPengguna().toLowerCase().equals(pengguna1)||
dk.getPengguna1().toLowerCase().equals(pengguna1)){
                              jumd1 = jumd1 + 1;
}if(dk.getPengguna().toLowerCase().equals(pengguna2)||
dk.getPengguna1().toLowerCase().equals(pengguna2)){
                              jumd2 = jumd2 + 1;}}
                  bobot = d1uniond2 / (jumd1 + jumd2);
                  if(d1uniond2 != 0.0){
                          snaAddVertex(pengguna1);
                          snaAddVertex(pengguna2);
snaAddEdge(edgelbl.toString(), pengguna1, pengguna2);
                          edgelbl += 1;}}}
              startidx = startidx + 1;}
        hitungFiturSNA();
        netStrenght = networkStrenght();
        return netStrenght;}

    public Double formClusteredPengguna(Clusters[] cl)
throws SQLException{
        Double Silhouette_Coeff = 0.0;
        for(int i = 0; i < cl.length; i++) {
            formInnerClusteredGraph(cl[i].clus);}
        for(int i = 0; i < cl.length-1; i++){
            for(int j = i+1; j < cl.length; j++){
formOutterClusteredGraph(cl[i].clus, cl[j].clus);}}
        System.out.println(g.getVertexCount());
        Silhouette_Coeff= countSilhouetteCoeff(cl);

        return Silhouette_Coeff;}

    private void
formInnerClusteredGraph(ArrayList<Point> listPoint1) {
        for(int i = 0; i < listPoint1.size(); i++){
    for(int j = i+1; j < listPoint1.size(); j++){
              g.addVertex(listPoint1.get(i).getTerm());

g.addVertex(listPoint1.get(j).getTerm());
                g.addEdge(edgelabel.toString(),
listPoint1.get(i).getTerm(),listPoint1.get(j).getTerm()
);
                edgelabel++;}}}
     private void
formOutterClusteredGraph(ArrayList<Point> listPoint1,
ArrayList<Point> listPoint2) throws SQLException {
        DokumenTweet dok = new DokumenTweet();
        LinkedList<DokumenTweet> listDok =
dok.readPartDokumen();
        for(int i = 0; i < listPoint1.size(); i++){
            for(int j = i+1; j < listPoint2.size();
j++){
```

```
                Double d1uniond2 = 0.0;
                Double jumd1 = 0.0;
                Double jumd2 = 0.0;
                Double bobot = 0.0;
                for(DokumenTweet dk : listDok){
                        if(
(dk.getPengguna().equals(listPoint1.get(i).getTerm())&&
dk.getPengguna1().equals(listPoint2.get(j).getTerm()))|
|

(dk.getPengguna1().equals(listPoint1.get(i).getTerm())&
&dk.getPengguna().equals(listPoint2.get(j).getTerm())))
{
d1uniond2 = d1uniond2 + 1; }
if(dk.getPengguna().equals(listPoint1.get(i).getTerm())
||
dk.getPengguna1().equals(listPoint1.get(i).getTerm())){
                            jumd1 = jumd1 + 1;}

if(dk.getPengguna().equals(listPoint2.get(j).getTerm())
||

dk.getPengguna1().equals(listPoint2.get(j).getTerm())){
                            jumd2 = jumd2 + 1;}}
                bobot = d1uniond2 / (jumd1 + jumd2);
                if(d1uniond2 != 0.0){
                        g.addEdge(edgelabel.toString(),
listPoint1.get(i).getTerm(),
listPoint2.get(j).getTerm());
                        edgelabel += 1;}}}}
    private void snaAddVertex(String s){
        g.addVertex(s);
    }

    private void snaAddEdge(String bobot, String node1,
String node2){

        g.addEdge(bobot, node1, node2);}
    public void drawGraph(){
        Layout<String, String> layout = new
ISOMLayout(g);
        layout.setSize(new Dimension(500, 500));

        VisualizationViewer<String, String> vs = new
VisualizationViewer<String, String>(layout);
        vs.setPreferredSize(new Dimension(1024, 768));

