TUGAS AKHIR – KI141502

IMPLEMENTASI PERINGKASAN MULTI-DOKUMEN BERITA BERBAHASA INDONESIA DENGAN PEMILIHAN KATA KUNCI TWITTER MENGGUNAKAN AUTOCORRELATION WAVELET COEFFICIENTS

OSHI PRAHTIWI GUSMAN
NRP 5112 100 195

Dosen Pembimbing I
Diana Purwitasari, S.Kom., M.Sc.

Dosen Pembimbing II
Dr. Eng. Chastine Fatichah, S.Kom., M.Kom.

Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2016

UNDERGRADUATE THESES – KI141502

IMPLEMENTATION OF MULTI-DOCUMENTS SUMMARIZATION NEWS IN BAHASA WITH KEYWORDS SELECTION IN TWITTER USING AUTOCORRELATION WAVELET COEFFICIENTS

OSHI PRAHTIWI GUSMAN
NRP 5112 100 195

First Advisor
Diana Purwitasari, S.Kom., M.Sc.

Second Advisor
Dr. Eng. Chastine Fatichah, S.Kom., M.Kom.

Department of Informatics
Faculty of Information Technology
Sepuluh Nopember Institute of Technology
Surabaya 2016

LEMBAR PENGESAHAN

IMPLEMENTASI PERINGKASAN MULTI-DOKUMEN BERITA BERBAHASA INDONESIA DENGAN PEMILIHAN KATA KUNCI TWITTER MENGGUNAKAN *AUTOCORRELATION WAVELET COEFFICIENTS*

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Komputasi Cerdas dan Visi
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh:
OSHI PRAHTIWI GUSMAN
NRP: 5112100195

Disetujui oleh Pembimbing Tugas Akhir:

Diana Purwitasari, S.Kom., M.Sc. .....
(NIP. *197804102003122001*) (Pembimbing 1)

Dr. Eng. Chastine Fatichah, S.Kom., M.Kom. .....
(NIP. *197512202001122002*) (Pembimbing 2)

SURABAYA
JUNI, 2016

IMPLEMENTASI PERINGKASAN MULTI-DOKUMEN BERITA BERBAHASA INDONESIA DENGA PEMILIHAN KATA KUNCI TWITTER MENGGUNAKAN AUTOCORRELATION WAVELET COEFFICIENTS

Nama Mahasiswa : OSHI PRAHTIWI GUSMAN
NRP : 5112100195
Jurusan : Teknik Informatika FTIF-ITS
Dosen Pembimbing 1 : Diana Purwitasari, S.Kom., M.Sc.
Dosen Pembimbing 2 : Dr. Eng. Chastine Fatichah, S.Kom., M.Kom.

ABSTRAK

Twitter digunakan untuk menyampaikan informasi berupa tweet yang merepresentasikan suatu kejadian yang mengakibatkan munculnya issue. Issue yang paling sering dibahas disebut dengan Trending Issue. Dalam tugas akhir ini, digunakan data twitter yang telah diambil selama bulan April dan Mei 2016

Metode Autocorrelation Wavelet coefficients berguna untuk mendapatkan kata kunci yang muncul secara periodik atau berulang (trivial) yang merepresentasikan kejadian biasa dan akan dieliminasi sehingga menyisakan kata kunci penting (non-trivial). Kata kunci penting digunakan untuk peringkasan berita menggunakan metode pembobotan kalimat berdasarkan trending issue sehingga menghasilkan ringkasan berita yang koheren.

Setelah dilakukan pengujian ada beberapa faktor utama yang mempengaruhi hasil ringkasan berita, diantaranya penggunaan keyword yang spesifik, jangkauan lokasi pengambilan tweet, penentuan confidence boundary, dan perlu atau tidaknya proses eliminasi kata kunci trival. Nilai silhouette terbaik ditunjukkan pada hasil ekstraksi trending issue dengan pembuangan kata kunci trivial sebesar 0,36322. Nilai rouge terbaik ditunjukkan pada hasil ringkasan tanpa pembuangan kata kunci sebesar 0,30199.

Kata kunci: Twitter, Wavelet Coefficients, Autocorrelations, Trending Issue

IMPLEMENTATION OF MULTI-DOCUMENTS SUMMARIZATION NEWS IN BAHASA WITH KEYWORDS SELECTION IN TWITTER USING AUTOCORRELATION WAVELET COEFFICIENTS

Nama Mahasiswa	**: OSHI PRAHTIWI GUSMAN**
NRP	**: 5112100195**
Jurusan	**: Teknik Informatika FTIF-ITS**
Dosen Pembimbing 1	**: Diana Purwitasari, S.Kom., M.Sc.**
Dosen Pembimbing 2	**: Dr. Eng. Chastine Fatichah, S.Kom., M.Kom.**

ABSTRACT

Twitter is used to deliver the information in the form of tweets that represents an event that resulted an issue. The most frequently discussed issue is called Trending Issue. In this Final Project, the twitter data is collected during April and May 2016.

Autocorrelation wavelet coefficients method is used to get the keywords that appear periodically. The repeated keywords (trivial) represent a regular event and will be eliminated thus leaving important keywords (non-trivial). Important keywords will be used to summarize news and become the purpose of this Final Project, using the phrase weighting method based trending issue to produce a coherent summary of the news.

After testing, a number of key factors that influence the outcome of a news summary, including the use of a specific keyword, scope of the location of the tweet, determination of the confidence boundary, and whether or not the elimination process trival keywords. Best silhouette value shown in the results of extraction trending issue with the disposal amounting to 0.36322 trivial keywords. Best rouge value shown in the summary results without disposal keywords by 0.30199.

Keywords : Twitter, Wavelet Coefficients, Autocorrelations, Trending Issue

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR KODE SUMBER

BAB I
PENDAHULUAN

1.1 Latar Belakang

Ringkasan berita dapat di artikan sebagai sebuah teks yang dihasilkan dari salah satu atau lebih kalimat yang mampu menyampaikan informasi penting dari sebuah berita. Dimana panjang dari sebuah ringkasan tidak lebih dari setengah panjang dokumen asli, dan biasanya lebih pendek [1]. Isi berita harus dibaca terlebih dahulu untuk mengetahui isu yang terjadi sebagai informasi penting. Hal ini memerlukan waktu yang cukup lama sehingga dibutuhkan suatu ringkasan berita yang dapat mempresentasikan isu dari berita tersebut.

Media sosial seperti Twitter merupakan salah satu media yang digunakan untuk menyampaikan berita. Berita yang disajikan oleh Twitter berupa pesan singkat yang disebut dengan *tweets* dimana *issue* dipresentasikan dengan kata kunci yang muncul pada kelompok *tweets*. *Issue* yang paling sering dibahas dan sering muncul disebut dengan *Trending Issue* [2]. Pemanfaatan data dari Twitter dapat dilakukan tidak hanya membuat ringkasan *tweet* tetapi juga ringkasan berita dengan memilih satu kalimat yang paling representatif.

Wavelet autocorrelation merupakan metode dengan melakukan pencarian kejadian berulang dengan mencari korelasi antar *coefficient (autocorrelation)* sehingga dapat dideteksi kemunculan kejadian yang berulang secara periodik (*wavelet*) pada kemunculan kata kunci [3]. Metode ini digunakan untuk mendapatkan kata kunci penting *(non-trivial)* dari data Twitter. Pemilihan kata kunci yang dianggap sebagai representasi kejadian biasa *(trivial)* atau berulang secara periodik akan dieliminasi [3] sehingga diharapkan akan mendapatkan kata kunci penting (*non-trivial*) yang dapat mempresentasikan informasi yang terkandung dari data Twitter. Hal ini sesuai dengan penelitian yang dilakukan oleh Rizal Perdana Setia dengan judul 'Pemilihan Kata Kunci

untuk Deteksi Kejadian Trivial Menggunakan *Autocorrelation Wavelet Coefficients* pada Peringkasan Dokumen Twitter'.

Pemilihan kata kunci digunakan untuk peringkasan berita dengan menggunakan metode pembobotan kalimat berdasarkan *Trending issue* dengan tetap mempertimbangkan informasi penting dari berita. Tujuannya agar dapat menyeleksi kalimat penting dari berita secara lebih tepat sesuai dengan *Trending issue* sehingga mampu menghasilkan berita yang lebih koheren [2]. Hal ini berdasarkan penelitian yang dilakukan oleh Nur Hayatin dengan judul 'Penentuan *Trending Issue* data Twitter menggunakan *Cluster Importance* untuk Peringkasan Multi Dokumen Berita'.

Tugas Akhir ini bertujuan untuk mengetahui penggunaan metode *autocorrelation wavelet coefficients* dengan pemilihan kata kunci Twitter dengan melakukan ekstraksi *trending issue* sehingga menghasilkan suatu kata penting (*non trivial)* yang dapat digunakan sebagai acuan untuk peringkasan dokumen berita. Hal ini dilakukan dengan analisis terhadap hasil pengujian yang didapat berdasarkan nilai *silhouette* sebagai evaluasi *clustering* untuk ekstraksi *trending issue* dan nilai *rouge* sebagai evaluasi hasil ringkasan. Berdasarkan hasil analisis tersebut diketahui bahwa penggunaan metode *autocorrelation wavelet coefficients* dapat digunakan dengan penggunaan *keyword* yang lebih umum ketika pengambilan data Twitter dan penentuan lokasi geografis yang lebih kecil pada saat pengambilan data Twiter.

1.2 Rumusan Masalah

Tugas Akhir ini mengangkat beberapa rumusan masalah sebagai berikut:

1. Bagaimana cara pengambilan data Twitter secara periodik yang digunakan untuk mendapatkan kata kunci Twitter dengan menggunakan *Autocorrelation Wavelet Coefficients*?
2. Bagaimana mendapatkan *Trending issue* dari kata kunci penting (*non-trivial)* hasil ekstraksi *tweets* dari data Twitter?

3. Bagaimana menghasilkan ringkasan berita dengan kata kunci *non trivial* menggunakan *Autocorrelation Wavelet Coefficients* berdasarkan *Trending issue*?

1.3 Batasan Permasalahan

Permasalahan yang dibahas pada Tugas Akhir ini memiliki batasan sebagai berikut:

1. Data yang dijadikan data uji adalah dokumen berita dan tweets berbahasa Indonesia yang dikumpulkan melalui API stream Twitter
2. Kejadian yang dianggap trivial adalah kejadian yang dibentuk oleh kata kunci yang berulang secara periodik
3. Data Twitter dan berita yang diambil berdasarkan periode waktu dan isu yang sama.
4. *Mother wavelet* yang digunakan berjenis *Coifman* atau yang biasa disebut *coiflet* dengan dekomposisi level 1 (satu).
5. Nilai *confidence boundary* yang ditetapkan berdasarkan pengamatan manual hasil analisa yang dilakukan sesuai dengan nilai batas kepercayaan pada *correlogram.*
6. Metode *clustering* yang digunakan adalah K-medoids untuk ekstraksi *trending issue* dengan metode evaluasi yang digunakan berdasarkan nilai *silhouette.*
7. Penyusunan ringkasan tidak mempertimbangkan urutan kalimat berdasarkan sistematika penulisan.
8. Jenis peringkasan dalam penelitian ini adalah ekstraktif untuk input multi berita.
9. Ringkasan yang dihasilkan berupa kumpulan kalimat, bukan berformat paragraf.

1.4 Tujuan

Tujuan dari Tugas Akhir ini antara lain mengimplementasikan metode *Autocorrelation Wavelet Coefficients* untuk mendapatkan kata kunci yang sesuai dari data Twitter berbasis wavelet sehingga dari kata kunci tersebut dapat dihasilkan ringkasan berita yang sesuai dengan *trending issue.*

1.5 Manfaat

Manfaat dari hasil pembuatan Tugas Akhir ini antara lain untuk mendapatkan informasi secara cepat dan tepat sesuai dengan *trending issue* pada berita tanpa harus membaca isi berita secara keseluruhan dengan kata kunci yang dihasilkan yang menggunakan *Autocorrelation Wavelet Coefficient.*

1.6 Metodologi

Pembuatan Tugas Akhir ini dilakukan dengan menggunakan metodologi sebagai berikut:

1.6.1 Penyusunan Proposal

Tahap awal Tugas Akhir ini adalah menyusun proposal Tugas Akhir. Pada proposal, diajukan gagasan untuk mengimplementasikan peringkasan multi-dokumen berita berbahasa Indonesia menggunakan *Autocorrelation Wavelet Coefficients* sebagai penentuan kata kunci penting (*non trivial)*, algoritma *K-Medoids* sebagai metode klasterisasi untuk mengelompokkan kata secara otomatis dan teknik pembobotan kalimat untuk peringkasan berita.

1.6.2 Studi Literatur

Pada studi literatur ini diharapkan dapat memberikan data, informasi, dan fakta mengenai peringkasan berita dengan pemilihan kata kunci menggunakan *Autocorrelation wavelet coeficients* yang akan diimplementasikan dan *library* yang digunakan. Studi literatur yang dilakukan mencakup pencarian dan mempelajari referensi-referensi yang terkait, seperti :

1. *Text preprocessing* yaitu *segmentation tweet*, *stopword removal* dan *stemming* dalam Bahasa Indonesia.
2. Metode transformasi *Wavelet* dan *Autocorrelation Coefficient Function* menggunakan *library pywt* dan *statsmodels.tsa.stattools.acf* yang terdapat pada Python

3. Ektraksi *Trending issue*.
4. Metode penghitungan jarak antar kata, *cosine similarity*
5. Ekstraksi fitur berita dari *tweet* berdasarkan fitur kata kunci klaster *tweet* berdasarkan hasil pembobotan.
6. Metode evaluasi hasil klaster dan hasil ringkasan berita.

1.6.3 Analisis dan Desain Perangkat Lunak

Sistem ini dibangun dengan menggunakan menggunakan bahasa pemograman Python. Data Twitter didapatkan pada API Streaming Twitter dengan metode Firehouse yaitu mengambil *stream* data *tweet* dengan batasan geografis tertentu. Sistem ini memiliki beberapa fase antara lain :

1. Fase transformasi wavelet, dimana setelah melalui proses *text processing,* data disimpan kedalam database. Data-data tersebut selanjutnya akan di proses untuk mendapatkan kata kunci penting (*non trivial)* menggunakan *Autocorrelation Wavelet Coefficients*.
2. Fase ekstraksi *trending issue,* yaitu pada tahap ini akan dilakukan pencarian isu yang sedang berkembang berdasarkan kata kunci penting yang sudah didapat.
3. Fase peringkasan berita, dimana pada tahap ini akan dilakukan penghitungan total bobot kalimat dan penyusunan ringkasan.

1.6.4 Pengujian dan Evaluasi

Pada tahap ini dilakukan pengujian dari kumpulan-kumpulan langkah-langkah dari metode yang akan diimplementasikan. Pengujian yang akan dilakukan sebagai berikut :

a. Pemilihan kata kunci biasa (*trivial)*
   - Penentuan nilai *confidence boundary* yaitu nilai batas atas dan bawah koefisien pada *correlogram* yang

menentukan sebuah *wavelet* dianggap memiliki tingkat keperiodikan tinggi atau rendah

- Penetuan nilai minimum kata kunci yang dianggap sebagai kata kunci *trivial* yaitu nilai batas bawah jumalh koefisien yang berada diatas nilai *coefficience boundary* yang dianggap sebagai kata kunci *trivial*

b. Pengujian kata kunci dengan *trending issue*
Penetuan jumlah *cluster (k)* yang digunakan dalam proses pengelompokan *tweets.*

c. Pengukuran kualitas klaster dan ringkasan yang dihasilkan berdasarkan *silhouette coefficient* dan *rouge* yang membandingkan dengan *Groundruth* atau proses manual

1.6.5 Penyusunan Buku

Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam tugas akhir ini serta hasil dari implementasi aplikasi perangkat lunak yang telah dibuat.

1.7 Sistematika Penulisan Laporan

Sistematika penulisan laporan Tugas Akhir adalah sebagai berikut:

Bab I Pendahuluan
Bab ini berisikan penjelasan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika penulisan dari pembuatan Tugas Akhir.

Bab II Tinjauan Pustaka
Bab ini berisi kajian teori dari metode dan algoritma yang digunakan dalam penyusunan Tugas Akhir ini. Secara garis besar, bab ini berisi tentang pengambilan data *tweets* menggunakan Twitter API, *text processing*,

Wavelet Autocorrelation, K-Medoids Clustering, teknik pembobotan kalimat dan sebagainya.

Bab III Analisis dan Perancangan Sistem

Bab ini berisi pembahasan mengena analisis dan perancangan sistem. Perancangan sistem meliputi analisis dan metode secara umum, data pengujian, dan perancangan antarmuka pada aplikasi.

Bab IV Implementasi

Bab ini menjelaskan implementasi dari perancangan perangkat lunak.

Bab V Hasil Uji Coba dan Evaluasi

Bab ini berisikan hasil uji coba dari metode *Autocorrelation Wavelet Coefficients* berdasarkan kata kunci *trivial*.

Bab VI Kesimpulan dan Saran

Bab ini merupakan bab yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan, masalah-masalah yang dialami pada proses pengerjaan Tugas Akhir, dan saran untuk pengembangan solusi ke depannya.

Daftar Pustaka

Bab ini berisi daftar pustaka yang dijadikan literatur dalam Tugas Akhir.

Lampiran

Dalam lampiran terdapat tabel-tabel data hasil uji coba dan kode sumber program secara keseluruhan.

BAB II
TINJAUAN PUSTAKA

Bab ini berisi pembahasan mengenai teori-teori dasar yang digunakan dalam Tugas Akhir. Teori-teori tersebut diantaranya adalah Secara garis besar, bab ini berisi tentang pengambilan data *tweets* menggunakan Twitter API, *text processing*, ekstraksi kata kunci *non trivial* menggunakan *Wavelet Autocorrelation, K-Medoids Clustering* dan *Cluster Importance* untuk ekstraksi *trending issue* dan teknik pembobotan kalimat untuk peringkasan dokumen dan metode evaluasi *sillhouette coefficients* untuk mengukur kualitas klaster dan metode evaluasi *Rouge* untuk mengukur hasil ringkasan berita.

2.1 Pengambilan Data *Tweets* Menggunakan Twitter API

Twitter adalah *social network* yang telah mendapatkan popularitas di seluruh dunia dengan total pengguna aktif lebih dari 500 juta sampai dengan tahun 2012 dan menghasilkan lebih dari 340 juta *tweets* per-hari [4]. *Tweets* merupakan pesan berupa teks singkat yang dapat ditulis hingga 140 karakter pada Twitter. Twitter menyediakan API (*Application Programming Interface*) yang dapat digunakan untuk mengambil data dari Twitter. Twitter API ini dapat memberikan kemudahan dalam mengoleksi dan mengumpulkan *tweets*. Pengguna Twitter API harus mendaftar menjadi developer Twitter terlebih dahulu pada situs https://dev.twitter.com/ dengan *sign in* menggunakan akun Twitter. Pada saat mendaftar, pengguna akan mendapatkan 4 buah key yaitu *customer key, consumer secret, acces token,* dan *access token* secret sebagai syarat *authentication* untuk mengakses data yang dimilik oleh Twitter.

Pada Tugas Akhir ini, proses pengambilan data atau *crawling* menggunakan *Library Twitter* untuk bahasa pemograman Python yang sudah disediakan oleh Twitter API. Data yang diambil adalah data *tweet* berserta informasi pemilik akun (*username*) dan waktu kemunculan. *Tweet* yang diambil berdasarkan wilayah

geografis Indoneisa pada periode waktu yang tertentu secara berkelanjutan atau terus menerus.

2.2 Text Prepocessing

Text preprocessing adalah tahap pertama yang dilakukan sebelum data di proses untuk dibentuk menjadi data frekuensi dan sinyal *Wavelet* [3]. Data yang digunakan adalah data Twitter yang memanfaatkan data teks dan waktu pengiriman. *Teks* preprocessing yang dilakukan adalah sebagai berikut :

- **Segmentasi**
  Segmentasi merupakan proses pengambilan kata-kata dari sebuah dokumen yang dipisahkan berdasarkan spasi dan tanda baca. Pada Tugas Akhir ini sebuah *tweet* dianggap sebagai sebuah dokumen dan kata – kata yang membangun *tweet* tersebut akan dipisahkan dan disimpan dalam sebuah *array*.
- ***Stopword Removal***
  Pada proses ini akan menghilangkan kata – kata yang dianggap tidak penting tetapi tetap tidak menghilangkan makna dari *tweet* tersebut contohnya seperti kata hubung yaitu “di”, “ke”, “dan”, dan lain - lain. Selain itu pembuangan terhadap URL yang sering muncul pada *tweet*. Pada Tugas Akhir ini digunakan *stopword list* atau daftar kata yang berisi kumpulan *stopword* dalam bentuk *file* berupa kamus kata.
- ***Stemming***
  Setelah kata yang dianggap tidak penting dibuang, maka proses selanjutnnya adalah mendapatkan kata dasar dari kata tersebut. Proses *stemming* dilakukan berdasarkan aturan-aturan yang terdapat dalam penulisan kata Bahasa Indonesia, seperti kata “mencari” menjadi “cari’. Algoritma *stemming* Bahasa Indonesia yang digunakan adalah algoritma yang ada pada penelitian [5].

2.3 Metode Transformasi *Wavelet*

Metode Transformasi *Wavelet* dapat digunakan untuk menyaring data, menghilangkan sinyal-sinyal yang tidak diinginkan serta mendeteksi kejadian-kejadian tertentu pada sinyal [6]. Transformasi wavelet dapat menganalisis sinyal *non – stasioner* yaitu sinyal yang kandungan frekuensinya bervariasi terhadap waktu. Data Twitter yang akan diproses adalah frekuensi kemunculan suatu kata yang tidak konstan atau *non-stasioner* sehingga data yang digunakan dapat sesuai dengan penggunaan *Wavelet*.

Transformasi *Wavelet* merupakan transformasi yang menggunakan kernel terintegrasi yang dinamakan wavelet. Wavelet bersifat *time-frequency localization* sehingga analisis menggunakan wavelet dapat mempelajari karakteristik sinyal secara lokal dan detail sesuai dengan skalanya, dimana hal ini menunjukkan kegunaan dari analisis pada sinyal *non – stationer,* karakteristik berbeda pada skala yang berbeda. Transformasi wavelet merubah sinyal dari *time-domain* menjadi *time-scale domain.*

Proses pemecahan sinyal akan menghasilkan *wavelet coefficients* dan himpunan basis fungsi. Himpunan basis fungsi disebut sebagai *wavelet family* terbentuk dari proses *scaling* dan *translating* dari *mother wavelet* $\Psi(t)$. *Mother wavelet* $\Psi(t)$ terdiri dari beberapa jenis yaitu *Haar, Daubechies, Coiflet* dan lainnya.

Transformasi *Wavelet* dibagi menjadi dua bagian, yaitu *continuous wavelet transformation (CWT)* dan *discrete wavelet transformation (DWT).* Secara umum, pada proses analisis CWT menghasilkan representasi sinyal yang berulang atau *redudant*. Selain itu CWT apabila dilakukan pemrosesan atau transformasi secara langsung membutuhkan waktu yang cukup lama. Sedangkan DWT, proses yang dilakukan akan menghasilkan representasi sinyal yang tidak berulang atau *non-redudant* dan tidak membutuhkan waktu yang banyak ketika melakukan transformasi.

Pada Tugas Akhir ini digunakan DWT sebagai pemroses wavelet karena data frekuensi Twitter berupa data diskrit. Data Twitter yang akan diproses adalah data frekuensi kata kunci yang dihasilkan pada tahap preproses dan diurutkan berdasarkan frekuensi kemunculan kata yang muncul tidak konstan atau *non-stasioner* sehingga penggunaan *wavelet* sesuai dengan data Twitter yang digunakan.

Proses pertama yang dilakukan adalah melakukan pembentukan sinyal yang berasal dari frekuensi data terhadap waktu. Dwt akan mengubah sinyal sumber menjadi dua klasifikasi sinyal yaitu frekuensi tinggi dengan resolusi waktu yang rendah serta frekuensi rendah dengan resolusi waktu yang tinggi. Proses pemecahan sinyal dalam DWT akan membagi sinyal menjadi dua bagian dengan jumlah *sampling* yang sama dengan menggunakan dua jenis *filter* yaitu *highpass filter* dan *lowpass filter.*

Proses selanjutanya adalah melakukan dekomposisi sinyal yaitu merubah data frekuensi kemunculan kata menggunakan jenis *mother wavelet Coifman* atau biasa disebut *coiflet.* Seluruh proses dalam satu level dekomposisi akan menghasilkan *wavelet coefficients* dengan membagi menjadi dua bentuk sinyal yang disebut *approximation coefficient* dari hasil *lowpass filter* dan *detail coefficient* dari hasil *highpass filter.* Ketika proses dekomposisi dilangsungkan kembali maka sinyal *approximation coefficient* akan menjadi *mother wavelet* dan terdekomposisi berdasarkan *highpass* dan *lowpass filter*. Dekomposisi secara berulang akan menentukan level DWT.

Misalkan terdapat data frekuensi kemunculan kata “makan” terhadap waktu pengambilan seperti Tabel 2.1. Tabel ini menunjukkan kemunculan kata “makan” pada interval waktu yang sudah ditentukan dimana pada data ini interval waktu yang digunakan adalah per 60 menit. Contoh data pada tanggal 8 Mei 2016 interval waktu pertama yaitu pukul 00.00, kata “makan” muncul sebanyak 186 kali. Sedangkan interval waktu kedua yaitu pukul 01.00 kata “makan” muncul sebanyak 85 kali.

Tabel 2.1 Data Frekuensi Kemunculan kata "makan"

No	Interval Waktu	Term Frekuensi	No	Interval Waktu	Term Frekuensi
1.	2016-05-08 00:00	186	13.	2016-05-08 13:00	204
2.	2016-05-08 01:00	85	14.	2016-05-08 14:00	201
3.	2016-05-08 02:00	53	15.	2016-05-08 15:00	166
4.	2016-05-08 03:00	26	16.	2016-05-08 16:00	187
5.	2016-05-08 04:00	15	17.	2016-05-08 17:00	180
6.	2016-05-08 05:00	31	18.	2016-05-08 18:00	216
7.	2016-05-08 06:00	62	19.	2016-05-08 19:00	210
8.	2016-05-08 07:00	96	20.	2016-05-08 20:00	176
9.	2016-05-08 08:00	132	21.	2016-05-08 21:00	176
10.	2016-05-08 09:00	164	22.	2016-05-08 22:00	177
11.	2016-05-08 10:00	170	23.	2016-05-08 23:00	151
12.	2016-05-08 11:00	231	24.	2016-05-08 24:00	160

Data pada Tabel 2.1 digunakan sebagai data untuk dekomposisi sinyal level 1 (satu) menggunakan *library Pywt* yang terdapat pada Python dengan *mother wavelet coiflet*. Dekomposisi sinyal ini menghasilkan *wavelet coefficients* dengan mengambil *approximation coefficient* saja. Data awal yang berjumlah 24, menjadi setengahnya setelah melalui tahap dekomposisi sinyal yang sudah dijelaskan diatas karena hanya mengambil

a*pproximation coefficient.* A*pproximation coefficient* hasil dekomposisi sinyal ditunjukkan pada Tabel 2.2

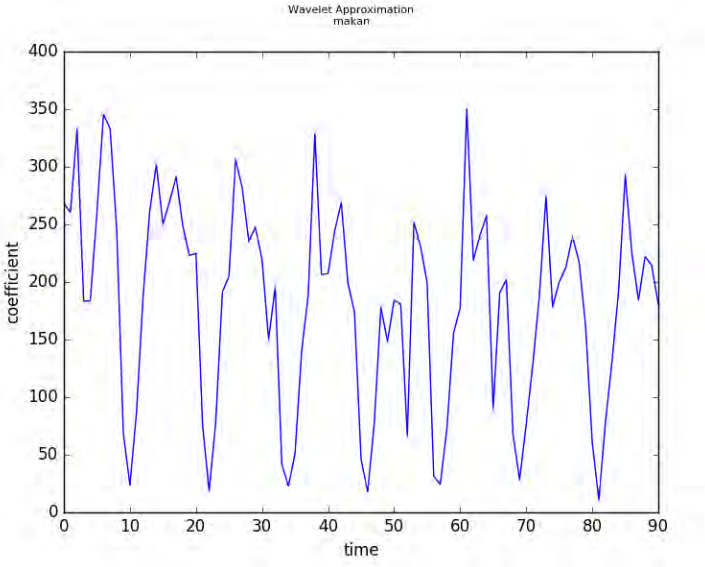
Tabel 2.2 A*pproximation coefficient* hasil dekomposisi kata "makan"

No.	A*pproximation coefficient*	No.	A*pproximation coefficient*
1.	230.11	7.	301.97
2.	68.81	8.	250.10
3.	23.64	9.	269.68
4.	87.02	10.	291.10
5.	187.60	11.	248.88
6.	261.67	12.	222.46

A*pproximation coefficient* Tabel 2.2 merupakan hasil dekomposisi sinyal untuk satu kata yaitu "makan" pada satu hari yaitu tanggal 8 Mei 2016. Hasil pada Tabel 2.2 bergantung pada Tabel 2.1 dimana dari interval pertama hingga kelima mengalami penurunan jumlah frekuensi kemunculan kata "makan" sehingga terlihat pada hasil a*pproximation coefficient* juga mengalami penurunan dari data pertama hingga ketiga dan mengalami kenaikan dimulai dari data keempat hingga mencapai puncaknya pada hasil a*pproximation coefficient* yang ketujuh. Hal ini terlihat sesuai dengan interval waktu dimana data pada Tabel 2.1 mengalami puncaknya yaitu pada pukul 11.00. Berdasarkan pengamatan kata "makan" akan sering muncul pada Twitter ketika jam – jam tertentu seperti jam sarapan, jam makan siang, dan jam makan malam.