        Transformer<String, Paint> vertexPaint = new
Transformer<String, Paint>() {
```

```
            @Override
            public Paint transform(String i) {

                if(i.equals("A")){
                    return Color.RED;
                }
                else if(i.equals("B")){
                    return Color.BLUE;
                }
                else{
                    return Color.BLUE;}}};
        Transformer<String, Shape> vertexSize = new
Transformer<String, Shape>() {

            @Override
            public Shape transform(String i) {
                 Ellipse2D circle = new
Ellipse2D.Double(-15,-15,20,20);
//                  else return
AffineTransform.getScaleInstance(degCent.getVertexScore
(i)/2,
degCent.getVertexScore(i)/2).createTransformedShape(cir
cle);}};

        float dash[] = {10.0f};
        final Stroke edgeStroke = new BasicStroke(1.0f,
BasicStroke.CAP_BUTT, BasicStroke.JOIN_MITER, 10.0f,
dash, 0.0f);
        Transformer<String, Stroke>
edgeStrokerTransform = new Transformer<String,
Stroke>() {

            @Override
            public Stroke transform(String i) {
                return edgeStroke;}};
vs.getRenderContext().setVertexShapeTransformer(vertexS
ize);

vs.getRenderContext().setVertexFillPaintTransformer(ver
texPaint);

vs.getRenderContext().setVertexLabelTransformer(new
ToStringLabeller());

vs.getRenderContext().setEdgeLabelTransformer(new
ToStringLabeller());

vs.getRenderer().getVertexLabelRenderer().setPosition(R
enderer.VertexLabel.Position.CNTR);
```

```
DefaultModalGraphMouse gm = new
DefaultModalGraphMouse();
        gm.setMode(ModalGraphMouse.Mode.TRANSFORMING);
        vs.setGraphMouse(gm);
        JFrame frame = new JFrame("Social Network
Analysis");
        frame.getContentPane().add(vs);

frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE)
;
        frame.pack();
        frame.setVisible(true);}
    public String getNamaDosen(String i) {
                String nama = "";
                try {
                    System.out.println(i);
                    Database koneksi = new Database();
                    koneksi.connectFirst();
                    String query = "SELECT DISTINCT
nama "+ "FROM `data_tweets` WHERE username = \"" + i +
"\"";
                    ResultSet rs =
koneksi.executeSelect(query);

                    if(!rs.first()){
                        query = "SELECT DISTINCT
username1 "
                        + "FROM `data_tweets` WHERE
username1 = \"" + i + "\"";
                        rs =
koneksi.executeSelect(query); {
while(rs.next()){
nama = rs.getString("nama1");}}}
                    else{
                        while(rs.next()){
                            nama =
rs.getString("nama");}}
                } catch (SQLException ex) {

Logger.getLogger(SNA.class.getName()).log(Level.SEVERE,
null, ex);}
 return nama;
            }
public void hitungFiturSNA() throws SQLException{

        countDegreeCentrality();
        countClosenessCentrality();
        countBetweenessCentrality();

    }
```

```
       public ArrayList<String> networkStrenght(){

        ArrayList<String> netStrenght = new
ArrayList<>();
        Metrics metric = new Metrics();
        Double coef = 0.0;
        //System.out.println("Graph Clustering Corff =
" + metric.clusteringCoefficients(g));
        Map<String,Double> clustCoeffMap =
metric.clusteringCoefficients(g);
        for(String s : g.getVertices()){
            Double value = clustCoeffMap.get(s);
            coef += value;
        }


netStrenght.add(String.valueOf(coef/g.getVertexCount())
);
        //System.out.println("Graph Density = " +
(double)(2*g.getEdgeCount())/((double)g.getVertexCount(
)*(double)g.getVertexCount()-1));

netStrenght.add(String.valueOf((double)(2*g.getEdgeCoun
t())/((double)g.getVertexCount()*(double)g.getVertexCou
nt()-1)));
        Transformer<String, Double> dc =
DistanceStatistics.averageDistances(g, new
UnweightedShortestPath(g));
        Double distance = 0.0;
        for(String s : g.getVertices()){
           distance += dc.transform(s).doubleValue();