Secara keseluruhan a*pproximation coefficient* dari data kata "makan" yang didapat ditampilkan pada Gambar 2.1. Terlihat bahwa gambar *wavelet* tersebut memiliki nilai *wavelet coefficient* yang tinggi pada rentang nilai yang hampir sama pada setiap puncak yang terbentuk. Sehingga menghasilkan pola data yang berulang secara teratur. Pola data seperti kata "makan" disebut sebagai pola data musimam dimana data pada pola musimam ini

akan mengalami fluktuasi, namun fluktuasi terebut akan terlihat berulang dalam suatu interval waktu tertentu. Oleh karena itu dapat disimpulkan bahwa kata kunci “makan” merupakan kata yang berulang secara periodik dan dapat dikatakan sebagai kunci biasa *(non-trivial)* yang dianggap tidak penting.



Gambar 2.1 Wavelet kata kunci "makan"

2.4 Autocorrelation Function

Perhitungan korelasi *coefficient* dilakukan dengan proses *autocorrelation* yaitu dengan menghitung korelasi antara *coefficient* sebuah wavelet dengan wavelet itu sendiri [3]. Pendeteksian sinyal berulang secara periodik pada sinyal non-stasioner dapat dilakukan dengan metode *autocorrelation* pada Wavelet. Sinyal yang berulang secara periodik merepresentasikan kejadian yang berulang pada data twitter sehingga tidak diperlukan dalam proses peringkasan. Prinsip kerja pada *autocorrelation*

adalah dengan menggeser sinyal dengan beberapa penyesuaian pada waktu atau periode.

Autocorrelation Function (ACF) merupakan alat diagnostik yang penting untuk analisis *time series* pada *time domain* [7]. Koefisien ACF akan menunjukkan keeratan hubungan antara nilai variabel dalam hal ini *wavelet coefficient* yang sama tetapi pada waktu yang berbeda. ACF direpresentasikan pada *correlogram* dimana terdapat *lag* yang merupakan jumlah tingkat pergeseran *wavelet* dalam proses pengkorelasian. *Autcocorrelation Function* pada *lag k* dinyatakan sebagai berikut [8] :

$$\rho_k = \frac{\gamma_k}{\gamma_0} = \frac{\sum_{t=1}^{n-k}(x_t-\bar{x})(x_{t+k}-\bar{x})}{\sum_{t=1}^{n-k}(x_t-\bar{x})^2} \qquad (2.2)$$

dimana ρ_k adalah fungsi *autocorrelation* pada *lag-k,* x_t adalah koefisien pada waktu ke-*t,* x_{t+k} adalah *wavelet coefficient* pada waktu ke-*t+k,* $\bar{x}$ adalah rata-rata dari seluruh *wavelet coefficient, k* adalah selisih waktu dan *n* adalah jumlah data dari *wavelet coefficient*.

Pada Tugas Akhir ini dilakuka perhitungan *autocorrelation* dengan meggunakan *coefficient wavelet* yang didapat dari hasil dekomposisi sinyal yaitu *approximation coefficient.* Perhitungan *autocorrelation function* ini menggunakan *library statsmodels.tsa* yang terdapat pada Python yaitu *statsmodels.tsa.stattools.acf()* dengan menggunakan *approcximation coefficient* itu sendiri sehingga menghasilkan *autocorrelation coefficient.* Masing – masing *autocorrelation coefficient* yang dihasilkan direpresentasikan pada masing - masing *lag* yang kemudian ditampilkan pada sebuah *correlogram.* Contoh perhitungan *autocorrelation* pada kata “makan” adalah sebagai berikut :

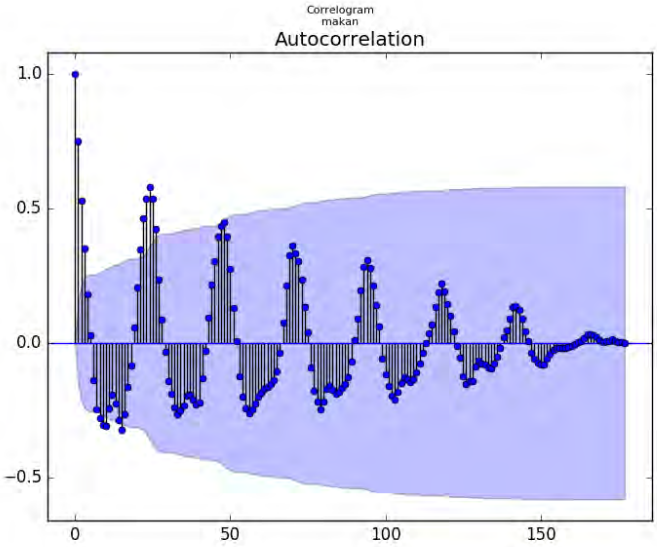
t	x_t	x_{t+1}	x_{t+2}	...,	...,	...,	x_{t-1}	x_{t-2}
1	230.11	68.81	23.64					
2	68.81	23.64	87.02				230.11	

t	x_t	x_{t+1}	x_{t+2}	...,	...,	...,	x_{t-1}	x_{t-2}
3	23.64	87.02	187.60				68.81	230.11
4	87.02	187.60	261.67				23.64	68.81
5	187.60	261.67	301.97				87.02	23.64
6	261.67	301.97	250.10				187.60	87.02
7	301.97	250.10	269.68				261.67	187.60
8	250.10	269.68	291.10				301.97	261.67
9	269.68	291.10	248.88				250.10	301.97
10	291.10	248.88	222.46				269.68	250.10
11	248.88	222.46					291.10	269.68
12	222.46						248.88	291.10
	$\bar{x}$ =203.59							

$$\rho_1 = \frac{(230.11-203.59)(68.81-203.59)+(68.81-203.59)(23.64-203.59)+\cdots+(248.88-203.59)(222.46-203.59)}{(230.11-203.59)^2+(68.81-203.59)^2+\ldots+(222.46-203.59)^2}$$

$\rho_1 = 0{,}702413$

Setelah semua perhitungan nilai *autocorrelation* dilakukan, maka hasilnya akan direpresntasi oleh *Lag* dalam *correlogram* seperti nampak pada Gambar 2.2. Terlihat bahwa pola yang ditampilkan tiap *Lag* secara bergantian menuju ke positif dan negatif dan berulang secara periodik. Hal ini menunjukkan terjadinya fluktuasi yang musiman. Sehingga kata "makan" dapat dikatakan sebagai kata biasa (*trivial)* yang memiliki tingkat korelasi tinggi. Terdapat banyak *lag* dari kata "makan" yang berada pada luar batas *confidence boundary* yaitu batas nilai kepercayaan pada *correlogram.* Hal ini menunjukkan bahwa kata "makan" memiliki nilai korelasi yang tinggi. Oleh karena itu, kata "makan" dianggap sebagai kata kunci *trivial* (biasa). Sedangkan untuk kata kunci *non trivial, lag* dari hasil perhitungan *autocorrelation* dari *wavelet coefficient* berada pada rentang batas *confidence boundary* yang berarti kata tersebut memiliki nilai korelasi yang rendah sehingga dapat dianggap sebagai kata penting atau kata *non trivial.*



Gambar 2.2 Correlogram kata kunci 'makan' yang berulang secara periodik

Confidence boundary adalah batas nilai yang digunakan untuk mengetahui jumlah *lag* dari *autocorrelation function coefficient* yang nilainya melebihi batas signifikansi *autocorrelation coefficient* yang telah ditentukan. Apabila suatu *autocorrelation coefficient* berada di luar rentang *confidence boundary* yang telah ditentukan, maka dapat disimpulkan *autocorrelation coefficient* tersebut signifikan antara nilai suatu *wavelet coefficient* dengan nilai *wavelet coefficient* itu sendiri dengan *time lag* 1 periode. Jumlah *lag* dari *autocorrelation coefficient* yang memiliki nilai lebih dari *confidence boundary* akan digunakan sebagai pertimbangan dari penentuan kata kunci. Hal ini bertujuan untuk mengetahui kata – kata mana yang memiliki nilai *autocorrelation* yang tinggi sehingga kata – kata tersebut dapat dianggap sebagai kata kunci *trivial* atau kata kunci yang berulang secara periodik.

2.5 K-Medoids Clustering

Algoritma K-Medoids *clustering* berguna untuk mengelompokan *tweets* berdasarkan kesamaan kata kunci. Hal ini bertujuan untuk mendapatkan *trending issue* dari *tweets.* Proses klasterisasi dibangun dengan cara menghitung jarak vektor antar dokumen menggunakan *cosine similarity distance.* Pada Tugas akhir ini dibangun matrik berdasarkan jarak vektor antar *tweets* pada setiap kata dalam *tweet* tersebut.

Clustering adalah metode pengelompokan data secara otomatis berdasarkan pola dari data itu sendiri atau tanpa menggunakan data latih (*unsupervised*). K-medoids atau yang sering disebut juga dengan *Partitioning Around Medoids* (PAM) merupakan algoritma *clustering* yang hampir sama dengan Kmeans [9]. K-medoids menggunakan *medoids* yang merupakan entitas dari dataset dan merupakan perwakilan dari kelompok dimana dia dimasukkan. Sedangkan K-means bekerja dengan *centroid* biasanya berupa nilai rata-rata dari jarak objek dalam satu kelompok yang sama.

K-medoids akan mengelompokkan sekumpulan *n* kata menjadi sejumlah *k cluster*. Algoritma ini menggunakan kata kunci pada kumpulan kata kunci untuk mewakili sebuah *cluster*. Kata yang mewakili sebuah *cluster* inilah yang disebut dengan m*edoids*.

Hasil dari proses klasterisasi nantinya berupa *k* kelompok *tweets* dengan *k* adalah *centroid* yang telah ditentukan. Hasil klasterisasi akan digunakan untuk proses ekstraksi *issue* berdasarkan kata kunci dari kelompok *tweets*. Penggunaan pembobotan kata dengan *Term Frequency Inverse Document Frequency (TF-IDF)* digunakan untuk menyeleksi kata yang memiliki bobot diatas nilai ambang, sehingga dapat menghasilkan kata kunci yang mempresentasikan *issue.* Selain itu pembobotan

issue sangat diperlukan untuk mendapatkan *trending issue* yang sesuai.

Algoritma K-medoids

1. Tentukan jumlah cluster yang akan dibentuk (*k*)
2. Pilih *k* kata secara random sebagai inisial *centroid* atau nilai tengah (*medoid)* sebanyak *k cluster*
3. Hitung jarak kemiripan *tweets* yang paling dekat dengan *medoid* menggunakan *cosine similarity distance*
4. Hitung total jarak kemiripan *tweets* terhadap *medoid*
5. Pilih *k* yang bukan *medoid* secara acak sebagai *medoid* baru
6. Hitung jarak kemiripan *tweets* terhadap *medoids* baru
7. Hitung total jarak kemiripan *tweets* terhadap *medoids*
8. Jika total jarak *medoids* baru > total jarak *medoids* lama maka *update medoids*, jika tidak *medoids* tetap
9. Ulangi langkah 3 sampai 7 sampai tidak ada perubahan

Cosine similarity adalah salah satu metode untuk mengukur kemiripan teks dengan menggunakan nilai *cosinus* sudut antara dua vektor [9]. Konsepnya adalah jika terdapat dua vektor dokumen d_i dan d_j maka nilai *cosinus* antara dua pasangan teks tersebut dapat dihitung dengan menggunakan persamaan 2.7. Dimana *w* adalah kata yang diekstrak dari koleksi dokumen. Pada Tugas Akhir ini, koleksi dokumen dianggap sebagai koleksi *tweets.*

$$similarity \underset{(d_i, d_j)}{\rightarrow\rightarrow} = \frac{\vec{d_i}.\vec{d_j}}{|\vec{d_i}|.|\vec{d_j}|} = \frac{\sum_{k=1}^{t}(w_{ki}.w_{kj})}{\sqrt{\sum_{k=1}^{t} w_{ki}^2 . \sum_{k=1}^{t} w_{kj}^2}} \quad (2.7)$$

2.6 Cluster Importance untuk Ekstraksi Trending Issue

Ekstraksi *trending issue* bertujuan untuk mendapatkan satu *issue* yang paling banyak dibicarakan. *Issue* dapat diidentifikasi

dari kata kunci yang muncul pada kelompok *tweets* [4]. *Trending Issue* adalah satu isu yang memiliki skor tertinggi dari *issue* yang ada. Salah satu cara yang dapat dilakukan untuk ekstraksi isu adalah dengan melalui ekstraksi kata kunci. Ekstraksi kata kunci (*keyword extraction*) adalah sebuah cara untuk mencari perwakilan kata yang paling menggambarkan sebuah teks [2]. Inilah yang menjadi dasar penggunaan kata kunci untuk menyaring kalimat yang mengandung *Trending Issue* dalam sebuah berita [2]. Isu yang dimaksud merupakan kumpulan kata kunci yang memiliki kesamaan topik.

Cluster Importance adalah metode pengurutan cluster untuk mengatasi kelemahan dari penggunaan jumlah kalimat pada penilaian sebuah cluster [11]. *Cluster Importance* mempertimbangkan bobot kata yang terkandung dalam kalimat yang berada pada satu *cluster* sehingga lebih dapat merepresentasikan isi kalimat dibandingkan hanya mempertimbangkan jumlah kalimat dalam *cluster*. Pembobotan *cluster* dilakukan untuk mengukur seberapa banyak informasi yang terkandung dalam sebuah *cluster*. Semakin besar bobot sebuah *cluster* maka *cluster* tersebut diangap sebagai *cluster* penting.

Pada Tugas Akhir ini kalimat dianggap sebagai *tweet* pada saat *ekstraksi trending issue* dan bobot akan diberikan pada *cluster* berdasarkan jumlah kata penting yang terkandung dalam *cluster*. Setiap kata penting atau kata kunci dianggap sebagai sebuah *cluster*. Kata penting diseleksi berdasarkan jumlah kemunculan kata pada *tweet* atau dokumen input (*count(k)). Count(k)* yang terseleksi nilainya harus lebih besar dari nilai ambang (*threshold*). Sehingga bobot sebuah *cluster* dihitung dengan menjumlahkan bobot dari seluruh kata penting yang muncul dalam sebuah *cluster*, cara penghitungan bobot *cluster (W(CI))* selengkapnya dapat dilihat pada persamaan 2.6.

$$bobot\ cluster\ CI, W(CI) = \sum_{w \in C} \log(1 + count(k)) \qquad (2.6)$$

2.7 Teknik Pembobotan Kalimat untuk Peringkasan Berita

Pembobotan kalimat (*sentence scoring*) merupakan salah satu fase penting dalam peringkasan dokumen dan telah banyak digunakan pada peringkasan dokumen secara ekstraktif, baik untuk *single document* maupun *multi document.* Secara umum, metode pembobotan kalimat pada peringkasan berita dikelompokan menjadi tiga kategori [12] yaitu berdasarkan bobot kata (*word-based scoring*), berdasarkan fitur kalimat (*sentence-based scoring*), dan berdasarkan pada relasi antar kalimat yang direpresentasikan dengan graf (*graph-based scoring*). Teknik pembobotan kalimat yang digunakan yaitu *Word Frequency (WF), Term Frequency Inverse Document Frequency (TF-IDF),* posisi kalimat, dan kemiripan kalimat terhadap judul berita [12]. Untuk membuat hasil peringkasan menjadi lebih koheren dengan berita yang ada, maka penambahan teknik pembobotan berdasarkan fitur tweet yaitu kemiripan kalimat terhadap *trending issue* [2]. Selain itu untuk mengurangi redudansi ditambahkan teknik pembobotan terkait redudansi.

1. Word Frequency (WF)

Konsep dari WF adalah semakin sering suatu kata muncul dalam sebuah teks, maka kata terebut dianggap sebagai kata penting [12]. Untuk mendapatkan kata penting dalam sebuah dokumen dilakukan pembobotan kata dengan menghitung frekuensi kemunculan kata tersebut pada dokumen. Langkah awalnya adalah melakukan ekstraksi kata atau *term* dari dokumen kemudian memberikan bobot pada tiap kata berdasarkan jumlah kemunculan kata pada dokumen. Kemudian merangking kata berdasarkan bobot dan menyeleksi kata yang memiliki bobot diatas nialai ambang (*threshold*).

Kata yang terselekti akan menjadi *Word Frequency List (WFList)*. *WFList* inilah yang nantinya digunakan sebagai fitur pada pembobotan kalimat dengan cara

mengukur kemiripan antara kalimat terhadap *WFList.* Metode untuk mengukur kemiripan dapat menggunakan *cosine similarity* atau metode pengukur kemiripan yang lain. Pada Tugas Akhir ini *WFList* dibuat berdasarkan kata yang terdapat pada setiap kalimat dalam berita seperti Gambar 2.3 dimana kemunculan kata dihitung berdasarkan kemunculan suatu kata per kalimat di berita tersebut.

```
Hasil WF
ini wf
[[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 2 0 0 0 0 1 0 1 0 0 2 0 0 0]
 [0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 1 2 1 0 1 0 0 0 0 0 0 0 1 2 0 1 1 0 1 0 1 0 1 2 1 0 0]
 [0 1 1 0 1 0 0 1 0 0 0 1 0 1 0 2 2 1 1 1 0 1 1 1 2 1 0 1 3 1 3 2 0 1 1 2 0 1 3 1 0 1]
 [0 1 1 1 1 0 0 1 1 0 1 1 0 1 0 2 2 1 1 2 0 1 1 1 2 1 0 1 3 1 4 2 1 1 1 2 0 1 3 1 0 1]
 [0 1 1 1 1 1 0 1 1 0 1 1 0 1 1 2 2 1 1 2 1 2 1 1 2 1 1 1 3 1 4 2 1 1 1 2 1 1 4 1 0 1]
 [1 1 1 1 2 1 1 1 2 0 1 1 0 1 1 4 2 1 1 2 1 3 1 2 2 1 1 1 4 1 4 2 2 1 1 2 1 1 4 1 0 1]
 [1 1 1 1 2 1 2 1 2 1 1 1 1 1 1 5 2 1 1 2 1 3 1 2 2 1 1 1 4 1 4 2 2 1 1 2 1 1 5 1 1 1]]
```

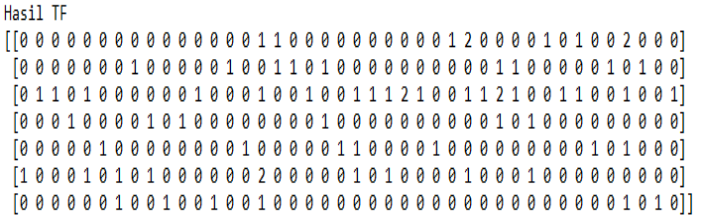
Gambar 2.3 Salah satu WF yang dihasilkan setelah dilakukan perhitungan untuk satu berita

2. Term Frequency – Inverse Document Frequency (TF-IDF)

Term Frequency Inverse Document Frequency (TF-IDF) adalah konsep pembobotan kata pada sebuah dokumen. Ketika TF-IDF diterapkan pada lingkup kalimat, maka sebuah kalimat akan diberlakukan sebagai dokumen. Jika ada "kata-kata yang spesifik" muncul pada kalimat tertentu maka kalimat tersebut relatif dianggap sebagai kalimat penting [12]. Metode ini melakukan perbandingan antara frekuensi kemunculan kata-*i* pada kalimat-*j* (TF_{ij}) dengan frekuensi dokumen yang mengandung kata-*j* (DF_j).

$$TFIDF_{w_i doc_j} = TF_{w_i doc_j} * IDF_{w_i} \quad (2.3)$$

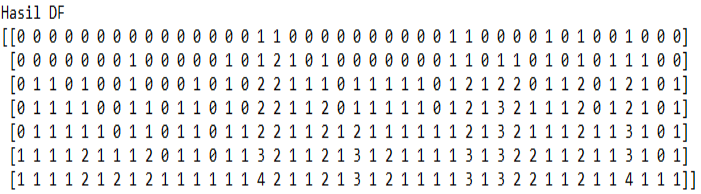
Hasil $TF_{w_i doc_j}$ dapat dilihat pada Gambar 2.4 dan DF_{w_i} pada Gambar 2.5 dimana setiap hasil perhitungan bobot kata ini akan digunakan untuk perhitungan TFIDF.



Gambar 2.4 Salah satu contoh hasil TF untuk satu berita

Bobot TF-IDF dari kata-*i* dapat dihitung dengan menggunakan persamaan 2.3, dimana TF$w_i doc_j$ adalah frekuensi kemunculan kata *w* ke-*i* (w_i) pada dokumen *doc* ke-*j* (doc_j) dengan memberikan pengukuran terhadap pentingnya kata (w_i) pada dokumen tersebut. Dimana *N* adalah jumlah dokumen, DFw_i adalah jumlah dari dokumen yang mengandung kata (w_i), persamaan 2.4.

$$IDF_{(N,DF_{w_i})} = log\left(\frac{N}{DF_{w_i}}\right) \tag{2.4}$$



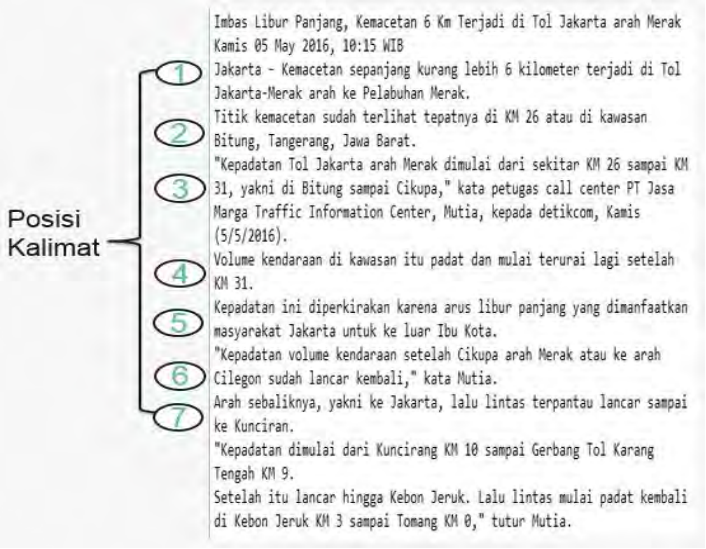
Gambar 2.5 Salah satu contoh IDF untuk satu berita

Normalisasi bobot TF-IDF yang menghasilkan nilai 0-1 dari suatu kata (w_i) pada dokumen (doc_j) ditunjukkan pada Persamaan 2.5 dimana *M* adalah jumlah kata (w_i) pada dokumen (doc_j).

$$TFIDF_{w_i doc_j} = \frac{TF_{w_i doc_j} * IDF_{w_i}}{\sqrt{\sum_{i=1}^{M} (TF_{w_i doc_j} * IDF_{w_i})^2}} \tag{2.5}$$

3. Posisi Kalimat

Posisi kalimat merupakan salah satu fitur yang dapat digunakan untuk pembobotan kalimat. Dimana penilaiannya berdasarkan pada letak kalimat dalam sebuah dokumen. Dengan menggunakan aturan, kalimat yang posisinya berada diawal dokumen memiliki skor lebih besar dibanding kalimat yang posisinya diakhir. Sedangkan untuk kemiripan kalimat terhadap judul berita digunakan sebagai dasar untuk mengetahui informasi untuk mengetahui kalimat penting dalam sebuah berita. Gambar 2.6 menunjukkan contoh penentuan posisi kalimat berdasarkan urutan kalimat pada berita tersebut.



Gambar 2.6 Penentuan posisi kalimat berdasarkan urutan kalimat pada berita

4. Kemiripan Kalimat terhadap Judul

Konsep dari teknik pembobotan kalimat berdasarkan kemiripan kalimat terhadap judul adalah bahwa bobot sebuah kalimat besar ketika nilai kemiripan antara judul dengan kalimat tinggi. Semakin besar bobot kalimat maka kalimat tersebut akan dianggap semakin penting. Kalimat yang mirip dengan judul dan kalimat yang mencakup kata-kata dalam judul yang akan dianggap sebagai kalimat penting [12].

5. Kemiripan Kalimat terhadap *Trending Issue*

Kemiripan kalimat terhadap *Trending Issue* merupakan teknik pembobotan kalimat berdasarkan dari isu yang berkembang. *Trending issue* didapat melalui ekstraksi *trending issue* dari ekstraksi kata kunci yaitu sebuah cara untuk mencari perwakilan kata yang paling menggambarkan sebuah teks. Inilah yang menjadi dasar penggunaan kata kunci untuk menyaring kalimat yang mengandung *Trending Issue* dalam sebuah berita. *Issue* diesktraksi berdasarkan kumpulan kata kunci yang memiliki kesamaan topik yang muncul pada *tweets*. Kelompok *tweets* yang memiliki bobot terbesar akan diseleksi dan digunakan mengekstrak *trending event*.

2.8 Metode Evaluasi

Metode evaluasi yang digunakan untuk mengukur kualitas *cluster* adalah metode *silhouette coefficient*. Sedangkan metode evaluasi yang digunakan untuk mengukur kualitas ringkasan adalah metode ROUGE (*Recall-Oriented Understudy for Gisting*

Evaluation). Pada Tugas Akhir ini akan dilakukan perbandingan hasil perhitungan *cluster* pada kumpulan data *tweet* yang hanya mengandung kata kunci penting (*non-trivial)* dengan kumpulan data *tweet* asli.

2.8.1 Silhouette Coefficients

Silhouette coefficient merupakan sebuah metode yang berfungsi untuk mengukur kualitas dari *cluster* yang dihasilkan [13]. Proses perhitungan s*ilhouette coefficient* memerlukan jarak antar dokumen yang mengindikasikan derajat kepemilikan setiap objek yang berada di dalam *cluster*. Nilai *Silhouette* dari sebuah objek berada pada rentang antara -1 sampai dengan 1. Derajat objek akan tinggi jika nilai *silhouette* –nya mendekati 1(satu).

$$S_i = \frac{(b_i - a_i)}{\max(a_i - b_i)} \qquad (2.8)$$

Pada Tugas Akhir ini *tweet* dianggap sebagai objek maka penghitungan nilai *sihouette* (S_i) untuk tiap *tweet* menggunakan persamaan 2.8, dimana a_i merupakan nilai rata – rata jarak *tweet i* terhadap seluruh *tweet* yang berada pada *cluster* yang sama dengan *tweet i.* Sedangkan b_i adalah nilai rata – rata terkecil jarak *tweet i* terhadap seluruh *tweet* yang berada pada seluruh *cluster* selain *cluster tweet i* berada.

Setelah nilai S_i didapat untuk tiap *tweet* pada tiap *cluster* langkah selanjutnya adalah mencari rata-rata nilai S_i untuk tiap cluster atau yang lebih dikenal dengan *Average Silhouette Width* (ASW) yang mampu mengindikasikan kualitas *clustering.* Nilai rata – rata *silbouette* dibedakan menjadi 4 [13] sebagai berikut :

1. sangat baik dengan range $0.71 \leq ASW \leq 1$,
2. sudah baik dengan range $0.51 \leq ASW < 0.71$,

3. cukup baik dengan range $0.26 \leq ASW < 0.51$,
4. kurang baik dengan range $ASW < 0.26$).

2.8.2 ROUGE (Recall-Oriented Understudy for Gisting Evaluation)

ROUGE (*Recall-Oriented Understudy for Gisting Evaluation*) adalah metode yang digunakan untuk mengukur kualitas dari sebuah ringkasan [14]. ROUGE akan membandingkan antara rangkuman yang dihasilkan oleh sistem terhadap rangkuman ideal (*Groundtruth*) yang dibuat oleh pakar. Dalam hal ini, ringkasan yang dihasilkan oleh sistem disebut dengan kandidat ringkasan sedangkan ringkasan yang digunakan sebagai *Groundtruth* atau *Human Gold Standart* (HGS) merupakan ringkasan sebagai referensi . ROUGE sangat efektif digunakan untuk mengevaluasi peringkasan dokumen [14].

ROUGE mengukur kualitas hasil ringkasan dengan menghitung unit-unit yang *overlap* seperti *N-gram*, urutan kata dan pasangan-pasangan kata antara ringkasan kandidat dan ringkasan sebagai referensi [14]. ROUGE-N adalah nilai *recall* dari *n-gram* yang ada pada kandidat ringkasan terhadap *Groundtruth*. ROUGE-*N* mengukur perbandingan *N-gram* dari dua ringkasan, dan menghitung berapa jumlah yang cocok. Perhitungan ROUGE-*N* dapat dilihat pada persamaan 2.9.

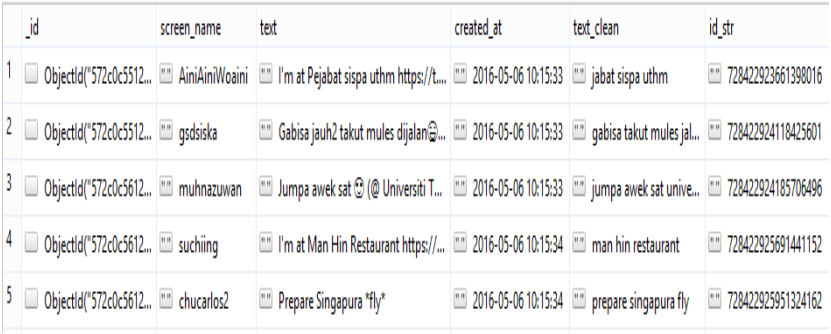
$$ROUGE - N = \frac{\sum_{S \in \{ReferenceSummaries)} \sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{S \in \{ReferenceSummaries)} \sum_{gram_n \in S} Count(gram_n)} \quad (2.9)$$

dimana *N* menunjukkan panjang dari *N-gram*, *S* merupakan ringkasan kandidat yang dihasilkan oleh sistem, $Count_{match}$*(N-gram)* adalah jumlah maksimum dari *N-gram* yang muncul pada ringkasan kandidat dan ringkasan sebagai referensi (*Groundtruth)*. *Count(N-gram)* adalah jumlah dari *N-gram* pada ringkasan sebagai referensi.

2.9 Database MongoDB untuk penyimpanan kata

MongoDB adalah database dokumen *open-source* yang memberikan kinerja tinggi, ketersediaan tinggi, dan skala otomatis [15]. Database MongoDB merupakan sistem basis data NoSQL. NoSQL adalah sebuah sistem basis data tidak hanya harus menggunakan perintah SQL untuk melakukan proses manipulasi data. MongoDB merupakan sistem basis data yang menggunakan konsep key-value, artinya setiap dokumen dalam MongoDB pasti memiliki key. Hal ini berbeda dalam RDMBS yang dapat membuat sebuah tabel tanpa menggunakan primary key. MongoDB secara otomatis akan memberikan key pada setiap dokumen [16]. Penggunaan konsep key-value membuat MongoDB menjadi sistem basis data yang sangat cepat jika dibandingkan dengan non key-value seperti RDBMS. MongoDB menggunakan bahasa BSON, dimana BSON merupakan singkatan dari Binary JSON. Seperti JSON, BSON mendukung embedding dokumen dan dokumen lainnya dalam array.

Library yang digunakan untuk mengakses MongoDB pada Python adalah *Pymongo.* Pada MongoDB tabel disebut dengan *collection.* Pada Tugas Akhir ini database MongoDB digunakan sebagai penyimpanan data Twitter dari hasil *crawling* dan disimpan didalam *collections.* Data *tweets* disimpan dalam array dengan format JSON. Gambar 2.7 merupakan salah satu contoh *collection* yang digunakan untuk penyimpanan data pada database MongDB. Struktur data dari *collection* ini terdiri dari "_id" yang secara *default* berasal dari MongoDB, "screen_name" digunakan untuk menyimpan nama akun pengguna Twitter, "text" digunakan untuk menyimpan *tweet* yang ditulis oleh pengguna Twitter, "created_at" berfungsi untuk menyimpan waktu *tweet* muncul, "text clean" berfungsi untuk menyimpan *tweet* yang sudah melewati preproses dan yang terakhir "id_str" untuk menyimpan *id* dari *string* kata.



	_id	screen_name	text	created_at	text_clean	id_str
1	ObjectId("572c0c5512...	AiniAiniWoaini	I'm at Pejabat sispa uthm https://t....	2016-05-06 10:15:33	jabat sispa uthm	728422923661398016
2	ObjectId("572c0c5512...	gsdsiska	Gabisa jauh2 takut mules dijalan😩...	2016-05-06 10:15:33	gabisa takut mules jal...	728422924118425601
3	ObjectId("572c0c5612...	muhnazuwan	Jumpa awek sat 😍 (@ Universiti T...	2016-05-06 10:15:33	jumpa awek sat unive...	728422924185706496
4	ObjectId("572c0c5612...	suchiing	I'm at Man Hin Restaurant https://...	2016-05-06 10:15:34	man hin restaurant	728422925691441152
5	ObjectId("572c0c5612...	chucarlos2	Prepare Singapura *fly*	2016-05-06 10:15:34	prepare singapura fly	728422925951324162

Gambar 2.7 Salah satu contoh *collection* yang digunakan untuk penyimpanan data pada database

BAB III
ANALISIS DAN PERANCANGAN SISTEM

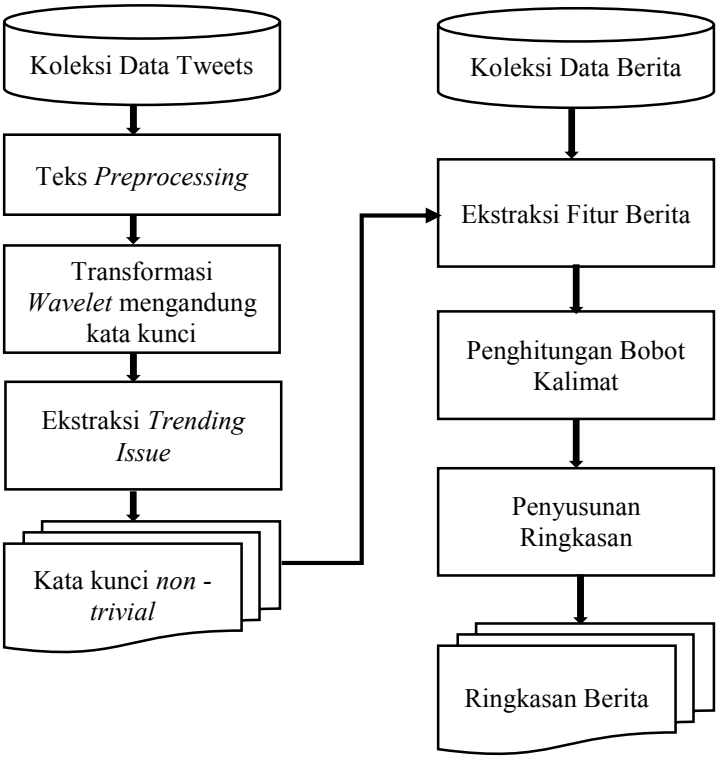
Bab ini membahas mengenai analisis dan perancangan pembuatan sistem perangkat lunak. Perancangan yang dijelaskan meliputi analisis dan desain metode serta perancangan antar muka. Analisis dan metode akan membahas data dan proses dari sistem ini. Data yang dimaksud adalah data yang akan diolah dalam perangkat lunak sehingga tujuan Tugas Akhir ini bisa tercapai. Selain itu akan dijelaskan juga desain metode secara umum pada sistem.

3.1 Analisis dan Desain Metode Secara Umum

Pada Tugas Akhir ini akan dibangun suatu sistem yang dapat mengimplementasikan penggunaan metode *Autocorrelation Wavelet Coefficients* untuk pemilihan kata kunci penting pada peringkasan multi-dokumen berita berbahasa Indonesia. *Autocorrelation Wavelet Coefficients* berfungsi untuk mendeteksi kejadian biasa (*trivial)* dalam hal ini kata yang muncul secara periodik atau berulang untuk mendapatkan kata kunci yang dianggap penting (*non-trivial)* sehingga dapat mempresentasikan isi dari kejadian dalam hal ini adalah berita. Pendeteksian kejadian dilakukan pada kata-kata dari setiap *tweet* yang terdapat pada Twitter.

Kata kunci penting yang didapat digunakan untuk peringkasan berita. Hal ini bertujuan untuk mendapatkan peringkasan berita yang sesuai dengan kejadian yang berlangsung dan isu yang berkembang pada saat itu. Secara garis besar sistem ini melalaui 3(tiga) fase, sebagai berikut :

1. Fase transformasi *wavelet*, dimana setelah melalui proses *text processing,* data disimpan kedalam database. Data-data tersebut selanjutnya akan di proses untuk mendapatkan kata kunci penting (*non trivial)* menggunakan *Autocorrelation Wavelet Coefficients*.



Gambar 3.1 Diagram Alur Ringkasan

2. Fase ekstraksi *trending issue,* yaitu pada tahap ini akan dilakukan pencarian isu yang sedang berkembang berdasarkan kata kunci penting yang sudah didapat.
3. Fase peringkasan berita, dimana pada tahap ini akan dilakukan penghitungan total bobot kalimat dan penyusunan ringkasan.Proses peringkasan berita menggunakan teknik pembootan kalimat yaitu kemiripan kalimat terhadap *trending issue, Word Frequency (WF), Term Frequency Inverse Document Frequency (TF-IDF),* posisi kalimat, dan kemiripan kalimat terhadap judul berita.

Berikut ringkasan alur proses perngerjaan Tugas Akhir Implementasi Peringkasan Multi-Dokumen Berita Berbahasa Indonesia dengan Pemilihan Kata Kunci menggunakan *Autocorrelation Wavelet Coefficients* sesuai pada Gambar 3.1 Diagram Alur Ringkasan

3.1.1 Penyimpanan Data

Data yang akan digunakan adalah data Twitter yang diambil dari hasil *crawling.* Data Twitter yang diambil adalah data teks yang biasa disebut *tweet* dan data waktu kemunculan *tweet* tersebut. Data *tweet* diambil dari Twitter dengan melakukan autentifikasi dengan 4 (empat) buah *key* pada Twitter API yang sudah dijelaskan pada subbab 2.1. *Library* yang digunakan adalah *library Twitter* yang tersedia pada Python. Sedangkan data berita didapat dari *link* yang ada pada *tweet* berita. *Link* tersebut akan diakses untuk mendapatkan dokumen berita pada website yang telah ditentukan oleh *link* tersebut.

Implementasi Tugas Akhir ini diawali dengan proses *crawling* data Twitter secara langsung. Proses ini dibatasi pada wilayah geografis Indonesia dalam rentang waktu tertentu. Data yang dihasilkan disimpan ke dalam *database* MongoDB dalam format JSON dengan 3(tiga) tabel penyimpanan yang berbeda fungsi yaitu tabel untuk menyimpan data *tweet* yang asli beserta *tweet* yang sudah melewati tahap *preprocessing*, tabel '*term frequency'* yang berisi kata dan jumlah kemunculan dari kata-kata tersebut, dan tabel '*term detail'* yang berisi kata dan waktu kemunculannya. Tabel 3.1 Struktur Tabel Penyimpanan Tweet menunjukkan struktur tabel penyimpanan *tweet* pada database.

Tabel 3.1 Struktur Tabel Penyimpanan *Tweet*

No.	Jenis Data	Keterangan
1.	ObjectId	Kode otomatis sebagai *key* dari database
2.	Id_str	Kode *tweet*
3.	Created_at	Waktu pengiriman *tweet*
4.	Text	*Tweet* asli sebelum *preprocess*

No.	Jenis Data	Keterangan
5.	Text_clean	*Tweet* setelah melalui *preprocess*
6.	Sceen_name	*Username* pengirim *tweet*

Dokumen *tweet* yang tersimpan memiliki bentuk seperti pada Gambar 3.2 dalam format JSON yang merupakan implementasi dalam satu baris data terdiri dari '_id' yang merupakan kode unik yang didapat secara otomatis dari MongoDB. '*screen_name*' merupakan id pengguna yang ditampilkan secara publik. '*text*' merupakan pesan teks dari sebuah *tweet* yang merupakan sumber informasi dari sebuah *tweet* yang akan diproses dan '*text_clean*' merupakan data pada '*text'* yang sudah melewati tahap teks *preprocessing.* '*created_at*' merupakan informasi waktu dari *tweet* dikirim dan '*id_str*' adalah kode id masing – masing *tweet.*

"_id" : ObjectId("572c0db412599f18c8215b40"),
"screen_name" : "randibocil",
"text" : "Tontonan lintas generasi dan cocok buat semua umur: DORAEMON :D",
"created_at" : "2016-05-06 10:21:24",
"text_clean" : "tonton lintas generasi cocok umur doraemon".

Gambar 3.2 Contoh data pada tabel *Tweet* dalam format JSON

Tabel 3.2 Struktur Tabel *Term Frequency*

No.	Jenis Data	Keterangan
1.	ObjectId	Id otomatis sebagai key dari database
2.	*Term*	Kata atau *term* identik
3.	*Count*	Jumlah kemunculan atau *term frequency*

Tabel 3.2 Struktur Tabel *Term Frequency* menunjukkan struktur tabel penyimpan *term frequency.* Pada tabel ini, *tweet* sudah melalui tahapan *preprocess,* dimana frekuensi kemunculan kata disimpan dan dihitung jumlah kemunculannya pada setiap *tweet* yang mengandung kata tersebut. Gambar 3.3 merupakan

contoh kata yang disimpan dalam tabel *term frequency* yang berisi '*_id*', '*term*' dan '*count*'dimana kata '*semangat'* memiliki frekuensi kemunculan sebanyak 7007 kali.


```
"_id" : ObjectId("572c0c4d336752028015a461"),
 "term" : "semangat",
 "count" : 7007
```


Gambar 3.3 Contoh data pada tabel *term frequency*

Sedangkan tabel *term frequency detail* akan menyimpan informasi setiap kemunculan kata disertai dengan waktu kemunculannya dan dicatat secara historikal. Struktur tabel *term frequency detail* ditunjukkan pada Tabel 3.3 dan contoh data yang akan disimpan pada tabel *term frequency detail* ditunjukkan oleh Gambar 3.4. Data pada 2(dua) tabel ini dibutuhkan saat proses transformasi *wavelet* yang memperhatikan urutan waktu kemunculan kata.

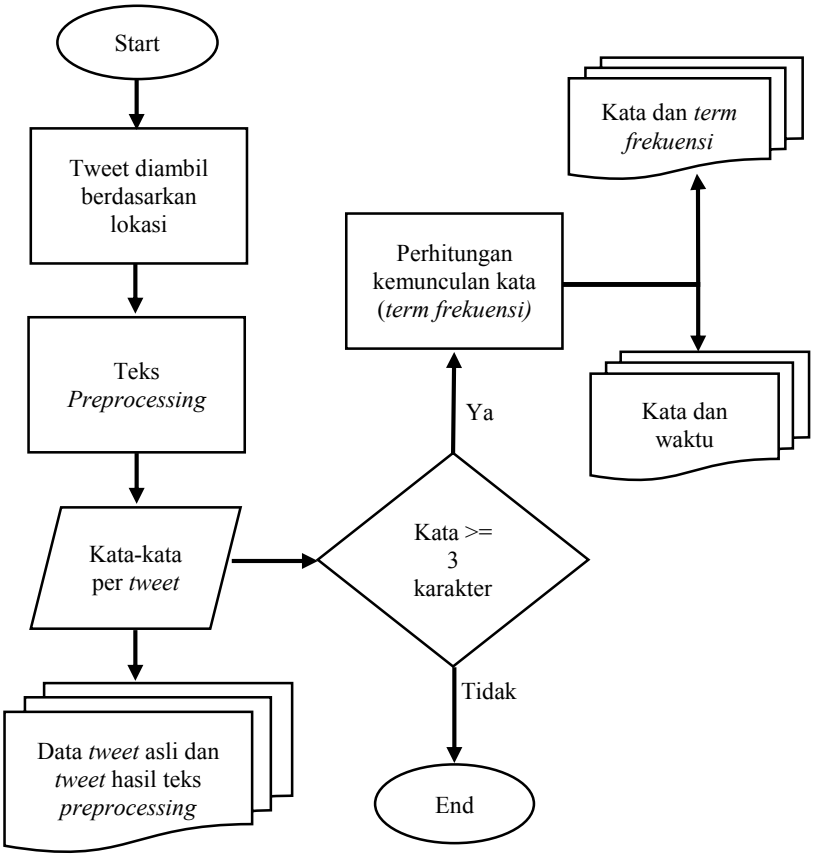

```
"_id" : ObjectId("572c0c4d12599f18c8213498"),
 "term" : "semangat",
 "created_at" : "2016-05-06 10:15:24"
```


Gambar 3.4 Contoh data pada tabel *term frequency* detail

Tabel 3.3 Struktur Tabel *Term Frequency Detail*

No.	Jenis Data	Keterangan
1.	ObjectId	Id otomatis sebagai key dari database
2.	*Term*	Kata atau *term* identik
3.	*Created at*	Waktu kemunculan term pada *tweet*

Secara umum proses penyimpanan data ditunjukkan pada Gambar 3.5 dimana proses dimulai dari pengambilan data *tweet,* tahapan teks *preprocessing,* dan penyimpanan *tweet* hasil *preprocessing* ke dalam database *MongoDB.*



Gambar 3.5 Alur proses peyimpanan data ke database

3.1.2 Teks Preprocessing

Tweet yang sudah terjaring akan melalui tahapan teks *preprocessing*. Teks *preprocessing* merupakan tahapan awal dalam pemrosesan teks. Teks *preprocessing* bertujuan untuk mendapatkan kata – kata pada suatu teks sehingga hasilnya dapat

digunakan untuk proses lainnya. Adapun tahapan yang dilakukan pada teks *preprocessing* adalah sebagai berikut :

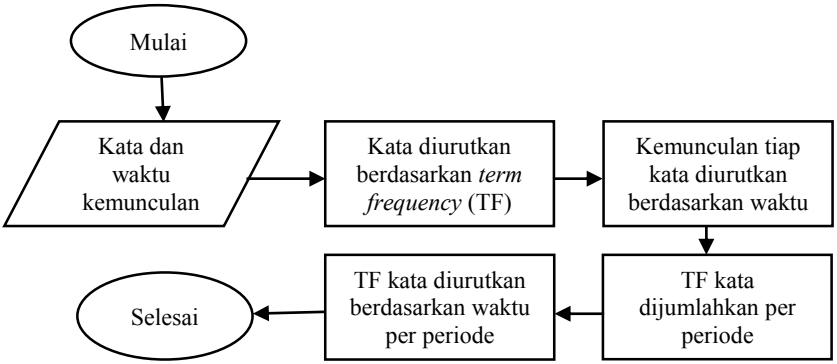
1. **Segmentasi**
   Pada tahapan ini, teks yang terdapat pada *tweet* akan dipisahkan atau dipotong – potong berdasarkan tiap-tiap kata yang membangun *tweet* tersebut. Kata-kata tersebut dipisahkan berdasarkan spasi dan tanda baca menggunakan *library re* dengan fungsi *split*() pada Python.
2. ***Stopword removal* atau Penghilangan *stopword***
   Kata – kata dari tiap *tweet* yang sudah didapat dari proses segmentasi kemudian dicek satu persatu. Proses pertama dilakukan pembuangan "URL". Setelah itu, setiap kata dalam *tweet* dibandingkan dengan kata – kata yang terdapat pada *stopword list* yang sudah disimpan berbentuk *file*. Jika terdapat kata yang sama pada *stopword* list, maka kata tersebut akan dibuang atau dihilangkan.
3. ***Stemming***
   Setelah kata yang dianggap tidak penting dibuang, maka proses selanjutnnya adalah mendapatkan kata dasar dari kata tersebut. Proses *stemming* dilakukan berdasarkan aturan-aturan yang terdapat dalam penulisan kata Bahasa Indonesia, seperti kata "mencari" menjadi "cari'. Proses yang dilakukan adalah seperti membuang awalan, imbuhan, akhiran, dan perubahan kata lainnya [5].

3.1.3 Transformasi *Wavelet* Mengandung Kata Kunci

Kata-kata yang sudah melalui tahap *preprocessing* akan diproses dengan melakukan perhitungan frekuensi kemunculan suatu kata pada setiap *tweet* yang mengandung suatu kata tersebut. Setiap kata yang dihitung adalah kata yang memiliki karakter lebih dari 3(tiga) huruf. Hal ini dimaksudkan untuk memaksimalkan kumpulan kata yang disimpan. Pada Tugas Akhir ini implementasi perhitungan kata dilakukan menggunakan *query* yang langsung disimpan ke dalam database. Perhitungan frekuensi kemunculan

kata dimaksudkan agar data Twitter yang tadinya berupa data teks menjadi data berupa angka dimana data angka tersebut adalah jumlah frekuensi kemunculan kata.

Setiap kata yang masuk akan disimpan waktu kemunculannya. Data kemunculan setiap kata disimpan berdasarkan waktu kata tersebut muncul dari setiap *tweet* yang masuk. Data waktu akan berguna untuk proses perubahan data teks menjadi data angka. Tahapan – tahapan yang dilakukan untuk merubah data teks menjadi data angka, dimana data yang akan dihasilkan adalah frekuensi kata per 60 menit berdasarkan waktu kemunculan sesuai pada Gambar 3.6, adalah sebagai berikut :



Gambar 3.6 Alur perubahan data teks menjadi frekuensi kata kunci per periode waktu

1. Pengurutan kata berdasarkan frekuensi kemunculannya (*term frequency)*. Terlihat seperti Tabel 3.4 yang menunjukkan kata 'kuala' memiliki frekuensi kemunculan kata yang paling tinggi. Kata "kuala" dan "lumpur" muncul paling tinggi dikarenakan banyak *tweet* yang terjaring mengandung kata – kata tersebut. Sedangkan kata "pagi", "makan" dan "happy" merupakan kata umum yang sering di muncul pada *tweet.*

Tabel 3.4 Frekuensi kemunculan kata

No	Kata	*Term Frequency*
1	kuala	45227
2	lumpur	39779
3	pagi	22766
4	makan	22647
5	happy	20777

2. Kemunculan tiap kata diurutkan berdasarkan waktu setiap kali kata tersebut muncul. Pada Tabel 3.5 kemunculan kata "kuala" disimpan setiap kali kata tersebut muncul.

Tabel 3.5 Kemunculan tiap kata "kuala" berdasarkan waktu

Kata	Waktu
Kuala	2016-05-06 10:15:28 . . . . 2016-05-09 12:47:05 . . . . 2016-05-13 21:58:54

3. *Term frequency* (TF) kata dijumlahkan per periode. Kemunculan kata akan dihitung jumlah kemunculannya per periode dalam hal ini per 60 (enam puluh) menit. Tabel 3.6 menunjukkan jumlah kemunculan kata "kuala" per periode dari waktu kemunculan awal hingga akhir.

Tabel 3.6 Jumlah kata "kuala" per periode

Periode ke -	*Term Frequency*
1	364
2	359

Periode ke -	*Term Frequency*
3	688
.	.
.	.
.	.
n	182

4. *Term frequency* (TF) kata diurutkan berdasarkan waktu per periode. Jumlah kata akan diurutkan setiap 60 menit yang merupakan periode dari interval waktu yang telah ditetapkan. Tabel 3.7 menunjukkan jumlah kemunculan kata pada waktu kemunculan setiap 60 menit.

Tabel 3.7 Frekuensi kemunculan kata per periode berdasarkan waktu

Waktu	*Term Frequency*
2016-05-06 10:00:00	364
2016-05-06 11:00:00	359
2016-05-06 12:00:00	688
.	.
.	.
.	.
2016-05-13 21:00:00	182

Frekuensi kemunculan kata per 60 menit ini akan digunakan untuk pembentukan *Wavelet*. Nilai dari frekuensi kata akan digunakan sebagai data yang nantinya akan diolah sehingga menghasilkan *wavelet coefficient. Wavelet coefficient* yang dihasilkan selanjutnya akan digunakan untuk perhitungan *autocorrelation.* Berikut tahapan – tahapan yang dilalui untuk mendapatkan kata *non trivial* dari *tweet* adalah sebagai berikut :

1. **Dekomposisi sinyal**

   Data frekuensi kemunculan kata yang sudah tersusun berdasarkan urutan waktu dirubah menjadi *coefficients wavelet.* Kata – kata yang diambil berjumlah 6500 kata yang memiliki frekuensi kemunculan tertinggi. Kemudian data dari kata – kata ini akan di dekomposisi untuk menghasilkan *coefficient wavelet.* Jenis *mother wavelet* yang digunakan adalah *coiflet* dengan level dekomposisi tingkat 1 (satu). Jenis *wavelet* ini memiliki sifat penempatan yang sama baik untuk *coefficient* aproksimasi ataupun detail.

2. **Deteksi puncak (*peak)* sinyal**

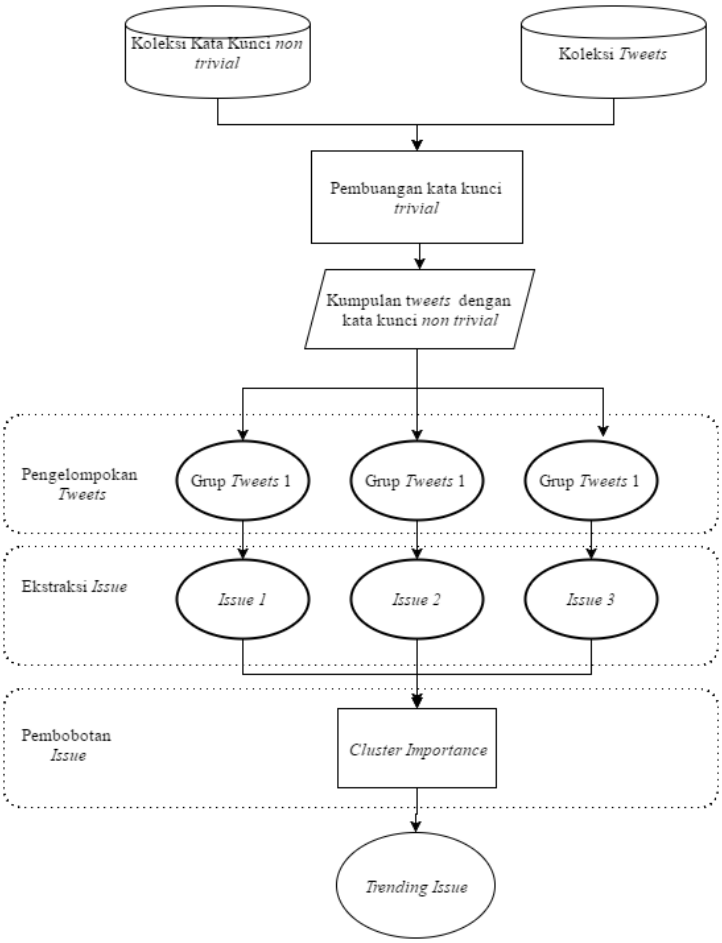
   Deteksi puncak dilakukan untuk menentukan kata kunci mana yang dianggap dapat merepresentasikan suatu kejadian penting. Apabila sebuah *wavelet* dari kata kunci terdeteksi tidak terdeteksi *peak* atau puncak maka kata kunci tidak akan dianggap sebagai kata kunci pada kejadian penting.

3. **Perhitungan *Coefficient Autocorrelation***

   Proses perhitungan korelasi *coefficient* dilakukan menggunakan *autocorrelation function* (*ACF).* Perhitungan *autocorrelation* dilakukan dengan cara menghitung korelasi antara *coefficient* sebuah *wavelet* dengan *coefficient wavelet* itu sendiri. Selain itu digunakan nilai *confidence boundary* untuk mengetahui kata mana yang dianggap memiliki tingkat keperiodikan tinggi atau rendah.

Kata – kata yang sudah melalui tahap transformasi sinyal *wavelet* akan menghasilkan kata – kata *non-trivial.* Kata *non-trivial* didapat dari eliminasi kata *trvial* berdasarkan perhitungan *coefficients autocorrelation.* Kata – kata yang memiliki nilai *autocorrelation* dan tingkat keperiodikan yang tinggi dianggap sebagai kata *trivial.* Kata – kata *non trivial* ini akan digunakan untuk proses ekstraksi *trending issue.*

3.1.4 Ekstraksi Trending Issue



Gambar 3.7 Alur Ekstraksi *Trending Issue*

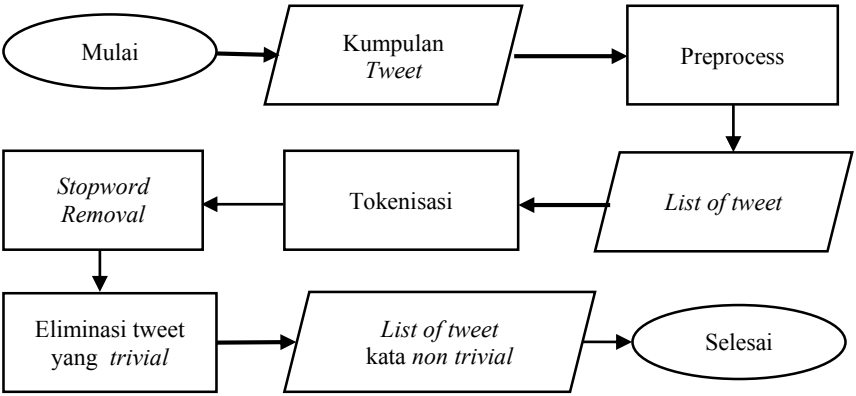
Proses ekstraksi *trnding issue* dilakukan berdasarkan kata *non trivial* hasil *autocorrelation coefficients*. Kata yang dianggap penting dari hasil ekstraksi *tweet* kemudian diseleksi berdasarkan *trending issue*. Proses ini berfungsi untuk mengetahui isu yang

sedang berkembang dari *tweet – tweet* tersebut. Gambar 3.7 menunjukkan proses ekstraksi *trending issue.* Langkah- langkah yang akan dilalui adalah pembuangan kata kunci pada *tweets,* pengelompokan *tweets* berdasarkan kata kunci dengan menggunakan algoritma k-medoids, melakukan ekstraksi *issue* yang didapat dari hasil pengelompokan *tweets*, dan yang terakhir melakukan pembobotan *issue* [2].