//System.out.println(dc.transform(s).doubleValue());
        }

netStrenght.add(String.valueOf(distance/g.getVertexCoun
t()));
        //System.out.println("Average Distance = " +
distance/g.getVertexCount());

        return netStrenght;
    }

    public void countDegreeCentrality() throws
SQLException{
        /*Menghitung degree centrality tiap vertex*/
        System.out.println("Degree Centrality");
        Database koneksi = new Database();
        koneksi.connectFirst();
        koneksi.executeUpdate("DELETE FROM
```

```
`nodesnafitur`");
        int n = g.getVertexCount();
        DegreeScorer degCent = new DegreeScorer(g);
        double[] degValues = new double[n];
        int i = 0;
        for(String vertex : g.getVertices() ){
            degValues[i++] =
(double)degCent.getVertexScore(vertex)/(double)n;
            koneksi.executeUpdate("INSERT INTO
`nodesnafitur`(`id_node`, `degCent`, `betCent`,
`closeCent`) "
                    + "VALUES
(\""+vertex+"\","+((double)degCent.getVertexScore(verte
x)/(double)n)+","+0.0+","+0.0+")");
        }
        koneksi.destroyConnection();

    }

    public void countClosenessCentrality() throws
SQLException{
        /*Menghitung closeness tiap node*/
        System.out.println("Closeness Centrality");
        Database koneksi = new Database();
        koneksi.connectFirst();
        int n = g.getVertexCount();
        System.out.println(n);
        double norm = n - 1;
        ClosenessCentrality closeCent = new
ClosenessCentrality(g);
        double[] closeValues = new double[n];


        int i = 0;
        for(String vertex : g.getVertices()){
            closeValues[i++] =
closeCent.getVertexScore(vertex);
            koneksi.executeUpdate("UPDATE
`nodesnafitur` "
                    + "SET
`closeCent`="+closeCent.getVertexScore(vertex)+" "
                    + "WHERE `id_node` =
\""+vertex+"\"");
        }
        koneksi.destroyConnection();
    }

    public void countBetweenessCentrality() throws
SQLException{
        /*Menghitung betweenes tiap node*/
```

```
        System.out.println("Betweeness Centrality");
        Database koneksi = new Database();
        koneksi.connectFirst();
        BetweennessCentrality betweenCent = new
BetweennessCentrality(g);

betweenCent.setRemoveRankScoresOnFinalize(false);
        betweenCent.evaluate();
        double normVal = ((g.getVertexCount()-
1)*(g.getVertexCount()-2))/2;
        for(String vertex : g.getVertices()){
            koneksi.executeUpdate("UPDATE
`nodesnafitur` "
                    + "SET
`betCent`="+betweenCent.getVertexRankScore(vertex)/norm
Val+" "
                    + "WHERE `id_node` =
\""+vertex+"\"");
        }
        koneksi.destroyConnection();
    }

    public void createJsonFile() throws
FileNotFoundException{
        int i = 1;
        StringBuilder jsonStr = new StringBuilder();

        System.out.println(g.getVertexCount());
        jsonStr.append("[\n");
        for(String vertex : g.getVertices()){
            if(i != g.getVertexCount()){
                jsonStr.append("{\"id\": \"" + vertex
+"\", \"label\": \""+ vertex +"\"},\n");
            }
            else{
                jsonStr.append("{\"id\": \"" + vertex
+"\", \"label\": \""+ vertex +"\"}\n");
            }
            i++;
        }
        jsonStr.append("]");
        //System.out.println(jsonStr.toString());
        PrintWriter pw = new
PrintWriter("C:\\Users\\Asus\\Documents\\NetBeansProjec
ts\\yakinbisa\\src\\yakinbisa\\TryJson.json");
        pw.println(jsonStr.toString());
        pw.close();

        g.getEdges();
    }
```

```
   public Double countAvgSim2() throws SQLException{

        ResultSet dokRepRS;
        ResultSet clustSetRS;
        Database koneksi = new Database();
        koneksi.connectFirst();
        ResultSet jd = koneksi.executeSelect("SELECT
COUNT(*)\n" +
                                    "FROM(\n" +
                                    "SELECT DISTINCT
id_data FROM data_tweets WHERE LIMIT 500\n" +
                                    ")as temp");
        int jumDokumen = 0;
        while(jd.next()){
           jumDokumen = jd.getInt(1);
        }