- **Pembuangan Kata Kunci *Trivial* pada *Tweet***

  Tweet yang digunakan merupakan *tweet* berita hasil *crawling* yang diambil pada waktu yang bersamaan atau pada tanggal yang sama dengan pengambilan *tweet* berdasarkan lokasi. Hal ini bertujuan untuk menyelaraskan *tweet* berita yang didapat dengan kumpulan *tweet* berdasarkan lokasi. *Tweet* berita ini disimpan didalam sebuah *file* untuk memudahkan proses selanjutnya. *File* yang berisi kumpulan *tweets* berita akan melakukan tahap preproses data yang meliputi penginisialisasian setiap *tweet* dari kumpulan *tweets,* pemenggalan kata dari setiap *tweet* dan pembuangan *stopword.*

  Setelah itu dilakukan proses pembuangan kata *trivial* pada setiap *tweet* berdasarkan kata *non trivial* yang sudah didapat dari proses *autocorrelation wavelet coefficients.* Kata *non – trivial* atau kata penting akan dibandingkan dengan kata pada *tweet* berita. Apabila kata yang berada didalam *tweet* berita tidak mengandung kata kunci *non trivial* maka kata tersebut akan dibuang atau di eliminasi, seperti Gamba 3.8. Eliminasi kata *trivial* atau kata biasa bertujuan untuk mendapatkan kata – kata *non trivial* pada *tweet* berita yang sesuai dengan kata *non – trivial* hasil proses *autocorrelation wavelet coefficients.* Hasil dari proses pembuangan kata kunci *trivial* pada *tweet* berita ini akan digunakan untuk proses selanjutnya yaitu pengelompokkan *tweets.*



Gambar 3.8 Pembuangan kata – kata trivial dalam *Tweets* menjadi *Tweet* kata penting

- **Pengelompokan *tweets***

  Tweet yang sudah berisi kata kunci *non-trivial* akan dikelompokan berdasarkan kesamaan isi dari *tweet* tersebut seperti yang terlihat pada Gambar 3.7. Pengelompokan *tweet* ini menggunakan algoritma K-medoids *clustering*. Pada tahap penglompokan *tweets* hal pertama yang dilakukan adalah dengan membangun matriks *tweet-by-term* berdasarkan kesamaan kata yang dimiliki oleh pasangan *tweets* dengan baris dan kolomnya berupa *tweets* yang berisi jarak *cosine* antar pasangan *tweets* [2]. Matriks yang sudah terbentuk akan melaluli proses klasterisasi menggunakan algoritma K-Medoids dimana akan dibentuk kelompok *tweet* berdasarkan jarak *cosine*. Penentuan *centroid* akan berpengaruh pada hasil klasterisasi untuk mendapatkan kelompok *tweets* yang sesuai. Hasil dari proses klasterisasi ini berupa kelompok *tweets* dengan *k* adalah *centroid* yang telah ditentukan sebelumnya [2].

- **Ekstraksi *Issue***

  Setiap kelompok *tweet* masing – masing akan menghasilkan kata kunci yang merepresentasikan isu dari

kelompok *tweets* tersebut. Kata kunci dapat dilihat dari frekuensi kemunculan kata pada *tweet.* Oleh karena itu, dilakukan pembobotan menggunakan *Term Frequency (TF)* dan *Inverse Term Frequency (IDF)*. Selain itu isu juga diseleksi berdasarkan *Word Frequency* (WF) dimana kemunculan kata pada seluruh tweet akan diperhitungkan. Bobot setiap kata akan dihitung dan akan diseleksi berdasarkan batas nilai ambang (*threshold).* Setiap kata kunci yang nilainya di atas nilai ambang (*threshold)* akan dipilih sebagai isu. Hasil dari proses ini adalah kata kunci yang merepresentasikan *issue* dari setiap kelompok *tweets.*

- **Pembobotan Issue dengan *Cluster Importance***

Pembobotan *Issue* dengan *Cluster Importance* bertujuan untuk menyeleksi satu *issue* yang paling *trending* atau paling banyak diperbincangkan [2]. Masing – masing isu akan diberi bobot menggunakan konsep *cluster importance* dimana *cluster* akan diurutkan berdasarkan kata penting yang muncul pada *cluster* dengan bobot terbesar. Setiap *issue* atau kata kunci dianggap sebagai sebuah *cluster,* kemudian *issue* akan diurutkan berdasarkan pada bobot *cluster importance.*

Kata kunci yang memiliki frekuensi kemunculan tinggi dari seluruh input *tweets* dan memiliki nilai melebihi batas nilai ambang *(threshold)* dianggap sebagai kata penting yang akan dihitung jumlah kemunculannya pada sebuah *issue.* Bobot dari tiap kata kunci dihitung berdasarkan dari jumlah kemunculan kata kunci tersebut. Seluruh bobot dari kata kunci yang terseleksi dari tiap *issue* dijumlahkan untuk mendapatkan pembobotan *issue* berdasarkan konsep CI yang sesuai dengan persamaan 2.6 *Issue* dengan bobot terbesar yang akan dipilih sebagai *trending Issue* (TI).

3.1.5 Ekstraksi Fitur Berita

Berita didapat dari proses *crawling tweet* berita berbahasa Indonesia berdasarkan *keyword* yang dimasukan *Link* berita yang terdapat pada *tweet* tersebut digunakan untuk mendapatkan dokumen berita yang sebenarnya. Kemudian berita – berita tersebut diseleksi berdasarkan *Trending Issue* dimana *Trending issue* tersebut digunakan sebagai *query* untuk mencari berita yang relevan. Hasil dari proses seleksi adalah sejumlah *n* berita yang relevan terhadap *Trending Issue* $D=\{d_1,d_2,...,d_n\}$

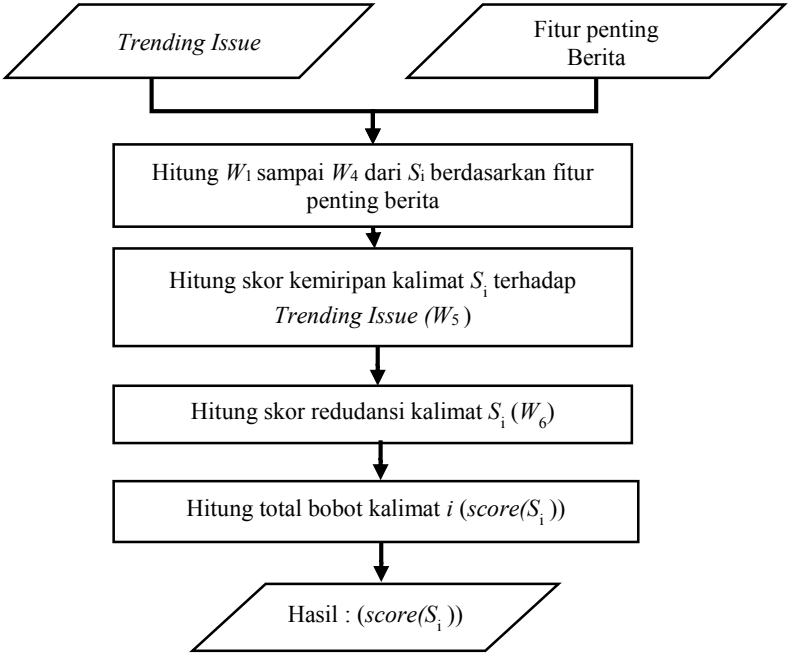
Sejumlah *n* berita yang didapat selanjutnya dijadikan input pada proses seleksi fitur berita. Dari gabungan *n* berita tersebut akan didapatkan kumpulan *m* kalimat $S = \{s_1,s_2,...,s_m\}$ [2]. Ekstraksi fitur berita dilakukan untuk mendapatkan fitur penting dari berita sebagai berikut :

1. *Term Frequency* (TF) yang merepresentasikan banyaknya kata *j* yang muncul pada kalimat *i* (TF*ij*).
2. *Document Frequncy (DF)* yang merepresentasikan banyaknya kalimat yang mengandung kata *j (*DF*i).*
3. *Word Frequency* (WF) adalah jumlah kemunculan kata *j* dari *n* berita (WF_j).
4. Posisi kalimat (*Pos)* adalah letak kalimat *i* pada berita *k* (Pos_{ik}), posisi direpresentasikan dengan indeks yang dimulai dari indeks ke-1 untuk kalimat pertama pada berita dan seterusnya.
5. Judul berita yang diambil dari setiap berita sehingga dari *n* berita akan didapatkan *n* judul berita.

3.1.6 Perhitungan Total Bobot Kalimat

Pembobotan kalimat dihitung berdasarkan 2 (dua) informasi penting yaitu berdasarkan *trending issue* dan fitur penting berita. Oleh karena itu, terdapat 6 teknik pembobotan kalimat antara lain kemiripan kalimat terhadap *Trending Issue* [2], *Word Frequency (WF), Term Frequency Inverse Document Frequency (TF-IDF),* posisi kalimat, dan kemiripan kalimat

terhadap judul berita [12] serta redudansi pada hasil akhir ringkasan.



Gambar 3.9 Algoritma pembobotan kalimat berdasarkan *Trending Issue* dan fitur penting berita [2]

Langkah pertama adalah mengukur kemiripan antara kalimat dengan *WFList* menggunakan *Cosine similarity* dimana *WFList* yang didapat dari sejumlah kata yang memiliki nilai *WF* memenuhi nilai ambang (*threshold), WFList = {*$WF_1,...,WF_k$*}. S* adalah kalimat dan w_1 adalah bobot kalimat berdasarkan *WF*. Sehingga $w_1(s_i)$ adalah nilai kemiripan kalimat s_i terhadap *WFList,* dimana $S=\{s_1,s_2,...,s_m\}$. Persamaan 3.1

$$w_1(s_i) = Sim\,(s_i, WFList) \tag{3.1}$$

Langkah kedua ialah menghitung bobot kalimat berdasarkan bobot TF-IDF dimana bobot tiap kata $TFIDF_{ij}$ didapat dengan menggunakan persamaan 3.2. w_2 merupakan hasil dari seluruh bobot kata *j* yang muncul pada kalimat *i* (s_i), persamaan 3.2. Kemudian pembobotan kalimat berdasarkan posisi dimana kalimat yang posisinya berada diawal dokumen memiliki skor lebih besar dibandingkan dengan kalimat yang posisinya diakhir. *w3* dihitung dengan menggunakan persamaan 3.3 [17].

$$w_2(s_i) = \sum_{j=1}^{n} TFIDF_{ij} \tag{3.2}$$

$$w_3(s_i) = \frac{1}{\sqrt{POS(s_i)}} \tag{3.3}$$

Langkah keempat berdasarkan kemiripan terhadap judul dari berita (w_4). Pembobotan w_4 dihitung menggunakan persamaan 3.4 [12] dimana jumlah kata pada judul yang muncul pada kalimat (*Ntw)* dibagi dengan jumlah seluruh kata yang terdapat pada judul (*T*). Sedangkan untuk pembobotan kalimat berdasarkan *Trending Issue* (w_5) dilakukan dengan mengukur kemiripan kalimat terhadap *TI* juga menggunakan *cosine similarity*. Kalimat yang memiliki skor kemiripan tinggi terhadap *TI* akan dianggap sebagai kalimat penting. Persamaan 3.5

$$w_4(s_i) = \frac{NTW}{T} \tag{3.4}$$

$$w_5(s_i) = Sim\ (s_i, TI) \tag{3.5}$$

Perhitungan bobot yang terakhir terkait dengan redudansi kalimat (w_6) dimana semakin tinggi nilai *overlap* antar kalimat maka nilai redudansinya akan semakin tinggi. Hal ini perlu dilakukan untuk meminimalisir terjadinya redudansi pada hasil akhir ringkasan. Redudansi kalimat diidentifikasi dari kemiripan kalimat ke-*i* (s_i) terhadap kalimat yang lain (s_j) dengan mengadopsi

konsep dari MMR [18]. Dimana *j* sebanyak jumlah kalimat yang ada pada dokumen (*D)*. Persamaan 3.6.

$$w_6(s_i) = Max_{s_j \in D} \frac{2*(s_i \cap s_j)}{s_i \cup s_j} \quad (3.6)$$

Setelah didapatkan bobot *w1* sampai *w*6 langkah berikutnya adalah menghitung total bobot kalimat *i* dengan menggunakan persamaan 3.7. Bobot kalimat yang didapat berdasarkan *w*1 sampai *w*5 seluruhnya dijumlahkan dan dikurangi denga bobot redudansi kalimat *w*6. Seluruh kalimat akan dihitung bobotnya, hasil dari persamaan 3.7 inilah yang akan menjadi total bobot kalimat *i.*

$$score(s_i) = w_1(s_i) + w_2(s_i) + w_3(s_i) + w_4(s_i) + w_5(s_i) - w_6(s_i) \quad (3.7)$$

3.1.7 Penyusunan Ringkasan

Ringkasan berita akan disusun berdasarkan kalimat yang memiliki total bobot terbesar. Semakin besar total bobot yang dimiliki oleh sebuah kalimat makan kalimat tersebut semakin penting. Tahap pertama yang dilakukan sebelum penyusunan kalimat dilakukan adalah mengurutkan kalimat berdasarkan *score* kalimat secara *descending.* Setelah itu akan diseleksi sejumlah *n* kalimat sehingga akan didapatkan jumlah kalimat yang lebih kecil dari *n.* Kalimat yang terseleksi inilah yang akan dikembalikan sebagai hasil akhir ringkasan.

3.2 Perancangan Data

Pada sub bab ini akan dijelaskan mengenai data yang digunakan sebagai masukan perangkat lunak untuk selanjutnya diolah dan dilakukan pengujian sehingga menghasilkan data keluaran yang diharapkan.

3.2.1 Data Masukan

Data masukan yang digunakan didapat melalui proses *crawling* data Twitter berupa teks yang disebut dengan *tweet.* Proses *crawling tweet* menggunakan Twitter API dengan memanfaatkan *library* Python Twitter 1.16.0. Proses pembatasan *crawling* pada *tweet* berbahasa Indonesia dilakukan dengan membatasi proses *stream* menggunakan konfigurasi *geolocation* atau lokasi geografis Negara Indonesia yaitu 94,-11,141,6. Pada Tugas Akhir ini dilakukan beberapa kali proses *crawling* dengan tata cara pengambilan sebagai berikut :

1. Proses *crawling* dengan kata kunci "Sanusi" pada tanggal 2 April 2016 – 25 April 2016 sebanyak 677607 kata pada pukul 06.00 hingga pukul 22.00 WIB dimana pada saat pengambilan data, isu yang sedang berkembang adalah penangkapan ketua Komisi D DPRD DKI Jakarta, M. Sanusi karena kasus korupsi.
2. Proses *crawling* tanpa menggunakan kata kunci dengan hanya dibatasi oleh *geolocation* Indonesia pada tanggal 5 Mei 2016 hingga 13 Mei 2016 selama 24 jam dengan jumlah kata 6898185.
3. Proses *crawling* untuk mendapatkan berita dengan melakukan *crawling tweet* berita dengan kata kunci "sanusi" dari akun *twitter* detik.com pada tanggal 2 April 2016 – 22 April 2016 dengan total 361 berita.
4. Proses *crawling* untuk mendapatkan berita dengan melakukan *crawling tweet* berita dengan kata kunci "puncak" dan "libur" dari akun *twitter* detik.com pada tanggal 12 mei 2016 dengan total 72 berita dimana pada tanggal tersebut merupakan puncak arus libur karena *long weekend.*

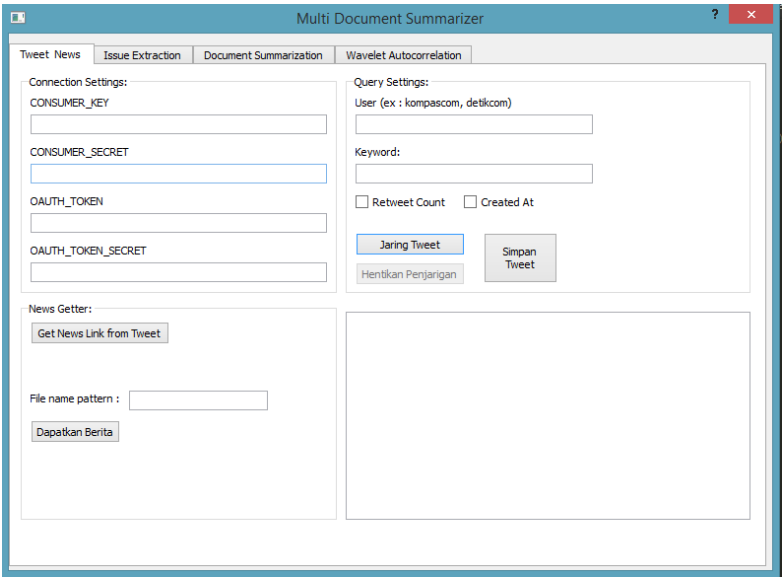
3.3 Perancangan Antar Muka Perangkat Lunak

Pada subbab ini akan dibahas mengenai perancangan antarmuka perangkat lunak yang bertujuan untuk dapat

mempermudah interaksi antara perangkat lunak dengan pengguna. Sistem ini hanya memiliki 4 halaman yang memiliki fungsi yang berbeda-beda.

3.3.1 Tab 1 Tweets News

Tab Tweets News berfungsi untuk mendapatkan berita berdasarkan *user* dan *keyword*. *User* adalah pemilik *tweet* yang menghasilkan *tweet* berita seperti detik.com dan kompas.com. Sedangkan *keyword* merupakan *query* atau kata kunci yang digunakan untuk mendapatkan *tweet* yang mengandung *keyword* tersebut. Berita diambil dari *link* yang terdapat pada *tweet* berita pada proses penjaringan *tweet*. Berita – berita yang didapat kemudian disimpan didalam sebuah folder berita.



Gambar 3.10 Tab Tweet News

Proses pertama yang dilakukan adalah memasukan kata kunci atau *keyword* lalu memilih pemilik akun berita *twitter* yaitu

detik.com atau kompas.com. Pilih ‘jaring tweet’ untuk memulai proses *crawling* dan ‘hentikan penjaringan’ untuk mengakhiri proses *crawling*. Tombol ‘simpan tweet’ digunakan untuk menyimpan kumpulan *tweet* ke dalam sebuah *file* pada suatu folder. Hasil *tweet* yang telah terjaring akan ditampilkan pada kotak hasil.

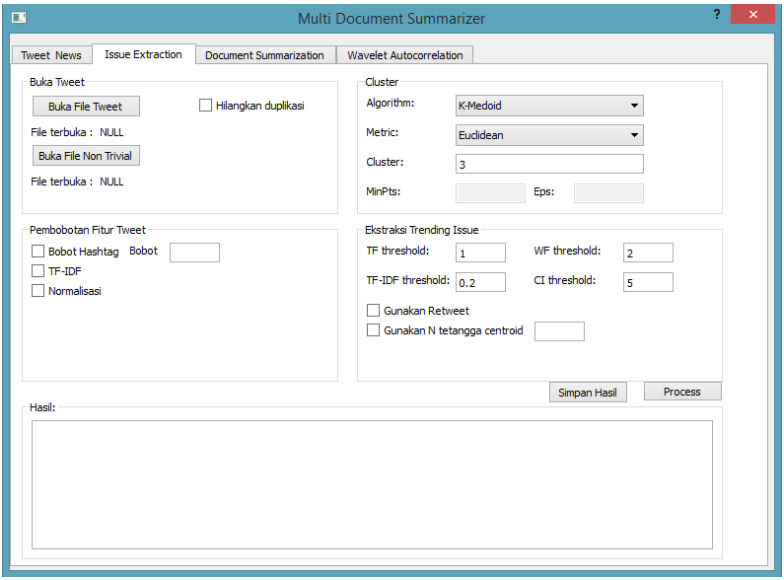
Selanjutnya adalah proses untuk mendapatkan berita. Berita didapat dari *link* pada *k*umpulan *tweet* yang sudah disimpan dengan menekan tombol ‘Get News Link from Tweet’. Setelah *link* didapat maka dilakukan pengambilan berita dari alamat *link* tersebut. Hasil berita yang didapat kemudian disimpan langsung ke dalam suatu folder yang telah ditentukan dengan nama *file* dari berita yang telah diisi terlebih dahulu. Berita yang telah tersimpan akan digunakan dalam proses peringkasan dokumen. Sedangkan kumpulan *tweet* yang disimpan akan digunakan dalam proses eskstraksi *trending issue.*

3.3.2 Tab 2 Issue Extraction

Tab Issue Extraction berfungsi untuk mendapatkan *trending issue* dari kumpulan *tweet* yang didapat dari tab 1 yaitu Tab Tweet News. *Trending issue* didapat dari *file* kumpulan *tweet* yang sudah disimpan hasil dari proses Tab 1 dan *file* kumpulan kata kunci *non trivial* yang merupakan hasil dari proses Tab 4. *Jika* menggunakan kumpulan *tweet* berita saja, maka proses ekstraksi *trending issue* hanya berdasarkan *tweet* berita saja. Sedangkan jika menggunakan kumpulan *tweet* berita dan kumpulan kata kunci *non trivial*, maka proses ekstraksi *trending issue* akan melalui proses pembuangan kata kunci *trivial* hasil proses *autocorrelation wavelet coefficients* terlebih dahulu sehingga diharapkan isu yang didapat lebih spesifik. Hasil dari *Issue Extraction* disimpan ke dalam sebuah *file* untuk proses peringkasan.

Proses pertama yang dilakukan adalah membuka *file tweet* yang sudah disimpan pada folder yang sudah ditentukan sebelumnya pada Tab.1 *Tweets News*. Selanjutnya memilih

pembobotan fitur *tweet* sesuai dengan kebutuhan. Ketiga, memilih algoritma dari klaster yang akan digunakan dan memilih perhitungan jarak yang akan dipakai sebagai contoh proses klastering menggunakan algoritma klaster K-Medoid dengan jarak *euclidean.* Proses yang terakhir adalah menentukan TF *treshold,* TF-IDF *treshold,* WF *treshold* dan CI *treshold* untuk ekstraksi *trending issue.*

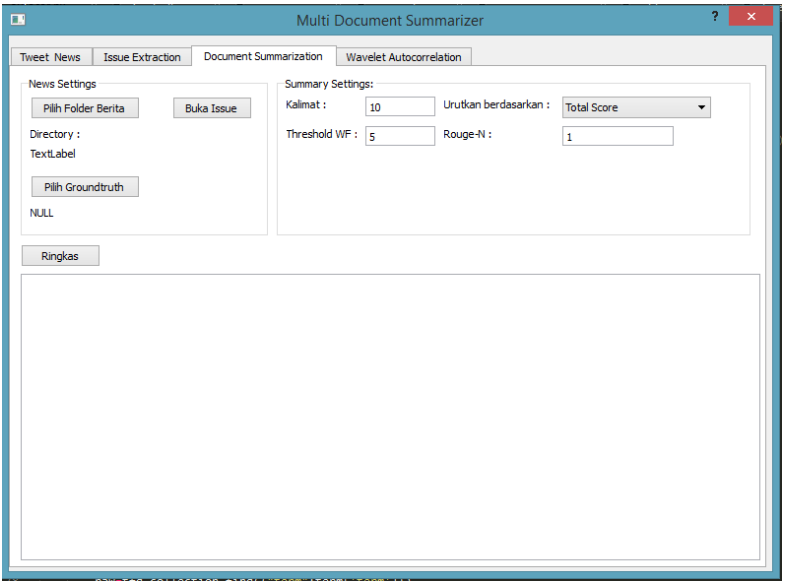


Gambar 3.11 Tab Issue Extraction

3.3.3 Tab 3 Document Summarization

Tab Document Summarization berfungsi untuk melakukan peringkasan berita. Pada bagian ini merupakan tahapan terakhir pada sistem. Berita yang akan diringkas berasal dari kumpulan berita yang sudah disimpan dalam sebuah folder berita dari hasil proses Tab Tweets News. Ringkasan yang dihasilkan berdasarkan isu yang sudah didapat dari hasil Tab Issue Extraction. Hasil ringkasan akan dapat dilihat sesuai dengan jumlah kalimat yang

diinginkan dan diurutkan berdasarkan total skor yang dibahas pada bab 3.1.6.



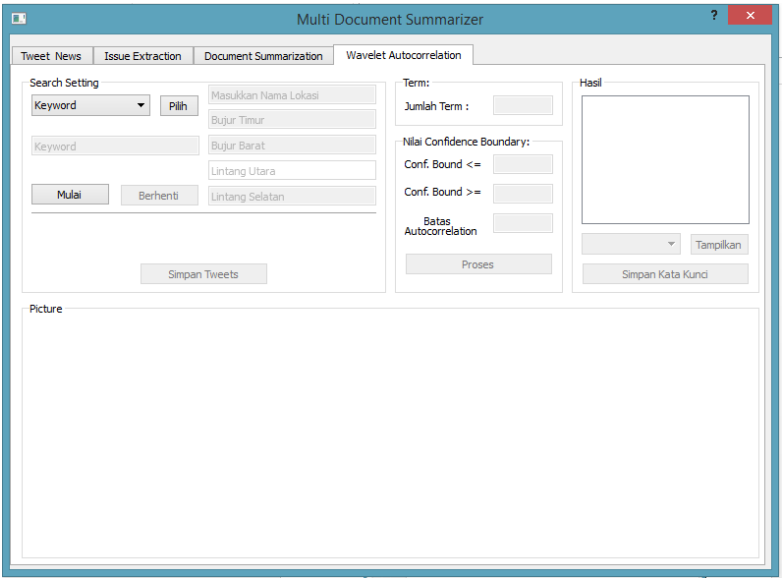
Gambar 3.12 Tab Document Summarization

Tahapan pertama adalah mengambil berita dari folder berita yang sudah dibuat sebelumnya dari proses Tab.1 *Tweets News*. Kemudian memilih isu yang sudah didapat dari proses ekstrasi *trending issue* dari Tab. 2 *Issue Extraction. Groundruth* dapat digunakan apabila mempunya referensi, jika tidak dapat dikosongkan. Selanjutnya adalah menentukan jumlah kalimat dan menentukan *treshold WF* yang diperlukan sebagai hasil peringkasan. Proses terakhir adalah menentukan urutan kalimat yang sudah didapat.

3.3.4 Tab 4 Wavelet Autocorrelation

Tab Wavelet Autocorrelation akan dilakukan proses untuk mendapat kata penting dari hasil *autocorrelation wavelet*

coefficients. Proses *crawling* data dapat dilakukan berdasarkan *keyword,* lokasi, atau *keyword* dan lokasi. Jumlah kata, nilai *confidence boundary* dan batas nilai *autocorrelation* dapat ditentukan oleh pengguna Hasil nilai *autocorrelation* akan ditampilkan pada sub "hasil" dan gambar yang merepresentasi *wavelet* dan *correlogram* akan ditampilkan pada *"picture"*.



Gambar 3.13 Tab Wavelet Autocorrelation

Hal pertama yag dilakukan adalah menentukan proses *crawling* yaitu berdasarkan *keyword* atau lokasi atau berdasarkan *keyword* dan lokasi. Jika berdasarkan *keyword,* maka harus memasukkan kata kunci yang diinginkan dengan lokasi yang sudah ditentukan sebelumnya. Sedangkan berdasarkan lokasi, maka harus menentukan lokasi geografis yang akan diambil data Twitternya dan *keyword* tidak perlu dimasukkan. Apabila memilih keduanya, makan *keyword* dan lokasi harus ditentukan sendiri. *Tweet* hasil *crawling* dapat disimpan didalam sebuah *file* jika

diinginkan, jika tidak *tweets* yang didapat sudah secara langsung tersimpan didalam *database.*

Proses kedua adalah menetukan jumlah *term* yaitu jumlah kata yang akan ditampilkan sebagai representasi dari keseluruhan kata. Selanjutnya menentukan batas *confidence boundary* yang akan mempengaruhi hasil nilai *autocorrelation.* Batas *confindence boundary* berkisari dari 0.05 hinga 0.25 tergantung kebutuhan. Proses terakhir adalah menentukan batas nilai *autocorrelation* yang merupakan batas minimal kata yang memiliki nilai korelasi tinggi sehingga dianggap sebagai kata kunci *trivial.* Proses ini berfungsi untuk menampilkan hasil perhitungan *wavelet coefficient* dan perhitungan *autocorrelation* dalam bentuk gambar.

BAB IV
IMPLEMENTASI

Bab ini membahas implementasi dari perancangan sistem sesuai dengan perancangan yang telah dibuat. Bahasa pemrograman yang digunakan untuk implementasi sistem adalah bahasa pemrograman Python dengan database MongoDB.

4.1 Lingkungan Implementasi

Lingkungan implementasi sistem yang digunakan untuk mengembangkan tugas akhir memiliki spesifikasi perangkat keras dan perangkat lunak seperti ditampilkan pada Tabel 4.1.

Tabel 4.1 Lingkungan Implementasi Perangkat Lunak

Perangkat	Jenis Perangkat	Spesifikasi
Perangkat Keras	Prosesor	Intel Core i7-3517u
	Memori	4GB
Perangkat Lunak	Sistem Operasi	Windows 8.1
	Perangkat Pengembang	1. Python 2.7 2. mongodb

4.2 Implementasi Proses

Pada sub bab implementasi ini menjelaskan mengenai pembangunan perangkat lunak secara detail dan menampilkan kode sumber yang digunakan mulai tahap *preprocessing* hingga peringkasan berita.

4.2.1 Implementasi Pengambilan Data Twitter

Pengambilan data dari Twitter diperlukan otentifikasi pada Twitter API dimana Pengguna Twitter API harus mendaftar menjadi developer Twitter terlebih dahulu pada situs https://dev.twitter.com/ dengan *sign in* menggunakan akun Twitter.

Pada saat mendaftar, pengguna akan mendapatkan 4 buah key yaitu *customer key, consumer secret, acces token,* dan *access token* secret sebagai syarat *authentication* untuk mengakses data yang dimiliki oleh Twitter. *Library* yang digunakan adalah *Twitter.*

```
1  import twitter
2  import json
3  def oauth_login():
4   CONSUMER_KEY         = '..............'
5   CONSUMER_SECRET      = '..............'
6   OAUTH_TOKEN          = '..............'
7   OAUTH_TOKEN_SECRET   = '..............'
8   auth = twitter.oauth.OAuth(OAUTH_TOKEN,
    OAUTH_TOKEN_SECRET,CONSUMER_KEY,
    CONSUMER_SECRET)
9   twitter_api = twitter.Twitter(auth=auth)
10  return twitter_api
```

Kode Sumber 4.1 Kode untuk mengambil data Twitter

4.2.2 Implementasi Tahap Penyimpanan Data

Data Twitter yang sudah diambil disimpan di dalam database MongoDB. Penggunaan *library Pymongo* diperlukan untuk mengakses MongoDB melalui *script* di Python. Hal yang dilakukan pertama kali adalah membuat koneksi database baru pada Mongodb. Koneksi pada MongoDB disebut dengan *Client.* Setelah koneksi terjadi, maka dilakukan pembuatan database dengan memberikan nama pada database yang diinginkan. Pada Tugas Akhir ini nama database yang digunakan adalah "tugasakhir' dengan 3 tabel atau pada MongoDB disebut *collection.*

```
1  client     = pymongo.MongoClient('localhost',
   27017);
2  db         = client['tugasakhir']
3  collection = db.q
```

```
4 | tf_collection   = db["tf."+q]
5 | term_detail     = db["tfd."+q]
```

Kode Sumber 4.2 Potongan Kode untuk membuat database MongoDB

4.2.3 Implementasi Teks Preprocessing

Subbab ini membahas mengenai implementasi tahap teks *preprocessing* dimana pada tahapan ini terdapat 3(tiga) proses yang dilakukan yaitu proses segmentasi, *stopword removal,* dan *stemming*. Semua proses dilakukan secara langsung sebelum data hasil *crawling* disimpan di dalam *collections* atau tabel pada database MongoDB dalam format *json*.

Hal pertama yang dilakukan ada proses segmentasi dengan cara menghilangkan semua tanda baca sehingga yang tersisa hanyalah huruf dan angka saja. Setelah itu masing – masing kata dipisahkan berdasarkan spasi dan disimpan di dalam *array*. Proses selanjutnya adalah pembuangan *stopword* dimana daftar kata yang akan dibuang sudah disimpan didalam *file* sehingga dapat dipanggil menggunakan fungsi yang telah dibuat. Proses terakhir adalah *stemming* yaitu proses perubahan kata menjadi bentuk dasar.

```
1  | # fungsi untuk menghapus stopword
2  | def get_stopword():
3  |     with open('lib\stopword_list.csv') as
   | csvfile:
4  |         reader = csv.DictReader(csvfile)
5  |         global stopword
6  |         stopword=[]
7  |         for row in reader:
8  |             stopword.append(row['kata'])
9  |
10 | #fungsi untuk mengambil semua kata di dalam
   | kamus
11 | def get_all_kata():
```

```
12      with open('lib\kamus.csv') as csvfile:
13          reader = csv.DictReader(csvfile)
14          global daftar_kata
15          daftar_kata=[]
16          for row in reader:
17              daftar_kata.append(row['kata'])
18
19  # fungsi untuk memeriksa kata pada tweet
    terdapat di dalam kamus atau tidak
20  def get_kata(kata):
21      if kata in daftar_kata:
22          return 1
23      else:
24          return 0
25
26  def potong_akhiran(kata):
27      kata        = kata.lower();
28      akhiran1    = get_akhiran_1();
29      akhiran2    = get_akhiran_2();
30      akhiran3    = get_akhiran_3();
31      akhir       = [None] * 3
32      akhir[0]    = ""
33      akhir[1]    = ""
34      akhir[2]    = ""
```

Kode Sumber 4.3 Potongan Kode Teks Preprocessing

4.2.4 Implementasi Perubahan Data Teks Menjadi Data Frekuensi Kata Per Periode

Setelah tahap *preprocessing,* dilakukan perhitungan frekuensi kemunculan kata pada setiap *tweet* yang mengandung kata tersebut. Kata yang dihitung adalah kata yang memiliki karakter lebih dari 3 huruf dan disimpan pada *collection term frekuensi (tf_collection)*. Setiap kata yang masuk akan disimpan waktu kemunculannya dan disimpan pada *collection term detail (tfd_collection).*

```
#generate term frequency
words = text_clean.split()
for current_word in words:
if len(current_word) > 3 :
     tf_collection.update
({'term':current_word}, {'$inc': {'count': 1}},
upsert=True)
     term_det =  [{"term":""+current_word+"",
"created_at":""+str(utc_created_at)+""}]
     term_detail.insert(term_det)
```

Kode Sumber 4.4 Potongan kode perhitungan Term Frekuensi dan Waktu

Kata – kata yang disimpan pada *tf_collection* kemudian diurutkan berdasarkan jumlah frekuensi kemunculan tertinggi. Kemudian kata diurutkan berdasarkan waktu kemunculan berdasarkan data yang sudah disimpan pada *tfd_collection*. Kemunculan kata yang sudah diurutkan berdasarkan waktu kemunculan kemudian dikelompokan per periode dalam hal ini 60 menit. Proses terakhir adalah mengurutkan kemunculan kata per 60 menit berdasarkan waktu kemunculan.

```
Semuaterm   =
tf_collection.find().sort([("count",pymongo.DESC
ENDING)])
```

Kode Sumber 4.5 Kode untuk mengurutkan kata berdasarkan Term Frekuensi

```
def get_key(d):
    # group by 60 minutes
    k = d + timedelta(minutes=-(d.minute %
int(60)))
    return datetime(k.year, k.month, k.day,
k.hour, k.minute, 0)
for term in semuaterm:
       print term['term'] # menampilkan term
       list_topterm.append(term['term'])
```

```
8
raw=tfd_collection.find({"term":term['term']})
9       if raw is not None:
10      #array created at
11          time_set=[]
12          for tweet in raw:
13              time_set.append (datetime.strptime
(tweet['created_at'],"%Y-%m-%d %H:%M:%S"))
14          g = groupby(sorted(time_set),
key=get_key)
15  result  =[]
16  table   =[]
17  for key, items in g:
18      temp=[]
19      for item in items:
20          temp.append(key)
21      result.append(len(temp))
22
table.append({"waktu":str(key),"count":len(temp)})
23  #untuk mengetahui banyakya kemunculan kata per
waktu dengan tambahan informasi waktu
24      print "table : " + str(table)
25  #untuk mengetahui kemunculan kata per waktu tanpa
informasi  waktu ex [4,3,5,7,...,5]
26      print ("result : " + str(result))
```

Kode Sumber 4.6 Potongan kode untuk membuat Frekuensi kemunculan kata per 60 menit

4.2.5 Implementasi Autocorrelation Wavelet Coefficients

Data frekuensi kemunculan kata per periode digunakan sebagai data untuk transformasi *wavelet.* Proses transformasi *wavelet* menggunakan *Discrete Wavelet Transformation* dengan jenis *mother wavelet* yang dipilih adalah *Coiflet* atau yang disebut juga sebagai *wavelet coifman.* Transformasi *wavelet* berawal dari pembentukan sinyal yang berasal dari data frekuensi pada interval waktu tertentu, dekomposisi sinyal menggunakan *wavelet* berjenis

coiflet dengan dekomposisi level 1 (satu) dan pembentukan *wavelet* yang menghasilkan *coefficient approximately* dan detail. Nilai koefisien yang digunakan adalah *coefficient approximately* yaitu cA. Perubahan data tersebut menjadi *wavelet coefficient* menggunakan *library* pada Python yaitu *pywt*.

```
1  # generate wavelet
2  if(len(result)>1):
3     cA, cD = pywt.dwt(result, 'coif1')
4     wavelet=[]
5     for data in cA:
6         wavelet.append(data)
```

Kode Sumber 4.7 Potongan kode untuk menjadi *wavelet coefficient*

Hasil *wavelet coefficient* yang sudah didapat digunakan untuk perhitungan *autocorrelation*. Proses perhitungan korelasi *coefficient* menggunakan fungsi *st.stattools.acf* dengan parameter masukan berupa *wavelet coefficient* dan *nlags* atau jumlah berapa kali pergeseran dari *wavelet.* Perhitungan *autocorrelation* dilakukan dengan cara menghitung korelasi antara *coefficient* sebuah *wavelet* dengan *coefficient wavelet* itu sendiri. Setelah menghasilkan nilai korelasi dari perhitungan *autocorrelation* maka dilakukan proses penghitungan jumlah korelasi yang lebih besar dan lebih kecil dari nilai negatif dari *confidence boundary* yang telah ditentukan. Hasil akhir dari proses penghitungan *autocorrelation* adalah berupa nilai yang akan digunakan sebagai penentuan apakah sebuah kata kunci merupakan kata kunci *trivial* atau bukan.

```
1  For data in
   st.stattools.acf(result,nlags=len(result)):
2      if float(data) >= ld:
3         autocorr_count+=1
4      if float(data) <= kd*-1:
5         autocorr_count+=1
6      autocorr.append(data)
```

```
7       {"waktu":str(key),"count":len(temp)}
8   autocorr_res.append({"term":term['term'],"count":
    autocorr_count})
```

Kode Sumber 4.8 Potongan Kode Autocorrelation Function

4.2.6 Implementasi Ekstraksi *Trending Issue*

Ekstraksi *trending issue* dilakukan untuk mendapat isu dari *tweet* yang sudah didapat dengan melakukan proses klasterisasi dan ekstraksi kata. *Tweet* berita disimpan kedalam sebuah *file,* kemudian dilakukan proses tokenisasi dari kumpulan *tweet* menjadi *list of tweets*. Setalah itu dilakukan proses pembuangan *stopword* dari *list of tweets* dan pembuangan kata – kata yang dianggap biasa (*trivial*) hasil *autocorrelation wavelet coefficient.* Sehingga setiap *tweet* dari *list of tweets* tersebut hanya berisi kata penting (*non-trivial*) saja.

```
1   def stopword_removal(tokens, stopwords):
2       result = []
3       for token in tokens:
4           result.append([ tok for tok in token if
    not stopwords.get(tok, False)])
5       return result
```

Kode Sumber 4.9 Potongan kode untuk proses *stopword removal*

```
1   def do_cluster(self, tweets, retweet_count):
2       tokenized_tweet, hastag_terms =
    tweet_preprocessing(tweets)
3       stopwords = load_stopwords()
4       tokenized_tweet =
    stopword_removal(tokenized_tweet, stopwords)
5       non_trivial = load_nontrivial()
6       tweet_nontrivial =
    get_nontrivial(tokenized_tweet, non_trivial)
```

Kode Sumber 4.10 Potongan Kode preproses untuk pembuangan kata biasa

Setiap *tweet* pada *list of tweets* dihitung *Term Frequency – Inverse Document Frequency (*TF-IDF) untuk mengetahui bobot dari setiap kata pada masing – masing *tweet.* Setelah bobot TFIDF didapat maka proses selanjutnya adalah proses klasterisasi. Proses klasterisasi dilakukan dengan cara membentuk matriks *tweet-by-term* dari *list of tweets* yang didapat dari proses sebelumnya.

```
1    tf_feat, terms = get_tfidf(tweet_nontrivial,
     get_terms=True)
2    cluster_id = KMedoids(tf_feat, distance =
     cosine_distance,  k = klaster).get_cluster_id()
```

Kode Sumber 4.11 Potongan kode untuk membentuk matriks

Matriks yang terbentuk dikelompokkan berdasarkan kesamaan isi menggunakan algoritma K-Medoids yang sudah dijelaskan pada subbab *2.5* dan dihitung berdasarkan jarak *cosine*. Penentuan *centroid* akan berpengaruh pada hasil klasterisasi untuk mendapatkan kelompok *tweets* yang sesuai. Pada implemnetasinya, nilai *k* yang sudah ditentukan di awal adalah 3 (tiga). Hasil dari proses klasterisasi ini berupa kelompok *tweets* dengan *k* adalah *centroid.* Hasil klasterisasi di evaluasi menggunakan *silhouette coefficients.* Penjelasan tentang *silhouette coefficients* berada pada subbab 2.8.1.

```
1   def get_tfidf(docs, get_terms = False,
    normalize = False):
2       N = float(len(docs))
3          terms = set()
4          for doc in docs:
5                terms = terms.union(set(doc))
6       df = dict.fromkeys(terms, 0)
7       for doc in docs:
8           for term in terms:
9           df[term] += int(term in doc)
10      matrix = []
11      CONST = 0
12      if N == 1: CONST = 1
```

```
13      for doc in docs:
14          line = dict.fromkeys(terms, 0)
15          for term in terms:
16              line[term] = doc.count(term) *
    np.log(N / df[term] + CONST)
17          matrix.append(line.values())
18      matrix = np.array(matrix, dtype=np.float64)
19      if normalize:
20         for i in range(len(docs)):
21             temp_sum = np.sum(matrix[i, :] ** 2)
22             if temp_sum != 0:
23                matrix[i, :] /=
    np.sqrt(np.sum(matrix[i, :] ** 2))
24      if get_terms:
25          return matrix, dict.fromkeys(terms,
    0).keys()
26      else:
27          return matrix
```

Kode Sumber 4.12 Potongan kode untuk melakukan perhitungan TFIDF

```
1   def cosine_distance(a, b, weigth = []):
2       if len(weigth) == 0:
3           weigth = np.array([1] * len(a))
4       c_sim = np.sum(a * b * weigth) /
    (np.sqrt(sum(a ** 2)) * np.sqrt(sum(b ** 2)))
5       return 1 - c_sim
```

Kode Sumber 4.13 Potongan Kode Cosine Distance

```
1   class KMedoids:
2       def __init__(self, features, distance =
    cosine_distance, k = 3):
3       self.distance = distance
4       distances = distance_matrix(features,
    distance)
5       while True:
6          self.cluster_id, self.medoid_id =
    fast_kmedoids(features, distances, k = k)
```

```
7       for doc in docs:
8           for term in terms:
9           df[term] += int(term in doc)
10      matrix = []
11      CONST = 0
12      if N == 1: CONST = 1
13      for doc in docs:
14          line = dict.fromkeys(terms, 0)
15          for term in terms:
16              line[term] = doc.count(term) *
    np.log(N / df[term] + CONST)
17          matrix.append(line.values())
18      matrix = np.array(matrix, dtype=np.float64)
19      if normalize:
20         for i in range(len(docs)):
21             temp_sum = np.sum(matrix[i, :] ** 2)
22             if temp_sum != 0:
23                matrix[i, :] /=
    np.sqrt(np.sum(matrix[i, :] ** 2))
24      if get_terms:
25          return matrix, dict.fromkeys(terms,
    0).keys()
26      else:
27          return matrix
```

Kode Sumber 4.14 Potongan kode Algoritma K-Medoids

```
1   def silhouette(features, cluster_id):
2       total = 0
3       n = len(np.unique(cluster_id))
4       for i, feat in enumerate(features):
    k.minute, 0)
5           inner = 0
6           outer = np.inf
7          for j in range(n):
8              dist = euclidean_distances(feat,
    features [cluster_id == j])[0]
9          if j == cluster_id[i]:
10             inner += dist.mean()
```

```
11          else
12              outer = min(outer, dist.mean())
13          total += ((outer - inner) / max(outer,
    inner) if max(outer, inner) != 0 else 0)
14      return total / len(features)
```

Kode Sumber 4.15 Potongan kode penghitungan *Silhouette coefficients*

Proses ekstraksi isu dilakukan pada masing – masing klaster. Isu merupakan kata yang ada pada klaster yang diseleksi berdasarkan kombinasi nilai parameter. Setiap kata di klaster berdasarkan empat parameter yaitu *Term Frequency* (TF), *Term Frequency – Inverse Document Frwquency* (TF-IDF), *Word Frequency* (WF) dan *Cluster importance* (CI). Kemudian kata dipilih berdasarkan masing –masing nilai ambang (*threshold)* dari 4 parameter tersebut yang telah ditentukan.

Setiap kata yang terpilih diambil dari klaster terpenting, jika nilai *Cluster importance* untuk semua klaster sama dengan 0, maka klaster akan dipilih berdasarkan kata terbanyak yang muncul. C*luster importance* dihitung dengan menggunakan logaritma dengan syarat kata tertentu pada klaster yang memiliki minimal kemunculan *threshold* CI . Setiap *issue* atau kata kunci dianggap sebagai sebuah *cluster,* kemudian *issue* akan diurutkan berdasarkan pada bobot *cluster importance.*

```
1   def get_tf(docs, get_terms = False):
2       terms = set()
3       for doc in docs:
4           terms = terms.union(set(doc))
5       matrix = []
6       for doc in docs:
7           line = dict.fromkeys(terms, 0)
8           for term in terms:
9               line[term] = doc.count(term)
10          matrix.append(line.values())
```

```
11      if get_terms:
12          return np.array(matrix),
    dict.fromkeys(terms, 0).keys()
13      else:
14          return np.array(matrix)
    (tweet['created_at'],"%Y-%m-%d %H:%M:%S"))
```

Kode Sumber 4.16 Potongan kode penghitungan TF

```
1   for i in range(N_CLUSTER):
2       cluster_tf, cluster_terms =
    get_tf(non_trivial[cluster_id == i].tolist(),
    get_terms = True)
3       cluster_tfidf =
    get_tfidf(non_trivial[cluster_id == i].tolist(),
    normalize = True)
4       cluster_wf =
    np.array(get_wf(non_trivial[cluster_id ==
    i].tolist()).values())
5       selected = np.argwhere(np.logical_and
    (np.logical_and ((cluster_tfidf.max(axis = 0) >
    TFIDF), cluster_wf > WF), cluster_tf.max(axis =
    0) > TF)).T[0]
6       keywords.extend([cluster_terms[idx] for idx
    in selected])
7
    self.result_dict['result'][i]['selected_terms']
    = [cluster_terms[idx] for idx in selected]
8       if len(self.result_dict['result'][i]
    ['selected_terms']) > t_max:
9           t_max = len(self.result_dict['result'][i]
    ['selected_terms'])
10          best_cluster = i
11      self.result_dict['result'][i]['members'] =
    cluster_tf.shape[0]
12
    self.result_dict['result'][i]['cluster_terms'] =
    cluster_terms
13  max_score = 0
14  for i in range(N_CLUSTER):
15      term_clust = []
16      for trm in non_trivial[cluster_id == i]:
```

```
17          term_clust.extend(trm)
18      score = 0
19      for keyword in keywords:
20          cou = term_clust.count(keyword)
21          if cou > CI_THRES:
22             score +=
np.log(term_clust.count(keyword) + 1)
23      if max_score < score:
24          max_score = score
25          best_cluster = i
26      self.result_dict['result'][i]['score'] =
score
27  self.selected_terms_final =
    self.result_dict['result'][best_cluster]['select
    ed_terms']
```

Kode Sumber 4.17 Potongan kode Ekstraksi Issue

4.2.7 Implementasi Ekstraksi Fitur Berita

Kumpulan berita didapat dari proses c*rawling tweet* berita dilakukan berdasarkan kata kunci (*keyword)* dan disimpan dalam sebuah *file* yang berisi kumpulan *tweet* berita tersebut. Berita diambil dari *link* yang terdapat pada masing-masing *tweet* sehingga menghasilkan sejumlah *n* berita yang disimpan ke dalam sebuah folder. Kumpulan berita ini yang digunakan untuk proses peringkasan dokumen.

Ekstraksi fitur berita dilakukan dengan cara mendapatkan fitur penting dari berita dengan melakukan penghitungan *Term Frequency* (TF), *Document Frequency (DF)* dan *Word Frequency* (WF), penentuan posisi kalimat (*Pos)* dan pengambilan judul berita dari setiap berita. Penghitungan TF dilakukan pada banyaknya kata *i* yang muncul pada kalimat *j*. DF dihitung berdasarkan kemunculan kata pada seluruh dokumen berita. Sedangkan WF merupakan jumlah kemunculan kata yang muncul pada keseluruhan dokumen berita.

```
total_sentences = []
weight = [[] for i in range(6)]
total_weight = []
for i, news in enumerate(file_list):
         title, content = open_news(news)
         date = get_news_date(news)
         sentences = get_sentences(content)
         total_weight.extend([[0, news, j + 1, s, date] for j, s in enumerate(sentences)])
         tok_sentences, _ = tweet_preprocessing(sentences)
         tok_sentences = stopword_removal(tok_sentences, stopwords)
         tf, terms = get_tf(tok_sentences, get_terms = True)
         tf_idf = get_tfidf(tok_sentences, normalize=True)
         wflist = get_wf(tok_sentences, thres = wf_thres)
         t_mat = wflist_to_term_matrix(wflist, terms)
```

Kode Sumber 4.18 Potongan kode untuk mendapatkan fitur berita

```
def get_wf(docs, thres = 0):
    terms = set()
    for doc in docs:
          terms = terms.union(set(doc))
          word_frequency = dict.fromkeys(terms, 0)
    for doc in docs:
          for term in terms:
                word_frequency[term] += doc.count(term)
    if thres != 0:
         for key in word_frequency.keys():
               if word_frequency[key] <= thres:
                     word_frequency.pop(key)
     return word_frequency
```

Kode Sumber 4.19 Potongan kode untuk mendapatkan Word Frequency

4.2.8 Implementasi Perhitungan Total Bobot Kalimat

Total bobot kalimat diambil berdasarkan 6 (enam) perhitungan yang diambil dari fitur penting berita dan *trending issue.* Berikut penghitungan bobot kalimat :

1. W_1 : Menghitung tingkat kemiripan kalimat terhadap *WFList.* Perhitungan WF berasal dari jumlah kemunculan kata pada keseluruhan dokumen berita. *WFList* didapat dari sejumlah kata yang memenuhi nilai ambang (*threshold*). Tingkat kemiripan kalimat terhadap *WFList* diukur menggunakan *cosine similarity.*

```
1  def get_w1(tf, t_mat):
2        w1 = [ ]
3         for tsen in tf:
4            val = 1 - cosine_distance(tsen,
   t_mat)
5            if math.isnan(val):
6                   val = 0
7            w1.append(val)
8       return w1
```

Kode Sumber 4.20 Potongan kode untuk menghitung W_1

2. W_2 : Menghitung bobot *TFIDF* masing – masing kalimat. Setiap kemunculan suatu kata pada suatu kalimat akan dihitung bobotnya.

```
1  def get_w2(tf_idf):
2        w2 = [ ]
3        for tsen in tf_idf:
4            w2.append(tsen.sum())
5       return w2
```

Kode Sumber 4.21 Potongan kode untuk menghitung W_2

3. W_3 : Menghitung bobot berdasarkan posisi kalimat dalam berita. Kalimat yang berada diawal dokumen akan memiliki bobot

yang lebih besar dibandingkan dari kalimat yang berada diakhir dokumen.

```
1  def get_get_w3(sentences):
2        w3 = [ ]
3        for i in range(1, len(sentences) + 1):
4            w3.append(1.0 / math.sqrt(i))
5       return w3
```

Kode Sumber 4.22 Potongan kode untuk menghitung W_3

4. W_4 : Menghitung tingkat kemiripan kalimat dengan judul berita. Setiap kata pada judul yang muncul pada kalimat akan dijumlahkan kemudian dibagi dengan jumlah seluruh kata yang terdapat pada judul.

```
1  def get_w4(title, stopwords, tok_sentences):
2        w4 = [ ]
3        tok_title, _ =
   tweet_preprocessing([title])
4        tok_title =
   stopword_removal(tok_title, stopwords)
5        for tsen in tok_sentences:
6
   w4.append(sentence_similarity(tok_title[0],
   tsen))
7        return w4
```

Kode Sumber 4.23 Potongan kode untuk menghitung W_4

5. W_5 : Menghitung tingkat kemiripan kalimat terhadap *Trending issue*. Pembobotan kalimat diukur menggunakan *cosine similarity*. Kalimat yang memiliki skor tinggi terhadap *trending issue* akan dianggap sebagai kalimat penting.

```
1  def get_w5(trending_issue, tok_sentences,
   weight = 0.6):
2        w5 = [ ]
3        for tsen in tok_sentences:
```

```
4               ttf = get_tf([trending_issue,
    tsen])
5               weight_vector = ttf[0] * weight
6               w5.append(1 -
    cosine_distance(ttf[0], ttf[1],
    weigth=weight_vector))
7       return w5
```

Kode Sumber 4.24 Potongan kode untuk menghitung W_5

6. W_6 : Menghitung tingkat redudansi antar kalimat dalam keseluruhan berita. Hal ini dilakukan untuk meminimalisir terjadinya redudansi pada hasil akhir ringkasan.

```
1   def get_w6(tok_sentences):
2       w6 = [-10] * len(tok_sentences)
3       for i in range(len(tok_sentences) - 1):
4           for j in range(i + 1,
    len(tok_sentences)):
5               val = 2 *
    sentence_similarity(tok_sentences[i],
    tok_sentences[j])
6               w6[i] = max(val, w6[i])
7               w6[j] = max(val, w6[j])
8       return w6
```

Kode Sumber 4.25 Kode untuk menghitung W_6

4.2.9 Implementasi Penyusunan Ringkasan

Penyusunan ringkasan dilihat dari bobot kalimat yang terbesar. Kalimat diurutkan berdasarkan dari total bobot terbesar. Kalimat yang memiliki total bobot terbesar dianggap sebagai kalimat yang penting dan akan dipilih sebagai ringkasan berita. Untuk evaluasi hasil peringkasan digunakan Groundruth menggunakan algoritma ROGUE-1.

```
1   if sorting == 1:
2       total_weight.sort(reverse=True)
```

```
3  elif sorting == 2:
4        total_weight.sort(reverse=True)
5        total_weight = total_weight[:n_sentences]
6  total_weight.sort(key=operator.itemgetter(4,
   1), reverse=True)
7  self.result_dict['summarization'] =
   total_weight[:n_sentences]
8  print len(self.result_dict['summarization'])
```

Kode Sumber 4.26 Kode pengurutan ringkasan

```
1  def rogue_evaluation (list_token_candidate,
   list_token_reference, n = 1):
2        candidate_grams = [ ]
3        for token in list_token_candidate:
4        candidate_grams.extend
   (construct_ngram_feature(token, n))
5        reference_grams = [ ]
6        for token in list_token_reference:
7              reference_grams.extend
   (construct_ngram_feature(token, n))
8        candidate_grams = set(candidate_grams)
9        reference_grams = set(reference_grams)
10       return float (len
   (candidate_grams.intersection(
   reference_grams))) / len(reference_grams)
```

Kode Sumber 4.27 Kode untuk evaluasi ROUGE

BAB V
HASIL UJI COBA DAN EVALUASI

Bab ini berisi penjelasan mengenai skenario uji coba dan evaluasi terhadap perangkat lunak yang telah dikembangkan dari implementasi peringkasan multidokumen berita berbahasa Indonesia menggunakan *autcorrelation wavelet coefficient.* Hasil uji coba didapatkan dari implementasi pada bab 4 dengan skenario yang berbeda. Bab ini berisikan pembahasan mengenai lingkungan pengujian, data pengujian, dan uji kinerja.

5.1 Lingkungan Pengujian

Lingkungan pengujian pada uji coba yang digunakan dalam pembuatan tugas akhir ini meliputi perangkat lunak dan perangkat keras yang digunakan untuk uji coba implementasi peringkasan multidokumen berita berbahasa Indonesia menggunakan *autocorrelation wavelet coefficients.* Lingkungan uji coba menggunakan spesifikasi keras dan perangkat lunak seperti yang ditunjukkan pada Tabel 5.1.

Tabel 5.1 Spesifikasi Lingkungan Pengujian

Perangkat	Jenis Perangkat	Spesifikasi
Perangkat Keras	Prosesor	Intel Core i7-3517u
	Memori	4GB
Perangkat Lunak	Sistem Operasi	Windows 8.1
	Perangkat Pengembang	1. Python 2.7 2. mongodb

5.2 Skenario Uji Coba

Sebelum melakukan uji coba, perlu ditentukan skenario yang akan digunakan dalam uji coba. Melalui skenario ini, perangkat akan diuji apakah sudah berjalan dengan benar dan bagaimana performa pada masing-masing skenario. Dan membandingkan skenario manakah yang memiliki hasil lebih baik. Terdapat 3 macam skenario uji coba, yaitu :

1. Pengujian metode *Autocorrelation Wavelet Coefficients* menggunakan *crawling* data Twitter berdasarkan *keyword* atau kata kunci.
   Pengujian ini bertujuan untuk mengetahui kata yang dianggap penting berdasarkan kaitannya terhadap kata kunci.
2. Pengujian *Autocorrelation Wavelet Coefficients* menggunakan *crawling* data Twitter berdasarkan lokasi geografis Indonesia tanpa menggunakan *keyword* atau kata kunci
   Pengujian ini bertujuan untuk mendapatkan kata penting berdasarkan *confidence boundary* yang telah ditentukan.
3. Pengujian kombinasi parameter untuk proses ekstraksi *trending issue.*
   Pengujian ini bertujuan untuk mendapatkan kombinasi parameter dan hasil klaster yang terbaik menggunakan *silhouette coefficients*
4. Pembandingan hasil ringkasan berita menggunakan pembuangan kata kunci *trivial* hasil *Autocorrelation Wavelet Coefficients* dengan hasil ringkasan berita tanpa menggunakan pembuangan kata kunci *trivial.*
   Pengujian ini bertujuan untuk mengetahui ringkasan terbaik yang dihasilkan berdasarkan nilai Rouge -1.

5.2.1 Skenario Uji Coba 1 : Pengujian Autocorrelation Wavelet Coefficients menggunakan hasil crawling data Twitter berdasarkan keyword atau kata kunci

Pada skenario ini dilakukan uji coba sistem dengan melakukan *crawling* data Twitter berdasarkan *keyword* sebagai inputan. Pengujian ini bertujuan untuk mengetahui kata mana yang dianggap penting berdasarkan kaitannya terhadap *keyword* yang dimasukkan. *Keyword* yang diinputkan sesuai dengan isu yang berkembang. Pada saat pengambilan data, isu yang sedang berkembang adalah penangkapan Ketua Komisi D di DPRD DKI Jakarta, M. Sanusi karena kasus korupsi oleh KPK. Sehingga pada saat pengambilan data, *keyword* yang diinputkan adalah 'sanusi' dengan tujuan mendapatkan *tweet* yang sesuai dengan *keyword* tersebut. Uji coba ini menggunakan 6500 kata dengan kemunculan kata tertinggi dari 677607 *tweets*. Hal ini dilakukan karena perangkat keras tidak mampu mengolah seluruh *tweets* yang didapat dari hasil *crawling*.