            ResultSet clustIdRS =
koneksi.executeSelect("SELECT DISTINCT `id_cluster`
FROM topikcluster ORDER BY `id_cluster` ASC ");
            Double sumBobotCluster = 0.0;
            while(clustIdRS.next()){

                Double bobotCluster = 0.0;

                Bag<String> termInCluster = new
TreeBag<>();
                int idCluster = clustIdRS.getInt(1);
                ResultSet temp2 =
koneksi.executeSelect("SELECT `term` FROM
`topikcluster` "
                     + "WHERE `id_cluster` = " +
idCluster);

                while(temp2.next()){

termInCluster.add(temp2.getString("term"));
                }
                ResultSet dokIdRS =
koneksi.executeSelect("SELECT DISTINCT `id_dok` FROM
representasi_dok_rake");

                while(dokIdRS.next()){
                    Bag<String> listRepDok = new
TreeBag<>();
                    ArrayList<String> listFrase = new
ArrayList<>();
                    listFrase.addAll(termInCluster);
```

```
   public Double countAvgSim2() throws SQLException{

        ResultSet dokRepRS;
        ResultSet clustSetRS;
        Database koneksi = new Database();
        koneksi.connectFirst();
        ResultSet jd = koneksi.executeSelect("SELECT
COUNT(*)\n" +
                                    "FROM(\n" +
                                    "SELECT DISTINCT
id_data FROM data_tweets WHERE LIMIT 500\n" +
                                    ")as temp");
        int jumDokumen = 0;
        while(jd.next()){
           jumDokumen = jd.getInt(1);
        }


            ResultSet clustIdRS =
koneksi.executeSelect("SELECT DISTINCT `id_cluster`
FROM topikcluster ORDER BY `id_cluster` ASC ");
            Double sumBobotCluster = 0.0;
            while(clustIdRS.next()){

                Double bobotCluster = 0.0;

                Bag<String> termInCluster = new
TreeBag<>();
                int idCluster = clustIdRS.getInt(1);
                ResultSet temp2 =
koneksi.executeSelect("SELECT `term` FROM
`topikcluster` "
                     + "WHERE `id_cluster` = " +
idCluster);

                while(temp2.next()){

termInCluster.add(temp2.getString("term"));
                }
                ResultSet dokIdRS =
koneksi.executeSelect("SELECT DISTINCT `id_dok` FROM
representasi_dok_rake");

                while(dokIdRS.next()){
                    Bag<String> listRepDok = new
TreeBag<>();
                    ArrayList<String> listFrase = new
ArrayList<>();
                    listFrase.addAll(termInCluster);
                    String idDok =
```

```
dokIdRS.getString(1);
                    ResultSet temp =
koneksi.executeSelect("SELECT `frase_rep` FROM
`representasi_dok_rake` "
                                        + "WHERE `id_dok`
= \""+ idDok +"\"");
                    while(temp.next()){

listRepDok.add(temp.getString("frase_rep"));

if(!listFrase.contains(temp.getString("frase_rep"))){

listFrase.add(temp.getString("frase_rep"));
                        }
                    }
                    Double cosine =
cosineSim(listFrase, listRepDok, termInCluster);
                    bobotCluster += cosine;

                }
                bobotCluster = bobotCluster/jumDokumen;
                sumBobotCluster += bobotCluster;
            }
            sumBobotCluster = sumBobotCluster;
            return sumBobotCluster;
   }

   //menghitung evaluasi dgn menggunakan Silhouette
Coefficien
    public Double countSilhouetteCoeff(Clusters[] cl) {