Seluruh data *tweets* disimpan di dalam database. Pada saat proses *crawling* dilakukan tahapan teks *preprocessing* secara langsung. Tahapan teks *preprocessing* tersebut adalah segmentasi yaitu pemisahan kata pada *tweet, stopword removal* yaitu pembuangan kata – kata yang dianggap tidak penting dan *stemming* yaitu pengambilan kata dasar pada kata tiap *tweet*. Selain itu dilakukan perhitungan kemunculan setiap kata pada setiap *tweet* yang terjaring serta kemunculan kata tersebut yang juga disimpan didalam database. Jumlah kemunculan tiap kata dihitung dan hasilnya ditampilkan pada Tabel 5.2 Daftar kata yang muncul dengan 30 TF tertinggi.

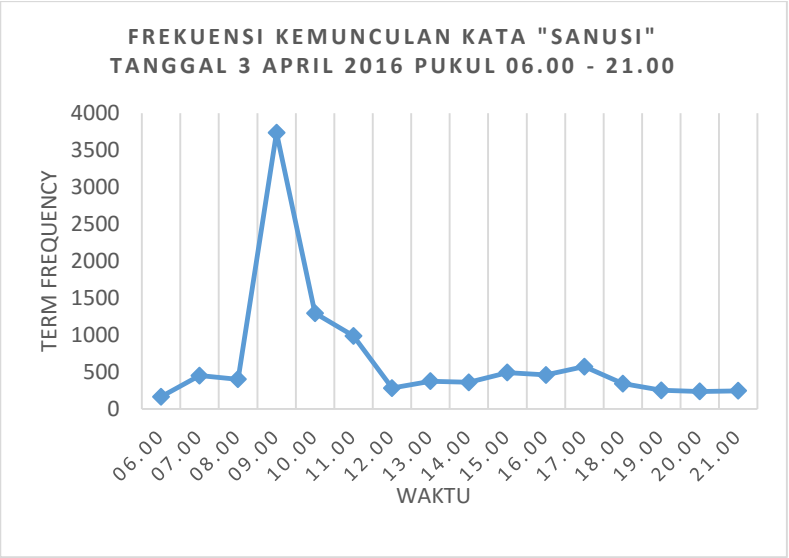
Tabel 5.2 Daftar kata yang muncul dengan 30 TF tertinggi

No	Kata	TF	No	Kata	TF
1	sanusi	92780	16	Rumah	5672
2	suap	17851	17	anggota	5596
3	ahok	16632	18	uang	5100

No	Kata	TF	No	Kata	TF
4	gerindra	15082	19	acara	4634
5	dprd	14099	20	prabowo	4400
6	reklamasi	9135	21	geledah	4084
7	tangkap	7475	22	agung	4038
8	sunny	7430	23	temu	3764
9	jakarta	7286	24	mohamad	3716
10	ketua	6446	25	pecat	3319
11	terima	6186	26	sidik	3300
12	periksa	5913	27	taufik	3040
13	korupsi	5912	28	partai	3012
14	sangka	5797	29	komisi	3010
15	podomoro	5745	30	kait	2932

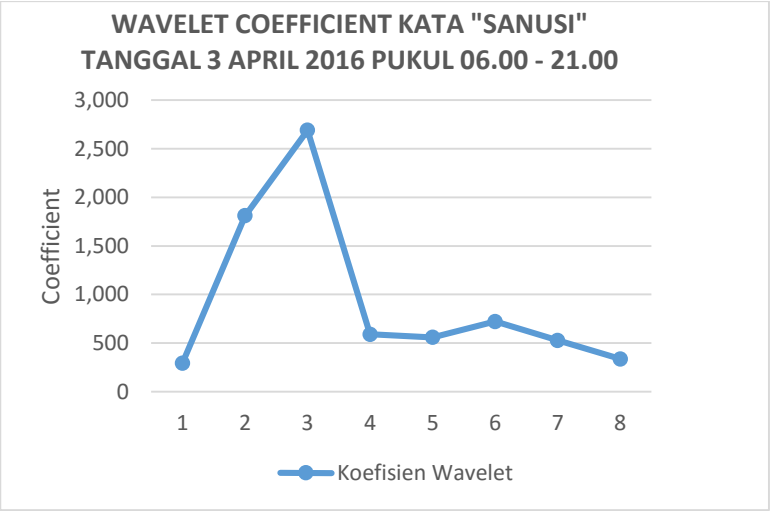
Kata – kata yang muncul memperlihatkan memiliki keterkaitan antar kata – kata nya. Sebagai contoh kata ‘sanusi’ memiliki kaitan terhadap kata ‘suap’, ‘ketua’ dan ‘jakarta’ karena pada berita yang sebenarnya terjadi bahwa M. Sanusi ditangkap karena tersandung kasus suap. Pada saat penangkapan M. Sanusi merupakan ketua dari DPRD di Jakarta. Secara umum kata – kata pada Tabel 5.2 merupakan kata – kata yang memiliki kaitan dengan korupsi.

Setelah data disimpan dan tahap *text processing* sudah dilalui, maka dilakukan proses transformasi data menjadi sinyal *wavelet*. Tahapan pertama yang dilakukan adalah mengelompokan kemunculan kata dalam kelompok periode waktu, dalam hal ini selama 60 menit. Gambar 5.1 merupakan data kemunculan kata ‘sanusi’ per periode pada tanggal 3 April 2016 dari pukul 06.00 – 21.00 WIB. Terlihat bahwa kemunculan kata paling tinggi terdapat pada pukul 09.00 yang muncul sebanyak 3736 kali kemudian turun pada pukul 10.00 dengan kemunculan sebanyak 1294 kali dan pukul 11.00 muncul 988. Setelah pukul 11.00 kemuculan kata “sanusi” berkisar pada 236 – 571 kali. Hal ini merepresentasikan bahwa berita tentang ‘sanusi’ banyak muncul pada rentang waktu pukul 09.00- 11.00.



Gambar 5.1 Grafik kemunculan kata "sanusi"

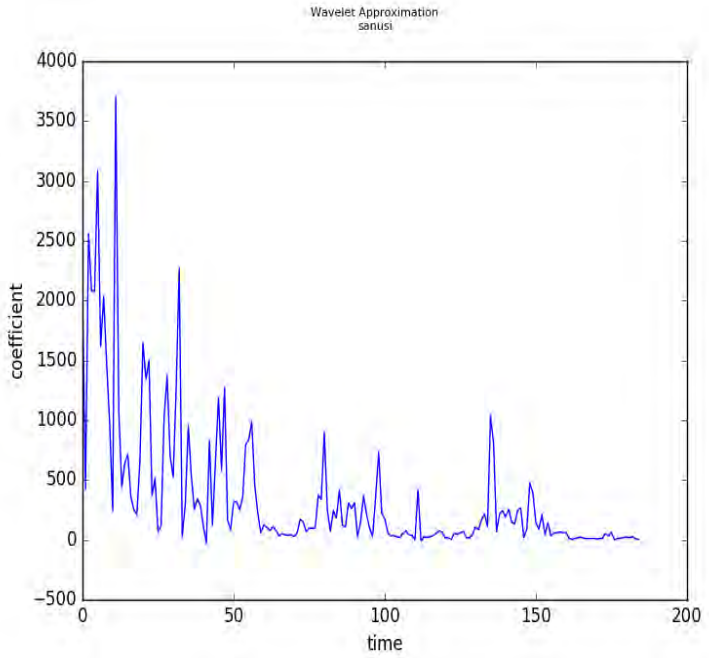
Frekuensi kemunculan kata kunci per periode waktu digunakan dalam proses transformasi *wavelet.* Pada proses ini menggunakan *mother wavelet* jenis *coiflet* menggunakan *library Pywt* yang terdapat pada Python. Proses pembentukan *wavelet* akan menghasilkan *wavelet coefficient* yang digunakan untuk proses perhitungan *autocorrelation*. Proses transformasi *wavelet* menghasilkan dua koefisien yaitu koefisien aproksimasi dan koefisien detail. Koefisien yang digunakan akan koefisien aproksimasi. Nilai *wavelet coefficient* pada kata 'sanusi' tanggal 3 April 2016 pukul 06.00 – 21.00 WIB dapat dilihat pada Gambar 5.2 yang menunjukkan *wavelet coefficient* mengalami kenaikan kemudian turun secara drastis. Hal ini sesuai dengan data pada Gambar 5.1 yang mengalami kenaikan dan turun secara drastis. Hasil *wavelet coefficient* yang diperoleh secara keseluruhan kemudian dipresentasikan secara grafis Gambar 5.3 yang merupakan *wavelet* pada kata kunci 'sanusi'.



Gambar 5.2 Grafik *wavelet coefficient* kata 'sanusi' pada tanggal 3 April 2016 pukul 06.00 – 21.00 WIB

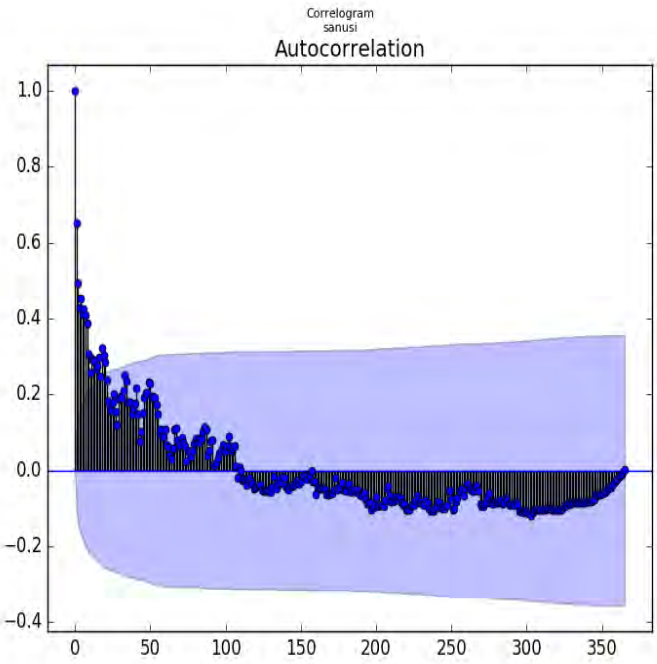
Pada Gambar 5.3 terlihat bahwa kata "sanusi" mengalami penaikan dan penurunan pada waktu – waktu tertentu. Kata 'sanusi' mengalami kenaikan pada waktu – waktu awal dimana pada saat tersebut kata 'sanusi' merupakan kata penting yang memang sedang hangat dibicarakan. Namun semakin lama, kata 'sanusi' mengalami penurunan seiring berjalannya waktu. Sinyal akan megalami kenaikan ketika frekuensi kemunculan kata 'sanusi' tinggi.

Penaikan dan penurunan yang terjadi pada kata 'sanusi' yang secara acak atau tidak teratur menunjukkan bahwa kata "sanusi" merupakan data *non stasioner*. Kemunculan kata tidak teratur tergantung dari waktu karena sinyal *wavelet* yang dihasilkan bervariasi terhadap waktu. Terlihat juga bahwa kata 'sanusi' berulang secara periodik. Proses selanjutnya adalah mendeteksi puncak pada kata 'sanusi'.



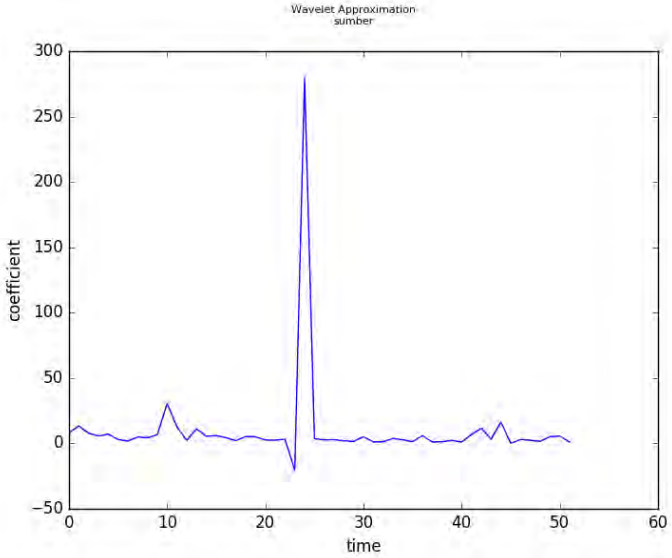
Gambar 5.3 *Wavelet* kata kunci Sanusi

Setelah kata kunci melalui tahap transformasi *wavelet* dalam bentuk *wavelet coefficient,* selanjutnya dilakukan pengamatan dengan mendeteksi kemunculan puncak (*peak)* pada *wavelet.* Pada kata 'sanusi' terdeteksi banyak puncak yang terlihat pada Gambar 5.3 yang mengakibatkan kata 'sanusi' merupakan suatu kata yang mempresentasikan kejadian penting sehingga perlu dilakukan perhitungan *autocorrelation* untuk mengetahui nilai *autocorrelation* dari kata tersebut. Perhitungan *autocorrelation* dilakukan menggunakan *library st.stattools.acf.*



Gambar 5.4 Corelogram kata kunci "sanusi" yang memiliki nilai korelasi yang tinggi dan kedekatan dari tiap Lag membuat kata ini menjadi tidak penting

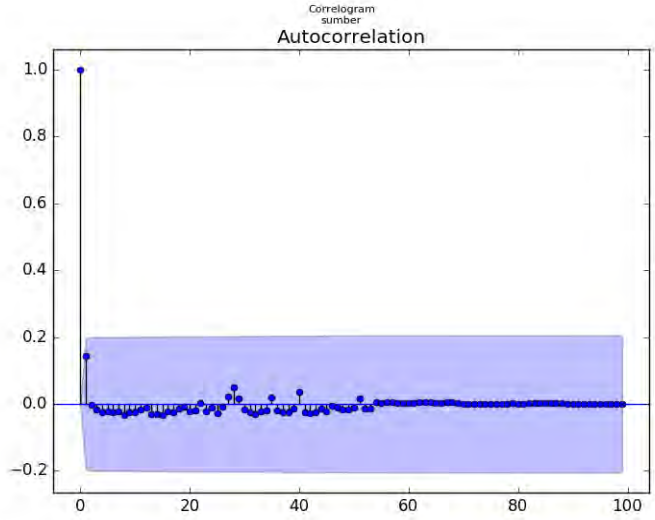
Gambar 5.4 menunjukkan adanya *lag* yang signifikan. Terdapat beberapa *lag* awal pada kata "sanusi" berada diatas ambang batas dari nilai *autocorrelation.* Hasil *correlogram* diatas menunjukkan bahwa kata "sanusi" memiliki korelasi yang tinggi sehingga dapat membuat kata tersebut dianggap sebagai kata *trivial.* Hal ini akan ditegaskan dengan hasil perhitungan jumlah nilai *autocorrelation* yang dihitung berdasarkan *confidence boundary* pada Tabel 5.2 dan Tabel 5.4 dimana kata "sanusi" memiliki jumlah nilai *autocorrelation* yang tinggi.



Gambar 5.5 *Wavelet* kata kunci Sumber

Gambar 5.5 menunjukkan hasil transformasi *wavelet* dalam bentuk *wavelet coefficient* yang memperlihatkan bahwa kata "sumber" merupakan kata yang tidak muncul secara periodik. Terdeteksi satu puncak dari *wavelet* tersebut menunjukkan bahwa kata tersebut mengalami kenaikan kemunculan pada rentang waktu ke-20 hingaa ke-30. Gambar tersebut juga menunjukkan bahwa kata kunci "sumber" memiliki frekuensi kemunculan yang rendah dan tidak berulang secara periodik.

Gambar 5.6 menunjukkan hasil perhitungan *autocorrelation* dari *wavelet coefficient* kata kunci 'sumber'. Tidak ada *lag* yang menunjukkan bahwa *lag* tersebut signifikan. *Lag* pada kata "sumber" berada dibawah ambang batas dari nilai *autocorrelation.* Sehingga dapat dipastikan bahwa kata 'sumber' bukan merupakan kata *trivial* karena *lag* menunjukkan bahwa kata tersebut memiliki korelasi yang rendah.



Gambar 5.6 Correlogram kata kunci Sumber

Perhitungan nilai *autocorrelation* yang digambarkan pada *correlogram* ditunjukkan dengan variasi nilai *confidence boundary*. Penentuan nilai *autocorrelation* dipengaruhi oleh nilai *confidence boundary* yang digunakan sebagai batasan kriteria sebuah kata kunci merupakan kata kunci *trivial* atau tidak. Nilai *confidence boundary* yang digunakan bervariasi, yaitu 0.05, 0.10, 0.15, dan 0.20.

Setiap *confidence boundary* di uji sebagai batas nilai korelasi. Total nilai *confidence boundary* dari hasil korelasi akan dihitung dari nilai korelasi dari setiap kata yang melebihi batas nilai *confidence boundary*. Kriteria penentuan batas nilai *confindence boundary* secara manual dipertimbangkan berdasarkan kedekatan hubungan sebuah kata kunci dengan keterangan waktu atau kata kunci terkait dengan hal-hal yang dilakukan secara jelas pada waktu-waktu tertentu.

Tabel 5.3 Tabel Jumlah Nilai *Confidence Boundary* 0.05 & 0.10

Kata Kunci	*Conf. Bound > 0.05*	Kata Kunci	*Conf. Bound > 0.10*
sanusi	276	sanusi	103
tangkap	197	maling	45
korupsi	163	ahok	42
ahok	142	tangkap	41
suap	137	suci	41
kalo	107	tangkep	40
salah	104	tawur	40
maling	97	tapak	40
news	94	koruptor	38
tangkep	87	kalo	30

Tabel 5.4 merupakan hasil perhitungan nilai korelasi dengan nilai *confidence boundary* 0.05 dan 0.10. Setiap kata memiliki perbedaan jumlah nilai korelasi yang melebihi batas nilai *confidence boundary*. Pada Tabel 5.4 Tabel Jumlah Nilai *Autocorrelation* berdasarkan *Confidence Boundary* 0.15 & 0.20merupakan hasil perhitungan nilai *autocorrelation* dengan nilai *confidence boundary* 0.15 dan 0.20. Dari kedua tabel diatas dapat disimpulan bahwa kata kunci yang memiliki frekuensi kemunculan kata yang tinggi dan berulang akan membuat jumlah nilai *autocorrelation* semakin tinggi. Semakin tinggi nilai *autocorrelation* semakin mengindikasikan bahwa kata tersebut berulang secara periodik.

Tabel 5.4 Tabel Jumlah Nilai *Autocorrelation* berdasarkan *Confidence Boundary* 0.15 & 0.20

Kata Kunci	*Conf. Bound > 0.15*	Kata Kunci	*Conf. Bound > 0.20*
Sanusi	50	sanusi	31
tangkap	37	tangkap	29

Kata Kunci	***Conf. Bound > 0.15***	**Kata Kunci**	***Conf. Bound > 0.20***
Ahok	21	korupsi	13
korupsi	17	koruptor	12
Salah	16	chirpified	12
Suci	16	sensasi	11
keren	16	minister	11
minister	16	salah	10
tangkep	15	bilang	10
suap	14	kalo	10

Pada saat pengamatan yang dilakukan secara manual, kata "sanusi" seharusnya merupakan kata penting karena kata "sanusi" memiliki kaitan dengan *trending issue* yang sedang terjadi tetapi menjadi kata tidak penting karena data kemunculan kata yang dihasilkan. Kata "sanusi' muncul secara terus – menerus sehingga kata tersebut berulang secara periodik yang mengakibatkan kata "sanusi" memiliki nilai korelasi yang paling tinggi. Selain kata "sanusi", kata "tangkap", "korupsi", dan "ahok" yang juga merupakan kata penting menjadi kata yang sering muncul dan memiliki nilai korelasi yang tinggi. Kata – kata tersebut dianggap penting karena kata – kata tersebut merupakan memiliki kaitan dengan kasus korupsi M. Sanusi.

Hal ini terjadi karena *keyword* yang dimasukan terlalu spesifik dan terlalu menujurus kepada isu yang terjadi. *Keyword* yang terlalu spesifik merupakan isu yang seharusnya dianggap sebagai kata penting. Pengujian ini merujuk pada penelitian [2] yang menggunakan *keyword* pada tahap pengambilan *tweets* yang dianggap sebagai isu untuk mendapatkan berita. Tetap *tweet* yang diambil hanyalah *tweet* yang mengandung berita dari akun berita pula.

Proses *crawling* pada pengujian ini mengambil seluruh *tweet* yang mengandung *keyword* dengan kata "sanusi" dengan

harapan kata yang didapat akan terkait dengan *trending issue.* Setiap kemunculan kata "sanusi" terus diambil dan dihitung sehingga membuat frekuensi kemunculannya menjadi lebih tinggi dan berulang secara periodik. Oleh karena itu, penggunaan *keyword* yang spesifik tidak cocok digunakan pada sistem.

5.2.2 Skenario Uji Coba 2 : Pengujian *Autocorrelation Wavelet Coefficients* menggunakan hasil *crawling* data Twitter berdasarkan lokasi geografis Indonesia tanpa menggunakan *keyword*

Pada skenario ini dilakukan uji coba sistem dengan melakukan *crawling* data Twitter berdasarkan lokasi tanpa menggunakan *keyword* sebagai inputan. Pengujian ini bertujuan untuk mendapatkan kata penting berdasarkan *confidence boundary* yang telah ditentukan sesuai dengan kaitannya terhadap isu yang sedang terjadi. Lokasi yang diinputkan sesuai dengan lokasi geografis Indonesia yaitu 6.076912' LU, -11.0074361' LS dan 95.0097069 BB, 141.0195621 BT dengan tujuan mendapatkan *tweet* berbahasa Indonesia. Uji coba ini menggunakan 6500 kata dengan frekuensi kemunculan kata tertinggi dari 6898185 *tweets* dengan lokasi Indonesia. Proses *crawling* dilakukan selama 24 jam dari tanggal 5 Mei – 13 Mei 2014.

Tabel 5.5 Kumpulan kata dengan TF tertinggi

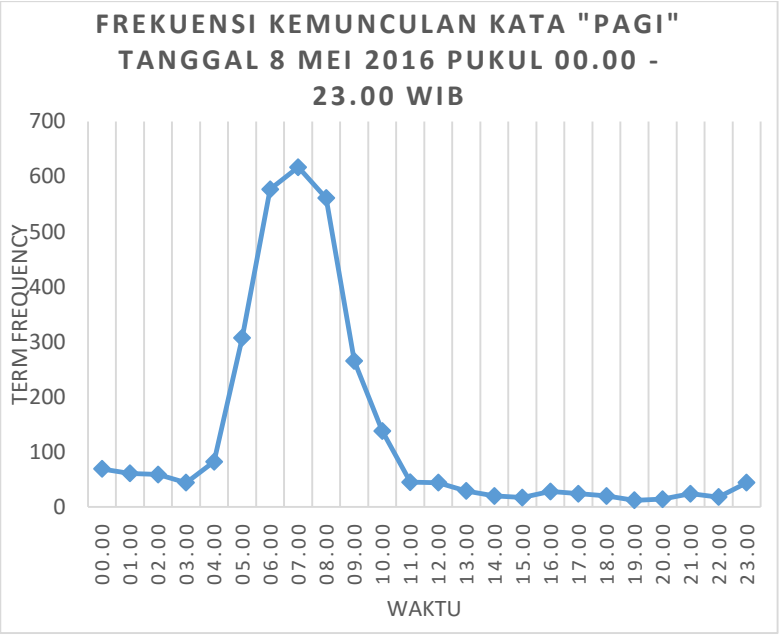
Kata	TF	Kata	TF	Kata	TF
langor	45347	johor	16613	singapore	13696
kuala	45227	time	15955	morning	12932
lumpur	39779	malam	15751	taman	12697
pagi	22766	shah	15352	hotel	12652
makan	22647	indonesia	15203	kerja	12416
happy	20777	love	14919	jakarta	12354
alam	19357	rumah	14143	cinta	12000
orang	18959	good	14118	hahahaha	11909
selamat	18403	hati	14070	kena	11538

Kata	TF	Kata	TF	Kata	TF
jaya	17733	jalan	13991	kota	11199

Proses yang dilakukan sama dengan skenario ujicoba 1 hanya saja *keyword* tidak diinputkan pada proses *crawling* data. *Crawling* data *tweets* menggunakan *library Twitter* dan langsung melakukan tahapan teks *preprocessing.* Selain itu dilakukan juga perhitungan frekuensi kemunculan kata pada setiap *tweets* yang berguna untuk proses transformasi *wavelet.* Seluruh data *tweets* disimpan didalam database MongoDB. Tabel 5.5 menampilkan kata dengan 30 TF tertinggi.

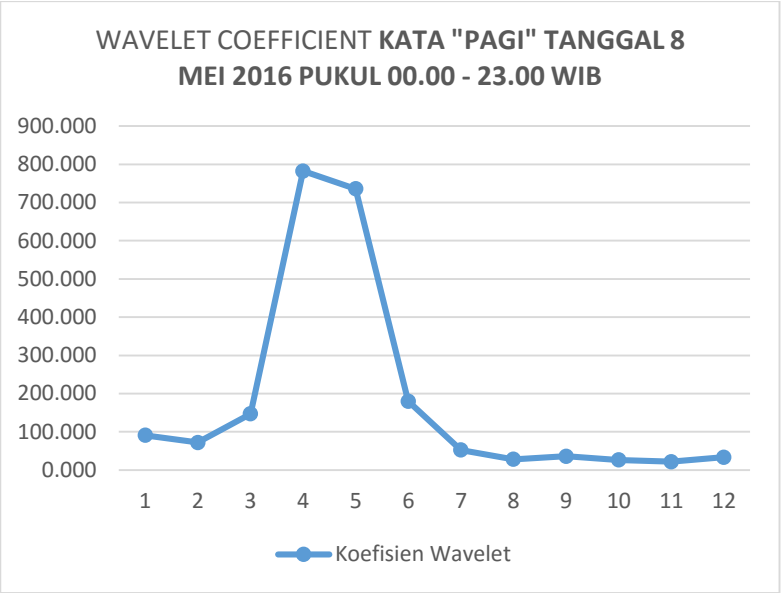
Setelah tahap teks *processing* selesai, proses selanjutnya adalah merubah data teks menjadi data angka untuk transformasi data menjadi sinyal *wavelet* menggunakan *library Pywt.* Data teks berupa frekuensi kemunculan kata dan waktu dirubah menjadi data angka yaitu data frekuensi kemunculan kata per periode waktu seperti yang telah dijelaskan pada subbab 3.1.3. Kemudian dilakukan dekomposisi sinyal yang berasal dari frekuensi kemunculan kata per periode menjadi *wavelet coefficient* menggunakan mother *wavelet coiflet*. *Wavelet coefficient* yang dihasilkan adalah koefisien detail dan koefisien aproksimasi. Tetapi hanya koefisien aproksimasi yang diambil untuk proses perhitungan *autocorrelation*.

Gambar 5.7 menunjukkan frekuensi kemunculan kata "pagi" selama 24 jam. Kemunculan kata "pagi" beranjak naik dari pukul 04.00 hingga pukul 11.00 dan mengalami puncak nya pada pukul 07.00. Pukul 04.00 merupakan waktu dimana orang – orang mulai bangun dan memulai aktifitas paginya. Pukul 07.00 merupakan waktu produktif untuk memulai pekerjaan selain itu pada waktu tersebut merupakan waktu dimana orang aktif membuka media sosial oleh karena itu kata "pagi" dapat mencapai puncaknya. Tentu saja hal ini dianggap wajar karena waktu tersebut adalah waktu pagi hari sehingga kata "pagi" sering muncul.



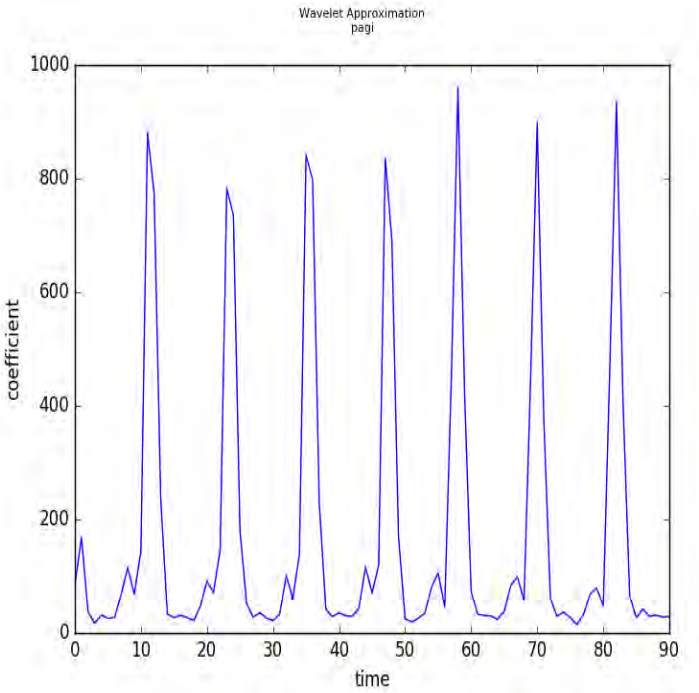
Gambar 5.7 Grafik *Term frequency* dan *wavelet coefficient* kata "pagi" pada tanggal 8 Mei 2016

Data kemunculan kata "pagi' digunakan untuk menghasilkan *wavelet coefficient* yang ditampilkan pada Gambar 5.8. Koefisien transformasi dikembalikan sebagai dua *array* yang mengandung masing - masing koefisien aproksimasi dan koefisien detail. Panjang dari *array* tergantung dari modus ekstensi sinyal yang dipilih, dimana pada Tugas Akhir ini koefisien transformasi setengah dari banyaknya data tergantung pembulatan yang dipilih. Oleh karena itu, data frekuensi kata yang di transfomasi menjadi wavelet yang berjumlah 24 data berubah menjadi 12 koefisien. *Wavelet coefficient* yang dipilih adalah koefisien aproksimasi.



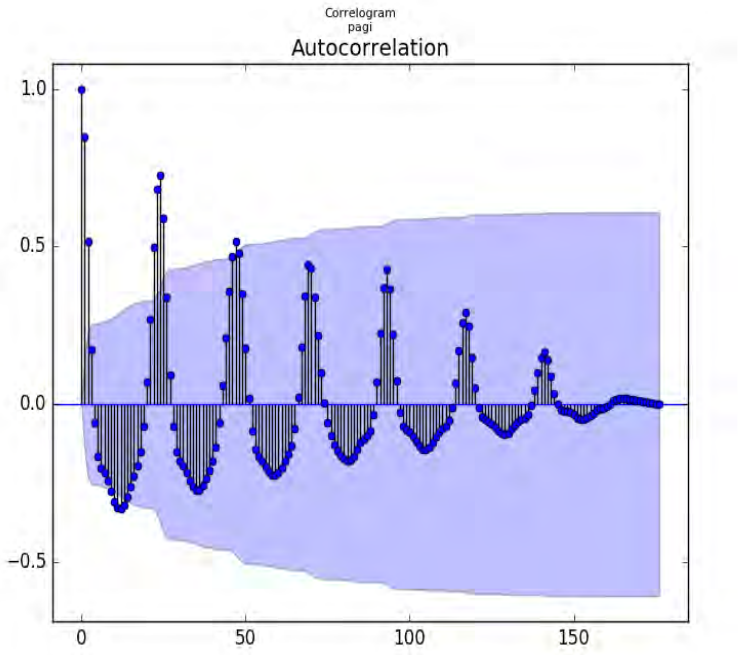
Gambar 5.8 *Wavelet coefficient* kata "pagi" tanggal 8 Mei 2016

Gambar 5.9 merupakan gambar hasil *wavelet coefficient* yang digambarkan pada grafik. Gambar diatas menunjukkan bahwa kata "pagi" mengalami penaikan dan penurunan pada waktu tertentu secara teratur. Kata "pagi" menunjukkan pola musiman yang merupakan fluktuasi dari data yang terjadi secara periodik. Oleh karena itu, kata 'pagi' dapat dianggap sebagai kata kunci biasa (*trivial*). Puncak yang terdeteksi pada kata kunci "pagi" terjadi di interval waktu yang hampir sama dengan kemunculan yang tinggi.



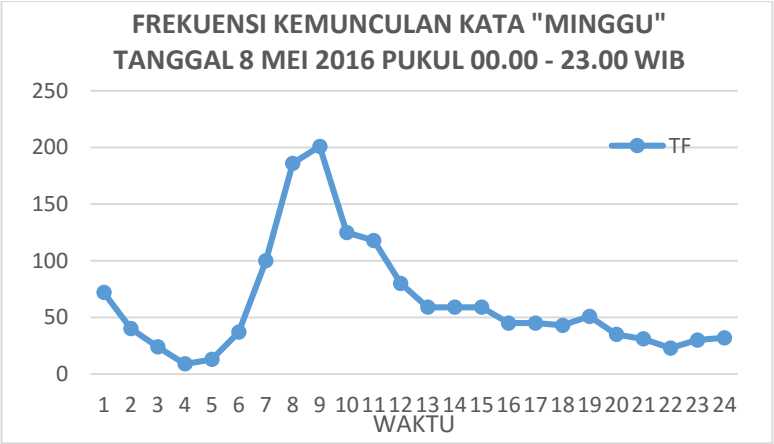
Gambar 5.9 Kata kunci "pagi" yang memiliki pola musiman yang teratur pada wavelet dan frekuensi kemunculan tinggi pada waktu yang hampir sama

Hal ini juga diperkuat dari hasil *autocorrelation* yang ditunjukkan pada *correlogram* Gambar 5.10. Terjadi intervensi pada beberapa *lag* yaitu *lag* yang melewati batas standar dari nilai *autcorrelation.* Kedekatan tiap *lag* menunjukkan bahwa adanya korelasi dari *lag* tersebut pada satu waktu. Penting atau tidaknya suatu kata tergantung dari jumlah *lag* yang melebihi batas nilai *autocorrelation* yang ditentukan berdasarkan *confidence boundary*. Jumlah nilai *autocorrelation* dari kata "pagi" yang melebihi batas *confindece boundary* 0.20 adalah sebanyak 52 sehingga semakin memperkuat kata "pagi" dianggap sebagai kata biasa (*trivial).*



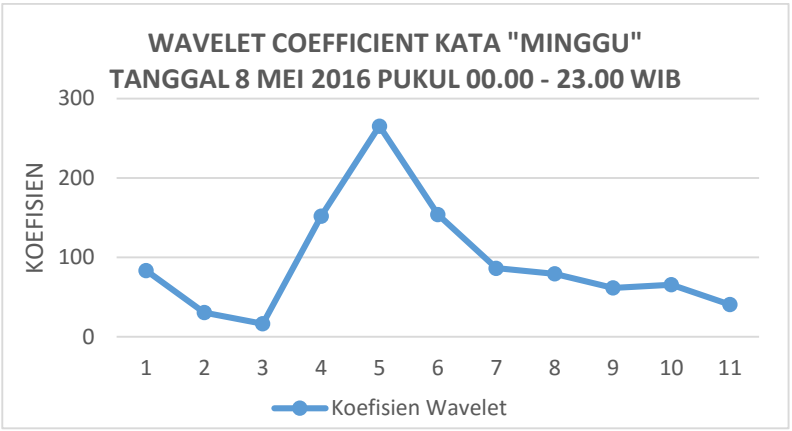
Gambar 5.10 Correlogram kata "pagi" menunjukkan hasil *autocorrelation* yang tinggi

Berbeda dengan kata 'pagi' yang merupakan salah satu kata yang dianggap sebagai kata biasa (*trivial*), kata 'minggu' merupakan kata yang dapat dianggap sebagai kata penting atau *non-trivial*. Kata 'minggu' muncul dengan rata – rata yang hampir sama pada interval waktu, tetapi mengalami kenaikan pada interval waktu ke-6 hingga ke-9 yang ditunjukkan pada Gambar 5.11. Tanggal 8 Mei 2016 kebetulan merupakan hari Minggu dan kenaikan kemunculan kata 'minggu' dimulai pada pukul 05.00 pagi hingga pukul 09.00 dan menurun secara perlahan pada pukul 10.00.

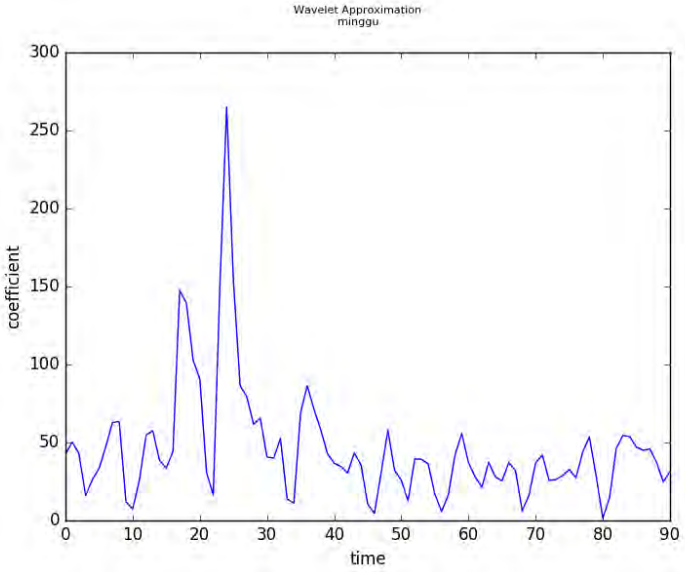


Gambar 5.11 Grafik kemunculan kata "minggu"

Gambar 5.12 merupakan hasil representasi *wavelet coefficient* pada tanggal 8 Mei 2016. Hasil transformasi *wavelet* berupa *wavelet coefficient* yang didapat dari data frekuensi kata. Koefisien yang digunakan adalah koefisien aproksimasi.

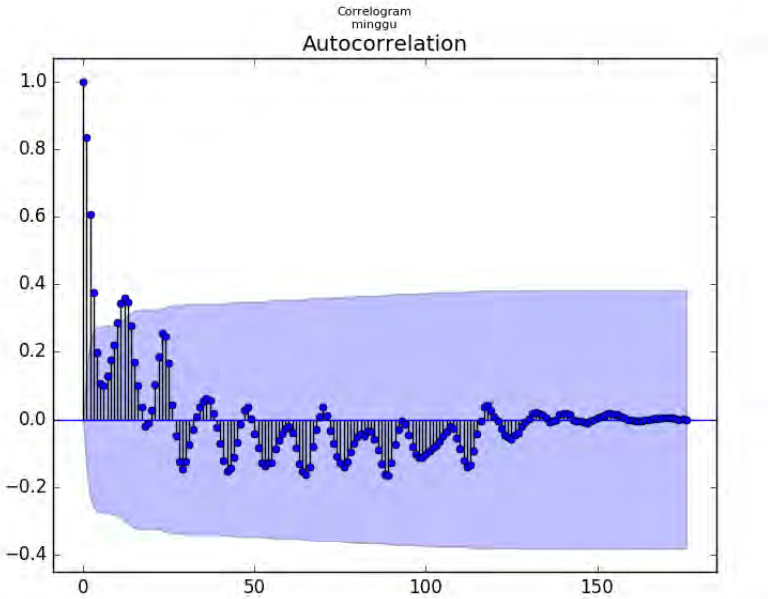


Gambar 5.12 *Wavelet coefficient* kata "minggu" tanggal 8 Mei 2016



Gambar 5.13 Kata kunci "minggu" yang mengalami kenaikan puncak hanya pada satu waktu

Gambar 5.13 merupakan hasil *wavelet coefficient* secara keseluruhan dari data yang didapat. Hasil Gambar 5.13 menunjukkan secara umum bahwa kata "minggu" selalu dibahas tetapi dengan intensitas yang rendah dan mengalami puncaknya pada rentang 20 – 30. Perhitungan *autocorrelation* pada kata 'minggu' digambarkan pada *correlogram* seperti Gambar 5.14. *Lag* yang berada diluar batas *confidence boudary* disimpulkan bahwa koefisien *autocorrelation* tersebut signifikan yang berarti ada hubungan signifikan antara *wavelet coefficient* dengan *wavelet coefficient* itu sendri dengan *time lag* 1 periode. Sedangkan pada kata "minggu" terlihat bahwa *lag* berada dibawah rentang batas *confidence boundary* yaitu 0.20 .Jumlah *lag* yang memiliki nilai dibawah batas nilai *autocorrelation* adalah 12. Hasil tersebut menunjukkan kata "minggu" merupakan kata *non trivial* (penting).



Gambar 5.14 Correlogram kata kunci "minggu" yang memiliki nilai autokorelasi rendah

Confidence boundary adalah nilai batas atas dan bawah koefisien pada *correlogram* yang menentukan sebuah *wavelet* dianggap memiliki tingkat keperiodikan tinggi atau rendah. Pengujian nilai *confidence boundary* dilakukan sebanyak 4 (empat) kali pengujian, yaitu 0.05, 0.10, 0.15 dan 0.20. Berdasarkan hasil pengamatan secara manual, batas nilai *confindence boundary* yang terbaik adalah penggunaan *confidence boundary 0.20.* Kriteria penentuan batas nilai *confindence boundary* secara manual dipertimbangkan berdasarkan kedekatan hubungan sebuah kata kunci dengan keterangan waktu atau kata kunci terkait dengan hal – hal yang dilakukan secara jelas pada waktu – waktu tertentu.

Tabel 5.6 Tabel Jumlah Nilai *Autocorrelation* berdasarkan *Confidence Boundary* 0.15 & 0.20 teratas

Kata Kunci	*Conf. Bound > 0.15*	Kata Kunci	*Conf. Bound > 0.20*
mall	95	mall	77
plaza	93	ayam	72
ayam	91	plaza	70
long	91	international	69
coffee	88	coffee	63
centre	87	bakso	63
international	86	pusat	62
cafe	85	bangun	61
pantai	85	airport	61
family	84	food	60

Nilai *confidence boundary* yang optimal menurut pengamatan adalah 0.20. Batas minimal dari hasil nilai *autocorrelation* menurut pengamatan adalah 17. Nilai *autocorrelation* yang berada pada lebih dari sama dengan 17 dianggap sebagai kata kunci *trvial* atau biasa dan dianggap sebagai kata yang berulang secara periodik. Penentuan batas ini bertujuan untuk menghasilkan kata kunci penting (*non-trivial)* dan mengeliminasi kata biasa yang dianggap berulang secara periodik. Perlu diketahui yang dimaksud dengan kata *non trivial* (penting) adalah kata yang memiliki nilai *autocorrelation* yang tinggi dan berulang secara periodik.

Tabel 5.7 Tabel kata yang dianggap penting dengan batas nilai kurang dari 17

Kata Kunci	*Conf. Bound > 0.20*	Kata Kunci	*Conf. Bound > 0.20*
hotel	16	laaa	16

Kata Kunci	*Conf. Bound > 0.20*	Kata Kunci	*Conf. Bound > 0.20*
cinta	16	untung	16
bukit	16	eting	16
raya	16	kite	16
abang	16	star	16
foto	16	beda	16
tolong	16	niat	16
keluarga	16	pandai	16
class	16	lemak	16
game	16	stadion	16
great	16	angkat	16
marang	16	latih	16
tanjung	16	serdang	16
mudah	16	kitchen	16
lambat	16	water	16
gonna	16	kuliner	16
kereta	16	bentar	16
minum	16	junior	16
instagram	16	king	16
kayak	16	urus	16

Hasil pengujian diatas menunjukkan penggunaan lokasi dapat berpengaruh pada hasil kata yang akan dihasilkan. Pada saat *crawling* terdapat beberapa *tweet* yang bukan merupakan *tweet* berbahasa Indonesia. Hal ini dikarenakan oleh terdapat beberapa negara yang berada di rentang geografis Indonesia. Lokasi geografis Indonesia terbentang luas dari barat hingga timur mengakibatkan kata – kata yang muncul beraneka ragam. Oleh karena itu, kata penting yang dihasilkan tidak terlalu spesifik yang

sesuai dengan isu atau kejadian yang sedang terjadi. Kata – kata yang dihasilkan pada Tabel 5.7 digunakan untuk skenario uji coba 3.

5.2.3 Skenario Uji Coba 3 : Pengujian kombinasi parameter untuk proses ekstraksi Trending Issue berdasarkan pembuangan kata kunci trivial

Kata – kata penting (*non – trivial)* yang sudah didapat dari uji coba 2 yaitu kata *non trivial* dengan nilai *autocorrelation* kurang dari 17 digunakan untuk proses eliminasi kata dalam proses ekstraksi *trending issue.* Kata *non-trivial* yang didapat disimpan pada sebuah *file* dalam bentuk kamus kata. Setiap kata kunci akan dibandingkan dengan kata – kata yang terdapat pada *tweets* berita yang sudah dikumpulkan. Kata pada *tweets* yang tidak mengandung kata kunci akan dibuang.

Tabel 5.8 Kumpulan beberapa kata kunci penting yang nilai *autocorrelation* kurang dari 17

No	Kata	No	Kata	No	Kata	No	Kata
1	hotel	11	great	21	laaa	31	angkat
2	cinta	12	marang	22	untung	32	latih
3	bukit	13	tanjung	23	eting	33	serdang
4	raya	14	mudah	24	kite	34	kitchen
5	abang	15	lambat	25	star	35	water
6	foto	16	gonna	26	beda	36	kuliner
7	tolong	17	kereta	27	niat	37	bentar
8	keluarga	18	minum	28	pandai	38	junior
9	class	19	instagram	29	lemak	39	king
10	game	20	kayak	30	stadion	40	urus

Kata – kata yang terdapat pada Tabel 5.8 Kumpulan beberapa kata kunci penting yang nilai *autocorrelation* kurang dari 17menunjukkan kata – kata *non-trivial* yang didapat dari hasil *wavelet autocorrelation coefficients* berdasarkan *crawling* lokasi

geografis Indonesia. Kata kunci inilah yang digunakan untuk ekstraksi *trending issue.* Secara umum dan penglihatan secara manual, kata – kata penting yang dihasilkan merupakan kata umum yang tidak terpaku pada satu topik yang berkaitan. Hal ini dikarenakan proses *crawling* yang dilakukan tidak menggunakan kata kunci atau *keyword* sehingga semua kata yang dihasilkan berupa kata – kata umum yang banyak muncul pada *tweet*

Kata *non trivial* yang digunakan sebanyak 5895 kata. Sedangkan *tweets* berita yang akan di proses untuk mendapatkan *trending issue* berjumlah 73 *tweets* berita yang diambil pada tanggal 12 Mei 2016. Pada *crawling tweets* berita, digunakan kata kunci dalam pengambilan berita tersebut. Hal ini bertujuan untuk mendapatkan *tweets* berita yang sesuai dan detik.com sebagai akun Twitter pemiliki *tweets* berita.

Pada uji coba ini, digunakan kumpulan *tweets* berita dengan kata kunci 'puncak' dan 'libur' yang sudah disimpan didalam sebuah *file*. Kata kunci dipilih dikarenakan pada saat itu terjadi libur panjang untuk masyarakat Indonesia atau dikenal dengan *long weekend.* Pembuangan kata yang dianggap tidak penting dilakukan berdasarkan kata kunci *non trivial* yang sudah disimpan dari proses sebelumnya. Sehingga yang tersisa pada tiap *tweets* hanyalah kata – kata penting saja, terlihat pada Tabel 5.9. dimana "u" berarti *unique* yang digunakan sebagai penanda yang berguna untuk bobot kata tersebut. Selanjutnya dilakukan proses ekstraksi *trending issue* dari kumpulan *tweets* yang berisi kata kunci *non trivial* dengan melakukan uji coba klasterisasi *tweets*.

Tabel 5.9 Contoh pembuangan kata *trivial* pada *tweets*

Kata pada *tweets* sebelum pembuangan kata *trivial*
[[u'gilas', u'espanyol', u'barca', u'pekan', u'puncak', u'klasemen', u'via'], [u'puncak', u'arus', u'bakauheni', u'siapkan', u'loket', u'penumpang', u'via'], [u'polisi', u'tol', u'cikampek', u'lancar', u'puncak', u'arus', u'jakarta', u'diprediksi', u'jelang', u'subuh'], [u'libur', u'pengunjung', u'antusias', u'lihat',u'puncak', u'monas', u'via'], [u'macet', u'kendaraan', u'arah', u'puncak', u'gerak',

Kata pada *tweets* sebelum pembuangan kata *trivial*
u'lepas', u'gerbang', u'tol', u'ciawi'], [u'puncak', u'arus', u'mudik', u'kendaraan', u'libur', u'long', u'weekend', u'mei'], [u'long', u'weekend', u'tol', u'cikampek', u'lancar', u'jagorawi', u'arah', u'puncak', u'padat'], [u'jam', u'parkir', u'jalan', u'puncak', u'rio', u'sekeluarga',u'lesehan', u'rerumputan'], [u'macet', u'arah', u'cikampek', u'lalin', u'puncak', u'penuh', u'dialihkan'],[u'jadwal', u'pemberlakuan', u'arus', u'arah', u'puncak', u'long', u'weekend'], [u'antrean', u'kendaraan', u'gerbang', u'tol', u'ciawi', u'puncak', u'sukabumi', u'capai'], [u'lalin', u'puncak', u'diberlakukan', u'arah', u'jakarta', u'ditutup'], [u'libur', u'arah', u'puncak', u'macet', u'pasar', u'cisarua'], [u'ide', u'long', u'weekend', u'lihat', u'jakarta', u'puncak', u'tugu', u'monas',u'malam', u'via'], [u'puncak', u'macet', u'libur', u'long', u'weekend', u'wib']]
Kata pada *tweets* setelah pembuangan kata kunci *trivial*
[[u'barca', u'pekan'], [u'arus'], [u'polisi', u'cikampek', u'arus', u'jelang'], [u'monas'], [u'macet', u'arah', u'gerak', u'gerbang', u'ciawi'], [u'arus', u'mudik'], [u'cikampek', u'arah', u'padat'], [u'parkir'], [u'macet', u'arah', u'cikampek', u'lalin'], [u'jadwal', u'arus', u'arah'], [u'gerbang', u'ciawi',u'sukabumi', u'capai'], [u'lalin', u'arah'], [u'arah', u'macet', u'pasar', u'cisarua'], [u'tugu', u'monas'], [u'macet']]

Pengujian dilakukan dengan cara pemberian beberapa variasi jumlah centroid (*k)* terhadap *clustering tweets* untuk mendapatkan satu hasil *clustering* yang terbaik. Proses *clustering* dilakukan berdasarkan 2(dua) cara, yaitu proses klaster dengan *tweets* yang hanya terdiri dari kata penting saja dan proses klaster dengan *tweets* tanpa eliminasi berdasarkan kata penting. Pengukuran kualitas dari hasil *clustering* menggunakan metode *Silhouette coefficients.* Penetuan jumlah *centroid* (*k)* sangat mempengaruhi hasil dari proses klaster karena menggunakan algoritma K-medoids.

Nilai *k* awal yang ditentukan sebagai *centroid* adalah *k=3* sampai *k*= $\sqrt{n/2}$ dimana *n* merupakan jumlah *tweets.* Perhitungan

jarak antar vektor menggunakan jarak *cosine.* Pada percobaan dilakukan beberapa uji parameter yaitu nilai *threshold* TF, *threshold* CI, *threshold* TFIDF dan *threshold* WF sudah ditentukan, dimana nilai TF memiliki rentang 0 -2, nilai CI memiliki rentang 1-6, TFIDF 0.1-0.6 dan nilai WF memiliki rentang 1-3.

Pengujian ini membandingkan 2 data *tweet* dalam proses klasterisasi yaitu data *tweet* yang sudah melalui pembuangan kata *non trivial* dan data *tweet* asli tanpa pembuangan kata *trivial.* Pengujian dilakukan dnegan 10 (sepuluh) kali percobaan untuk masing – masing data. Hasil dari percobaan ini adalah data yang memiliki kombinasi parameter terbaik dengan nilai *silhouette* tertinggi. Berikut adalah pengujian pada data pertama yaitu *tweet* yang sudah melalui pembuangan kata *trivial* yang menghasilkan nilai *silhouette* terbaik dari 5 paramater :

Tabel 5.10 Hasil percobaan ekstraksi *tweets* dengan pembuangan kata *trivial* untuk mendapatkan *trending issue*

Percobaan ke	Jumlah Klaster	TF Thres	CI Thres	TFIDF Thres	WF Thres	Nilai Sillhouette
1	4	1	5	0.5	1	0.36322
Kata : arah						
2	4	0	4	0.4	1	0.35063
Kata : cikampek, macet, lalin						
3	4	0	4	0.2	1	0.36322
Kata : cikampek, macet, lalin						
4	4	0	1	0.1	1	0.36159
Kata : arah,cikampek, macet, lalin						
5	4	1	6	0.4	1	0.35063
Kata : arah						
6	4	0	2	0.2	1	0.36159
Kata : arah,cikampek, macet, lalin						
7	4	0	6	0.6	2	0.36322
Kata : arus						
8	4	0	1	0.4	2	0.34674
Kata : macet						
9	4	0	2	0.2	3	0.34674

Percobaan ke	Jumlah Klaster	TF Thres	CI Thres	TFIDF Thres	WF Thres	Nilai Sillhouette
Kata : macet						
10	4	0	4	0.1	2	0.36322
Kata : macet, arah						

Berdasarkan hasil tabel diatas, jumlah klaster yang paling baik untuk menghasilkan nilai *silhouette* yang terbaik adalah 4 (empat). Terdapat 4 (empat) nilai *Silhouette* tertinggi dari 10 (sepuluh) percobaan diatas yaitu 0.36322, 0.35063, 0.36159 dan 0.34674. Keempat-nya menghasilkan kombinasi parameter yang berbeda dengan menghasilkan *trending issue* yang berbeda. Hal ini dipengaruhi oleh algoritma klaster yang digunakan yaitu algoritma K-medoids karena terjadi proses *update centroid* dari *medoids* yang dihasilkan berdasarkan penghitungan jarak kemiripan antara *tweets* terhadap *medoids.* Setiap kombinasi parameter mempengaruhi *trending issue* yang dihasilkan. Rata – rata dari nilai *silhouette* menunjukkan bahwa hasil klaster sudah cukup baik yaitu 0.35708 dimana angka tersebut berada pada rentang $0{,}26 \leq \text{ASW} < 0{,}51$.

Berikut adalah percobaan kedua yaitu *tweet* tanpa melalui pembuangan kata *trivial* yang menghasilkan nilai *silhouette* terbaik dari 5 paramater :

Tabel 5.11 Hasil percobaan ekstraksi *tweets* tanpa pembuangan kata *trivial* untuk mendapatkan *trending issue*

Percobaan ke	Jumlah Klaster	TF Thres	CI Thres	TFIDF Thres	WF Thres	Nilai Sillhouette
1	4	1	3	0.5	1	0.32623
Kata : -						
2	4	0	1	0.1	2	0.32224
Kata : libur, macet, long, arah, weekend						
3	4	0	6	0.1	3	0.32224
Kata : arah						
4	4	0	1	0.1	2	0.32224
Kata : libur, macet, long, arah, weekend						

Percobaan ke	Jumlah Klaster	TF Thres	CI Thres	TFIDF Thres	WF Thres	Nilai Sillhouette
5	4	0	5	0.2	3	0.32224
Kata : arah						
6	4	0	4	0.1	2	0.32618
Kata : cikampek, arah						
7	4	0	5	0.3	1	0.32394
Kata : cikampek, macet, lalin, arus, libur						
8	4	1	3	0.1	2	0.32224
Kata : arah						
9	4	1	6	0.2	1	0.32011
Kata : arah						
10	4	0	6	0.1	3	0.32623
Kata : arah						

Berdasarkan hasil tabel diatas, jumlah klaster yang paling baik adalah 4 (empat). Terdapat nilai *Silhouette* yang paling tinggi dari 10 (sepuluh) percobaan diatas yaitu 0.32622 dengan kombinasi parameter yang menghasilkan *trending issue* sebagai berikut yaitu batas nilai *threshold* TF adalah 0 dan untuk TFIDF adalah 0.1. Sedangkan untuk nilai CI, batas nilai *threshold* yang terbaik adalah 6 dan batas nilai *threshold* untuk WF adalah 3. Rata – rata dari nilai *silhouette* menunjukkan bahwa hasil klaster sudah cukup baik yaitu 0.32339 dimana angka tersebut berada pada rentang $0{,}26 \leq \text{ASW} < 0{,}51$.

Berdasarkan 2(dua) pengujian di atas terlihat bahwa jumlah klaster yang menghasilkan nilai *silhouette* yang baik adalah *n=4* untuk kedua pengujian. Nilai *silhouette* lebih baik pada pengujian pertama, yaitu pada *tweets* yang hanya berisi kata penting (*non trivial*) saja dengan nilai *silhouette* tertinggi 0.36322 sedangkan pada pengujian kedua, nilai *silhouette* yang dihasilkan adalah 0.32622. Hal ini membuktikan bahwa *tweets* dengan pembuangan kata kunci biasa (*trivial*) lebih baik daripada *tweets* tanpa pembuangan kata kunci biasa (*trivial).*

5.2.4 Skenario Uji Coba 4 : Pengujian hasil ringkasan berdasarkan ekstraksi trending issue

Implementasi dari Tugas Akhir ini adalah hasil ringkasan yang nantinya berguna untuk mengetahui berita tanpa harus membaca keseluruhan berita tersebut. Ringkasan berita yang dihasilkan berdasarkan *trending issue* yang didapat dari uji coba 3 dan fitur dari berita tersebut. *Trending issue* didapat dari proses ekstraksi *tweets* sehingga menghasilkan kata kunci atau isu. Sedangkan fitur berita yang diambil adalah TF, DF, WF, posisi kalimat, dan judul berita yang sudah dijelaskan pada subbab 3.1.5.

Teknik pembobotan kalimat digunakan untuk menghasilkan ringkasan yang sesuai dengan *trending issue.* Setiap isu yang dihasilkan pada proses ekstraksi *trending issue* disimpan kedalam *file* yang nantinya akan digunakan untuk mengukur kemiripan kalimat terhadap *trending issue.* Kualitas hasil peringkasan ini akan diukur berdasarkan metode evaluasi nilai ROUGE -1. Berita yang akan diringkas adalah berita yang sudah didapat dari *tweets* berita yang sudah dikumpulkan sebelumnya dengan kata kunci 'libur' dan 'puncak' dengan jumlah 72 berita. Groundtruth yang digunakan merupakan hasil peringkasan berita secara manual yang dibuat oleh seorang mahasiswa ITS Teknik Informatika.

Ringkasan yang didapat diurutkan berdasarkan *score* dari hasil penghitungan bobot kalimat menggunakan 6 teknik pembobotan kalimat. Pengujian dilakukan berdasarkan nilai *silhouette* tertinggi hasil uji coba 3 dan nilai Rouge tertinggi.