        Double SillhouetteCoeff = 0.0;
        Double SillhouetteX = 0.0;
        int jumData = 0;
        for(int i = 0; i < cl.length; i ++){
            jumData += cl[i].clus.size();
        }


        for(int i = 0; i < cl.length; i++){
            for(int j = 0; j < cl[i].clus.size(); j++){
                Double aX = countAX(cl[i].clus,
cl[i].clus.get(j));
                Double bX = countBX(i,
cl[i].clus.get(j), cl);
                SillhouetteX = (bX - aX) / Math.max(aX,
bX);
                SillhouetteCoeff += SillhouetteX;
            }
        }
```

```
        SillhouetteCoeff = SillhouetteCoeff/jumData;

        return SillhouetteCoeff;
    }

    private Double countAX(ArrayList<Point> clus, Point
get) {
        Double aX = 0.0;
        for(Point p : clus){
            aX += distance(p, get);
        }
        aX = aX / clus.size();
        return aX;
    }

    private Double countBX(int idx, Point get,
Clusters[] cl) {
        Double minbX = Double.MAX_VALUE;
        Double avgClusterDist = 0.0;
        for(int i = 0; i < cl.length; i++){
            if(i != idx){
                for(int j = 0; j < cl[i].clus.size(); j
++){
                    Double dX =
distance(cl[i].clus.get(j), get);
                    avgClusterDist += dX;
                }
                avgClusterDist =
avgClusterDist/cl[i].clus.size();
                if(avgClusterDist < minbX){
                    minbX = avgClusterDist;
                }
            }
        }
        return minbX;
    }

    public Double distance(Point x, Point y) {
        Double distance = 0.0;
        for(int i = 0; i < x.getVektor().size(); i++){
            distance += Math.pow(x.getVektor().get(i) -
y.getVektor().get(i), 2);
        }
        distance = Math.sqrt(distance);
        return distance;
    }

    private Double cosineSim(ArrayList<String>
listFrase, Bag<String> listRepDok, Bag<String>
termInCluster) {
```

```
        Double cosine = 0.0;
        ArrayList<Integer> dok = new ArrayList<>();
        ArrayList<Integer> cluster = new ArrayList<>();
        double top = 0.0;
        double sumDok = 0.0;
        double sumCluster = 0.0;

        for(String s : listFrase){
            dok.add(listRepDok.getCount(s));
            cluster.add(termInCluster.getCount(s));
        }

        for(int i = 0; i < listFrase.size(); i++){
            sumDok += Math.pow(dok.get(i),2);
            sumCluster += Math.pow(cluster.get(i),2);
            top += dok.get(i)*cluster.get(i);
        }

        cosine =
top/(Math.sqrt(sumDok)*Math.sqrt(sumCluster));

        return cosine;
    }

}
```

"Halaman ini sengaja dikosongkan"

BIODATA PENULIS

Penulis bernama lengkap **Siti Nur Diana**, lahir di Mojokerto, 27 Oktober 1994. Anak ketiga dari pasangan Makruf dan Siti Masfufah, serta memiliki kakak perempuan Siti Masrifah dan kakak laki-laki Mukhammad Sofwan. Penulis mengikuti pendidikan dasar dari Madrasah Ibtidaiyah hingga Sekolah Menegah Atas di Kota Mojokerto. Penulis menempuh pendidikan di MI Miftahul Huda Sumolawang, SMP Negeri 1 Mojoanyar, dan SMA Negeri 1 Puri. Setelah Lulus dari SMAN 1 Puri pada tahun 2013 yang lalu, penulis melanjutkan pendidikan tingginya di Institut Teknologi Sepeluh Nopember (ITS) Surabaya dengan mengambil Departemen Matematika dengan bidang minat Ilmu Komputer. Selama mengikuti perkuliahan di ITS, penulis turut aktif dalam beberapa kegiatan kemahasiswaan sebagai Sekretaris II Himatika ITS periode 2014/2016, dan Sekretaris Umum Himatika ITS Periode 2015/2016. Selain aktif dalam beberapa kegiatan kemahasiswaan, penulis juga mengikuti Kerja Praktek PT.Dutacipta Konsultama Surabaya dan mengerjakan *project* Web Aplikasi *Freshoes*. Informasi lebih lanjut mengenai Tugas Akhir ini dapat ditujukan ke penulis melalui email: sitinurdianamatematika@gmail.com.