Ringkasan yang dihasilkan Pengujian dilakukan dengan 10 (sepuluh) kali percobaan yang menunjukkan nilai *silhouette* tertinggi yang dihasilkan oleh proses ekstraksi *trending issue* yang ditampilkan pada Tabel 5.12. Rata – rata nilai rouge yang dihasilkan berdasarkan *trending issue* dengan pembuangan kata *trivial* adalah 0.1854. Sedangkan rata – rata nilai rouge yang dihasilkan untuk *trending issue* tanpa pembuangan kata kunci *trivial* adalah 0.0656. Nilai *silhouette* yang diambil hanya berdasarkan 10 nilai teratas. Oleh karena itu, pada percobaan

pertama ekstraksi *trending issue* tanpa pembuangan kata *trivial* tidak terdapat isu yang dihasilkan oleh 10 nilai *silhouette* teratas.

Tabel 5.12 Perbandingan hasil ringkasan dari nilai rouge berdasarkan nilai *silhouette* tertinggi

Dengan pembuangan kata kunci *trivial*			**Tanpa pembuangan kata kunci *trivial***		
Nilai Rouge	**Nilai Silhouette**	**Isu**	**Nilai Rouge**	**Nilai Silhouette**	**Isu**
0.1739	0.36322	arah	-1	0.32623	-
"Satu arah sudah kami berlakukan sejak pukul 08.00 WIB. Kami prioritaskan dulu yang ke atas ke Puncak dan Sukabumi. Sore baru kami berlakukan satu arah ke Jakarta," ujar Kasat Lantas Polres Bogor Kabupatem AKP Bramanstya kepada detikcom, Kamis Score : 4.371063					
0.1660	0.35063	cikampek, macet, lalin	0.1779	0.32224	libur, macet, long, arah, weekend
"Tol dari Jakarta mengarah ke Cikampek pagi ini terpantau lancar. Ada kepadatan sekitar KM 37 karena sempat terjadi kecelakaan, namun saat ini kendaraan telah dievakuasi," ujar petugas Jasa Marga Ema			"Jakarta - Arus balik ke arah Ibu Kota Jakarta usai liburan long weekend diprediksi mencapai puncaknya menjelang subuh, Senin besok" Score : 4.660223		

Dengan pembuangan kata kunci *trivial*			Tanpa pembuangan kata kunci *trivial*		
Nilai Rouge	**Nilai Silhouette**	**Isu**	**Nilai Rouge**	**Nilai Silhouette**	**Isu**
saat dihubungi detikcom, Jumat Score : 4.2739					
0.1660	0.36322	cikampek, macet, lalin	0.1739	0.32224	arah
"Tol dari Jakarta mengarah ke Cikampek pagi ini terpantau lancar. Ada kepadatan sekitar KM 37 karena sempat terjadi kecelakaan, namun saat ini kendaraan telah dievakuasi," ujar petugas Jasa Marga Ema saat dihubungi detikcom, Jumat Score : 4.27394			"Satu arah sudah kami berlakukan sejak pukul 08.00 WIB. Kami prioritaskan dulu yang ke atas ke Puncak dan Sukabumi. Sore baru kami berlakukan satu arah ke Jakarta," ujar Kasat Lantas Polres Bogor Kabupatem AKP Bramanstya kepada detikcom, Kamis Score : 4.371063		
0.1976	0.36159	arah, cikampek, lalin, macet	0.1779	0.32224	libur, macet, long, arah, weekend
"Jakarta - Arus balik ke arah Ibu Kota Jakarta usai liburan long weekend diprediksi mencapai puncaknya menjelang subuh, Senin besok" Score : 4.537747			"Jakarta - Arus balik ke arah Ibu Kota Jakarta usai liburan long weekend diprediksi mencapai puncaknya menjelang subuh, Senin besok" Score : 4.660223		

Dengan pembuangan kata kunci *trivial*			Tanpa pembuangan kata kunci *trivial*		
Nilai Rouge	**Nilai Silhouette**	**Isu**	**Nilai Rouge**	**Nilai Silhouette**	**Isu**
0.1739	0.35063	arah	0.1739	0.32224	arah
"Satu arah sudah kami berlakukan sejak pukul 08.00 WIB. Kami prioritaskan dulu yang ke atas ke Puncak dan Sukabumi. Sore baru kami berlakukan satu arah ke Jakarta," ujar Kasat Lantas Polres Bogor Kabupatem AKP Bramanstya kepada detikcom, Kamis Score : 4.371063			"Satu arah sudah kami berlakukan sejak pukul 08.00 WIB. Kami prioritaskan dulu yang ke atas ke Puncak dan Sukabumi. Sore baru kami berlakukan satu arah ke Jakarta," ujar Kasat Lantas Polres Bogor Kabupatem AKP Bramanstya kepada detikcom, Kamis Score : 4.371063		
0.1976	0.36159	arah, cikampek, lalin, macet	0.2016	0.32618	cikampek, arah
"Jakarta - Arus balik ke arah Ibu Kota Jakarta usai liburan long weekend diprediksi mencapai puncaknya menjelang subuh, Senin besok" Score : 4.537747			"Jakarta - Arus balik ke arah Ibu Kota Jakarta usai liburan long weekend diprediksi mencapai puncaknya menjelang subuh, Senin besok" Score : 4.567885		
0.2411	0.36322	arus	0.2016	0.32394	cikampek ,macet, lalin, arus, libur
"Ada beberapa titik, baik itu di jalur Pantura, tengah dan selatan. Di lokasi tersebut ada			"Tol dari Jakarta mengarah ke Cikampek pagi ini terpantau lancar. Ada kepadatan sekitar		

Dengan pembuangan kata kunci *trivial*			Tanpa pembuangan kata kunci *trivial*		
Nilai Rouge	**Nilai Silhouette**	**Isu**	**Nilai Rouge**	**Nilai Silhouette**	**Isu**
pasar tumpah dan aktivitas masyarakat," kata Dirlantas Polda Jabar Kombes Pol Sugihardi di Mapolda Jabar, Jalan Soekarno Hatta, Kota Bandung, Rabu Score : 4.746942			KM 37 karena sempat terjadi kecelakaan, namun saat ini kendaraan telah dievakuasi," ujar petugas Jasa Marga Ema saat dihubungi detikcom, Jumat Score : 4.25603		
0.1660	0.3467	cikampek, macet, lalin	0.1739	0.32224	arah
"Tol dari Jakarta mengarah ke Cikampek pagi ini terpantau lancar. Ada kepadatan sekitar KM 37 karena sempat terjadi kecelakaan, namun saat ini kendaraan telah dievakuasi," ujar petugas Jasa Marga Ema saat dihubungi detikcom, Jumat Score : 4.273945			"Satu arah sudah kami berlakukan sejak pukul 08.00 WIB. Kami prioritaskan dulu yang ke atas ke Puncak dan Sukabumi. Sore baru kami berlakukan satu arah ke Jakarta," ujar Kasat Lantas Polres Bogor Kabupatem AKP Bramanstya kepada detikcom, Kamis Score : 4.371063		
0.1858	0.3467	macet ,arah	0.2016	0.32011	arah
Disampaikan Kasubdit Pembinaan dan Penegakan Hukum Ditlantas Polda Metro Jaya AKBP Budiyanto hingga Minggu pukul 1000 WIB ini, antrean sudah mulai tampak di Tol Jagorawi arah Jakarta, imbas antrean gerbang tol di			"Satu arah sudah kami berlakukan sejak pukul 08.00 WIB. Kami prioritaskan dulu yang ke atas ke Puncak dan Sukabumi. Sore baru kami berlakukan satu arah ke Jakarta," ujar Kasat Lantas Polres Bogor Kabupatem		

Dengan pembuangan kata kunci *trivial*			Tanpa pembuangan kata kunci *trivial*		
Nilai Rouge	**Nilai Silhouette**	**Isu**	**Nilai Rouge**	**Nilai Silhouette**	**Isu**
ruas Cimanggis dan karena penyempitan proyek di KM 9" Score : 3.71932			AKP Bramanstya kepada detikcom, Kamis Score : 4.371063		
0.1858	0.36322	macet ,arah	0.1739	0.32623	arah
"Disampaikan Kasubdit Pembinaan dan Penegakan Hukum Ditlantas Polda Metro Jaya AKBP Budiyanto hingga Minggu pukul 1000 WIB ini, antrean sudah mulai tampak di Tol Jagorawi arah Jakarta, imbas antrean gerbang tol di ruas Cimanggis dan karena penyempitan proyek di KM 9" Score : 3.71932			"Satu arah sudah kami berlakukan sejak pukul 08.00 WIB. Kami prioritaskan dulu yang ke atas ke Puncak dan Sukabumi. Sore baru kami berlakukan satu arah ke Jakarta," ujar Kasat Lantas Polres Bogor Kabupatem AKP Bramanstya kepada detikcom, Kamis Score : 4.371063		

Hasil analisa dari Tabel 5.12 nilai rouge yang dihasilkan berdasarkan nilai *silhouette* tertinggi jika dilakukan perbandingan tiap – tiap percobaan menunjukkan bahwa nilai rouge yang dihasilkan berdasarkan nilai *silhouette* tertinggi hasil ekstraksi *trending issue* tanpa pembuangan kata *trivial* lebih tinggi daripada berdasarkan nilai *silhouette* tertinggi hasil ekstraksi *trending issue* dengan pembuangan kata *trivial*. Tetapi rata – rata yang dihasilkan menunjukkan nilai rouge yang dihasilkan berdasarkan nilai *silhouette* tertinggi hasil ekstraksi *trending issue* dengan pembuangan kata *trivial* lebih baik yaitu 0.1854 daripada nilai rouge yang dihasilkan berdasarkan nilai *silhouette* tertinggi hasil ekstraksi *trending issue* tanpa pembuangan kata *trivial*.

Pengujian juga dilakukan dengan melihat nilai rouge tertinggi yang dihasilkan dari hasil 10 (sepuluh) kali percobaan.

Nilai rouge tetap dibandingkan berdasarkan ekstraksi *trending issue* dengan pembuangan kata *trivial* dan tanpa pembuangan kata *trivial.* Rata - rata nilai rouge dengan pembuangan kata *trivial* adalah 0.28816 sedangkan rata- rata nilai rouge tanpa pembuangan kata *trivial* adalah 0.30199.

Tabel 5.13 Perbandingan hasil ringkasan berdasarkan nilai Rouge tertinggi

Dengan pembuangan kata kunci *trivial*			**Tanpa pembuangan kata kunci *trivial***		
Nilai Rouge	**Nilai Silhouette**	**Isu**	**Nilai Rouge**	**Nilai Silhouette**	**Isu**
0.3439	0.1456	tourin g, jabar, blitar, arah, tips, seru, arus, macet	0.3439	0.0965	tol, batu, anak
Ada beberapa titik, baik itu di jalur Pantura, tengah dan selatan. Di lokasi tersebut ada pasar tumpah dan aktivitas masyarakat," kata Dirlantas Polda Jabar Kombes Pol Sugihardi di Mapolda Jabar, Jalan Soekarno Hatta, Kota Bandung, Rabu Score : 4.83179			Ini kan habis liburan. Makan sana sini atau ada perilaku buruk yang muncul. Makanya dilaksanakan pemeriksaan urine. Saya yakin, semunya enggak ada," ujar Aher didampingi Deddy Mizwar sambil memperlihatkan tabung berisi air urine serta alat tes yang menampilkan dua garis merah atau negatif narkoba Score : 4.75378		
0.2609	-0.02296	tourin g,	0.2609	0.09853	bandun g,

Dengan pembuangan kata kunci *trivial*			Tanpa pembuangan kata kunci *trivial*		
Nilai Rouge	**Nilai Silhouette**	**Isu**	**Nilai Rouge**	**Nilai Silhouette**	**Isu**
		jabar, blitar, normal, arah, tips, seru, jelang, macet			surabaya, libur
Ada beberapa titik, baik itu di jalur Pantura, tengah dan selatan. Di lokasi tersebut ada pasar tumpah dan aktivitas masyarakat," kata Dirlantas Polda Jabar Kombes Pol Sugihardi di Mapolda Jabar, Jalan Soekarno Hatta, Kota Bandung, Rabu Score : 4.82694			Ini kan habis liburan. Makan sana sini atau ada perilaku buruk yang muncul. Makanya dilaksanakan pemeriksaan urine. Saya yakin, semunya enggak ada," ujar Aher didampingi Deddy Mizwar sambil memperlihatkan tabung berisi air urine serta alat tes yang menampilkan dua garis merah atau negatif narkoba Score : 4.75378		
0.2609	0.17636	touring, jabar, blitar, normal, tips, seru, jelang	0.2688	0.2284	via, macet, arah, monas, arus, libur, lihat

Dengan pembuangan kata kunci *trivial*			Tanpa pembuangan kata kunci *trivial*		
Nilai Rouge	**Nilai Silhouette**	**Isu**	**Nilai Rouge**	**Nilai Silhouette**	**Isu**
		, macet			
Ada beberapa titik, baik itu di jalur Pantura, tengah dan selatan. Di lokasi tersebut ada pasar tumpah dan aktivitas masyarakat," kata Dirlantas Polda Jabar Kombes Pol Sugihardi di Mapolda Jabar, Jalan Soekarno Hatta, Kota Bandung, Rabu Score : 4.83179			Disampaikan Kasubdit Pembinaan dan Penegakan Hukum Ditlantas Polda Metro Jaya AKBP Budiyanto hingga Minggu pukul 1000 WIB ini, antrean sudah mulai tampak di Tol Jagorawi arah Jakarta, imbas antrean gerbang tol di ruas Cimanggis dan karena penyempitan proyek di KM 9 Score : 5.0166		
0.2688	0.1285	tourin g, jabar, pasar, blitar, seru arah, tips, abang	0.3439	0.0934	tol, anak
"Satu arah sudah kami berlakukan sejak pukul 08.00 WIB. Kami prioritaskan dulu yang ke atas ke Puncak dan Sukabumi. Sore baru kami berlakukan satu arah ke Jakarta," ujar Kasat Lantas Polres Bogor Kabupatem AKP Bramanstya kepada detikcom, Kamis			Ini kan habis liburan. Makan sana sini atau ada perilaku buruk yang muncul. Makanya dilaksanakan pemeriksaan urine. Saya yakin, semunya enggak ada," ujar Aher didampingi Deddy Mizwar sambil memperlihatkan tabung berisi air urine serta alat tes yang menampilkan		

Dengan pembuangan kata kunci *trivial*			Tanpa pembuangan kata kunci *trivial*		
Nilai Rouge	**Nilai Silhouette**	**Isu**	**Nilai Rouge**	**Nilai Silhouette**	**Isu**
Score : 4.371063			dua garis merah atau negatif narkoba Score : 4.75378		
0.2609	-0.0278	touring, jabar, pasar, blitar, seru, tips, jelang, macet, abang	0.2846	0.04532	batu, gunung, surabaya, jelang, wisatawan, abang, bandung, soetta, pasar, isi, tanah
Ada beberapa titik, baik itu di jalur Pantura, tengah dan selatan. Di lokasi tersebut ada pasar tumpah dan aktivitas masyarakat," kata Dirlantas Polda Jabar Kombes Pol Sugihardi di Mapolda Jabar, Jalan Soekarno Hatta, Kota Bandung, Rabu Score : 4.8608			Ada beberapa titik, baik itu di jalur Pantura, tengah dan selatan. Di lokasi tersebut ada pasar tumpah dan aktivitas masyarakat," kata Dirlantas Polda Jabar Kombes Pol Sugihardi di Mapolda Jabar, Jalan Soekarno Hatta, Kota Bandung, Rabu Score : 4.81930		
0.2688	0.1677	jabar, pasar, arah, tips, abang	0.3162	0.1023	touring, jawa, batu, amp, gunung, timur, surabaya

Dengan pembuangan kata kunci *trivial*			Tanpa pembuangan kata kunci *trivial*		
Nilai Rouge	**Nilai Silhouette**	**Isu**	**Nilai Rouge**	**Nilai Silhouette**	**Isu**
					, bandung, weekend, long, kelud, blitar, libur, seru
Ada beberapa titik, baik itu di jalur Pantura, tengah dan selatan. Di lokasi tersebut ada pasar tumpah dan aktivitas masyarakat," kata Dirlantas Polda Jabar Kombes Pol Sugihardi di Mapolda Jabar, Jalan Soekarno Hatta, Kota Bandung, Rabu Score : 4.907938			Ada beberapa titik, baik itu di jalur Pantura, tengah dan selatan. Di lokasi tersebut ada pasar tumpah dan aktivitas masyarakat," kata Dirlantas Polda Jabar Kombes Pol Sugihardi di Mapolda Jabar, Jalan Soekarno Hatta, Kota Bandung, Rabu Score : 4.779013		
0.3439	-0.16501	jabar, arah, tips, arus, macet	0.3439	0.1039	tol, anak
Ada beberapa titik, baik itu di jalur Pantura, tengah dan selatan. Di lokasi tersebut ada pasar tumpah dan aktivitas masyarakat," kata Dirlantas Polda Jabar Kombes Pol Sugihardi di Mapolda Jabar, Jalan Soekarno Hatta, Kota Bandung, Rabu Score : 4.8543			Ini kan habis liburan. Makan sana sini atau ada perilaku buruk yang muncul. Makanya dilaksanakan pemeriksaan urine. Saya yakin, semunya enggak ada," ujar Aher didampingi Deddy Mizwar sambil memperlihatkan tabung berisi air urine serta alat tes yang menampilkan		

Dengan pembuangan kata kunci *trivial*			Tanpa pembuangan kata kunci *trivial*		
Nilai Rouge	**Nilai Silhouette**	**Isu**	**Nilai Rouge**	**Nilai Silhouette**	**Isu**
			dua garis merah atau negatif narkoba Score : 4.75378		
0.3439	0.1650	jabar, arah, tips, arus, macet	0.2885	0.07823	touring, timur, jabar, weekend, long, terminal, arus, pengunjung
Ada beberapa titik, baik itu di jalur Pantura, tengah dan selatan. Di lokasi tersebut ada pasar tumpah dan aktivitas masyarakat," kata Dirlantas Polda Jabar Kombes Pol Sugihardi di Mapolda Jabar, Jalan Soekarno Hatta, Kota Bandung, Rabu Score : 4.8543			Ada beberapa titik, baik itu di jalur Pantura, tengah dan selatan. Di lokasi tersebut ada pasar tumpah dan aktivitas masyarakat," kata Dirlantas Polda Jabar Kombes Pol Sugihardi di Mapolda Jabar, Jalan Soekarno Hatta, Kota Bandung, Rabu Score : 4.83458		
0.2608	0.13658	touring, jabar, pasar, blitar, normal, seru, jelang, macet	0.2688	0.28081	via, macet, arah, monas, arus, libur, lihat

Dengan pembuangan kata kunci *trivial*			Tanpa pembuangan kata kunci *trivial*		
Nilai Rouge	**Nilai Silhouette**	**Isu**	**Nilai Rouge**	**Nilai Silhouette**	**Isu**
		, abang			
Ada beberapa titik, baik itu di jalur Pantura, tengah dan selatan. Di lokasi tersebut ada pasar tumpah dan aktivitas masyarakat," kata Dirlantas Polda Jabar Kombes Pol Sugihardi di Mapolda Jabar, Jalan Soekarno Hatta, Kota Bandung, Rabu Score : 4.8669			Disampaikan Kasubdit Pembinaan dan Penegakan Hukum Ditlantas Polda Metro Jaya AKBP Budiyanto hingga Minggu pukul 1000 WIB ini, antrean sudah mulai tampak di Tol Jagorawi arah Jakarta, imbas antrean gerbang tol di ruas Cimanggis dan karena penyempitan proyek di KM 9 Score : 5.016597		
0.2688	0.000247	tourin g, jabar, pasar, blitar, seru, arah, tips, abang	0.3004	0.0599	tol, wisata wan, batu, anak
Ada beberapa titik, baik itu di jalur Pantura, tengah dan selatan. Di lokasi tersebut ada pasar tumpah dan aktivitas masyarakat," kata Dirlantas Polda Jabar Kombes Pol Sugihardi di Mapolda Jabar, Jalan Soekarno Hatta, Kota Bandung, Rabu Score : 4.8742			Ada beberapa titik, baik itu di jalur Pantura, tengah dan selatan. Di lokasi tersebut ada pasar tumpah dan aktivitas masyarakat,\" kata Dirlantas Polda Jabar Kombes Pol Sugihardi di Mapolda Jabar, Jalan Soekarno Hatta, Kota Bandung, Rabu Score : 4.74694		

Hasil analisa dari Tabel 5.13 nilai rouge yang dihasilkan berdasarkan nilai *silhouette* tertinggi jika dilakukan perbandingan tiap – tiap percobaan menunjukkan bahwa nilai rouge yang dihasilkan berdasarkan nilai *silhouette* tertinggi hasil ekstraksi *trending issue* tanpa pembuangan kata *trivial* lebih tinggi daripada berdasarkan nilai *silhouette* tertinggi hasil ekstraksi *trending issue* dengan pembuangan kata *trivial*. Rata – rata menunjukkan nilai rouge yang dihasilkan berdasarkan hasil ekstraksi *trending issue* lebih baik tanpa pembuangan kata *trivial* yaitu 0.30199.

5.3 Evaluasi Umum Skenario Uji Coba

Pada subbab ini akan dijelaskan analisa dari hasil ujicoba yang sudah dilakukan. Analisa ini akan dilakukan pada tiap skenario uji coba yang sudah dilakukan.

Skenario 1 melakukan analisa hasil pengujian *autocorrelation Wavelet Coefficients* menggunakan hasil *crawling* data Twitter yang menggunakan *keyword* atau kata kunci. Pada pengujian ini kata kunci yang digunakan adalah kata kunci "sanusi" yang sesuai dengan isu yang sedang terjadi pada waktu *crawling*. Kata kunci yang dimasukkan sudah sesuai dengan isu yang sedang berkembang, hal ini mengakibatkan penggunaan *wavelet autocorrelation coefficient* menjadi tidak bermanfaat. Kata kunci "sanusi" yang seharusnya menjadi kata kunci penting atau *non trivial* menjadi kata kunci biasa. Hal ini dikarenakan setiap *tweets* yang mengandung kata "sanusi" akan diambil secara terus menerus sehingga mengakibatkan kata tersebut muncul secara berulang dan mengakibatkan kata "sanusi" menjadi kata dengan kemunculan paling tinggi. Dengan demikian dapat disimpulkan bahwa penggunaan *keyword* yang terlalu spesifik mengakibatkan kata – kata yang penting menjadi kata yang muncul secara periodik.

Skenario 2 melakukan analisa hasil pengujian *autocorrelation wavelet coefficients* tanpa menggunakan kata kunci tapi berdasarkan lokasi geografis Indonesia. Pada pengujian ini terlihat bahwa setiap *tweets* yang masuk berasal dari letak

geografis Indonesia, tetapi letak geografis Indonesia yang luas mengakibatkan terdapat beberapa *tweets* yang berasal dari negara tetangga Indonesia seperti Malaysia dan Singapura. Pengujian ini menunjukkan bahwa *tweets* yang masuk terlalu banyak dalam ruang lingkup yang terlalu besar mengakibatkan kata – kata yang di proses menjadi lebih banyak. Kata – kata yang dihasilkan tidak terkait pada satu topik saja sehingga tidak dapat disimpulkan apa yang sedang dibahas atau yang sedang ramai diperbincangkan.

Penentuan *confidence boundary* sebagai batas nilai untuk fungsi *autocorrelation* menunjukkan bahwa semakin tinggi batas yang ditentukan akan semakin sedikit jumlah nilai *autocorrelation* pada kata tersebut. Batas nilai *confidence boundary* yang ditentukan dari hasil pengujian adalah 0.20 dengan batas nilai *autocorrelation* 17. Penentuan batas nilai *confidence boundary* ini berdasarkan dari niai batas kepercayaan 95% dari *autocorrelation function*. Sedangkan batas nilai *autocorrelation* 17 dipilih menurut keterkaitan antar kata berdasarkan waktu.

Skenario 3 melakukan analisa hasil pengujian parameter untuk ekstraksi *trending issue.* Pada pengujian ini dibandingkan beberapa kombinasi 4 parameter yaitu jumlah kaster, TF *threshold, CI threshold, TFIDF threshold* dan WF *threshold.* Hasil perbandingan dari keempat parameter ini menunjukkan bahwa nilai *silhouette* yang paling tinggi pada ekstraksi *trending issue* dengan pembuangan kata *trivial* adalah 0.36322. Sedangkan ekstraksi *trending issue* tanpa pembuangan kata *trivial* menghasilkan nilai *silhouette* yang paling tinggi adalah 0.32623. Dari hasil nilai *silhouette* menunjukkan bahwa ekstraksi *trending issue* dengan pembuangan kata *trivial* menghasilkan nilai yang lebih baik.

Skenario 4 menunjukkan hasil ringkasan berdasarkan ekstraksi *trending issue* dengan dua kondisi. Kondisi pertama melihatkan hasil ringkasan berdasarkan nilai *silhouette* tertinggi hasil ekstraksi *trending issue* dengan dan tanpa pembuangan kata *trivial*. Kondisi kedua melihatkan hasil ringkasan berdasarkan nilai rouge tertinggi dengan hasil ekstraksi *trending issue* dengan dan

tanpa pembuangan kata *trivial*. Hasil analisa menunjukkan bahwa ringkasan berdasarkan hasil ektraksi *trending issue* lebih baik tanpa melakukan pembuangan kata *trivial* dengan rata – rata nilai rouge yang dihasilkan adalah 0.30199.

Pada Lampiran akan ditampilkan hasil Wavelet dan correlogram hasil perhitungan *autocorrelation wavelet coefficients* dari hasil *crawling* data yang memunujukkan kata yang dianggap penting dan dianggap tidak penting.

LAMPIRAN

Tabel WF kata kunci keyword 'sanusi'

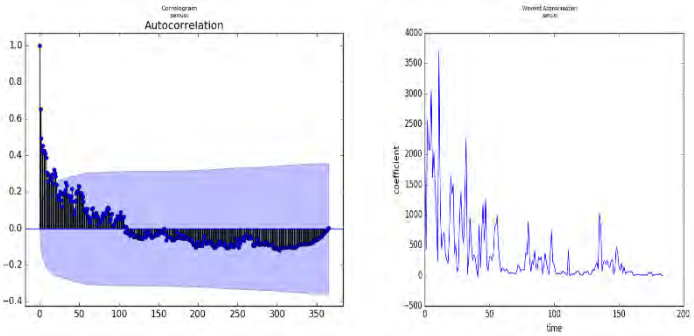
KATA	TF	KATA	TF	KATA	TF
sanusi	276	banget	59	waras	45
tangkap	197	followme	58	kayak	45
korupsi	163	baca	57	emang	45
ahok	142	kait	55	periksa	44
suap	137	tangan	55	taufik	44
kalo	107	newsupdate	55	hukum	44
salah	104	kota	55	fakta	44
maling	97	orange	54	tuju	44
news	94	reklamasi	53	lamido	43
tangkep	87	emir	53	jabat	43
berita	84	nasional	53	kalah	43
indonesia	80	teriak	53	moga	43
suci	80	biar	52	gtgt	43
bilang	79	teman	51	lucu	43
pake	78	makan	51	tuit	43
koruptor	78	penjara	51	duga	42
santun	76	populer	50	bikin	42
https	75	kakak	50	politis	42
tawur	73	ancam	49	hehe	42
tapak	73	takut	48	bagus	42
orang	71	podomoro	47	pakai	41
demo	71	libat	47	bersih	41
rompi	70	niat	47	tionsatu	41
korup	63	bongkar	46	hidup	40
kaya	61	jahat	46	dukun	40
teluk	60	sunny	45	metro	40
assadiqulhaq	59	islam	45	ntar	40
abang	59	tidur	45	alas	39

Tabel 0.1 Frekuensi Kemunculan Kata "sanusi"

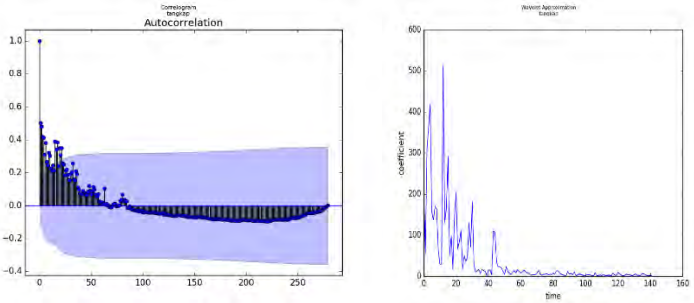
No	Interval Waktu	TF	No	Interval Waktu	TF
1.	2016-04-03 06.00	166	9.	2016-04-03 14.00	361
2.	2016-04-03 07.00	451	10.	2016-04-03 15.00	491
3.	2016-04-03 08.00	401	11.	2016-04-03 16.00	460
4.	2016-04-03 09.00	3736	12.	2016-04-03 17.00	571
5.	2016-04-03 10.00	1294	13.	2016-04-03 18.00	343
6.	2016-04-03 11.00	988	14.	2016-04-03 19.00	251
7.	2016-04-03 12.00	283	15.	2016-04-03 20.00	236
8.	2016-04-03 13.00	375	16.	2016-04-03 21.00	246

Tabel 0.2 Tabel *Wavelet coefficient* pada kata kunci "sanusi"

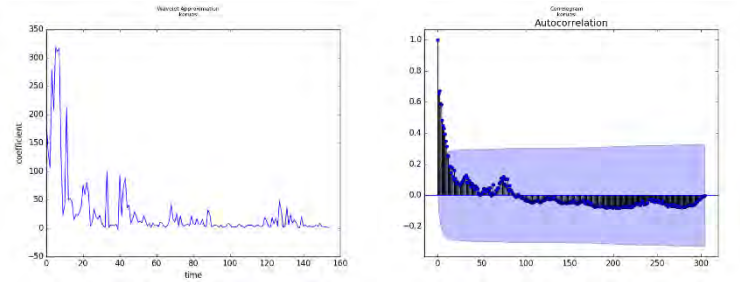
No	Nilai *Wavelet coefficient*	No	Nilai *Wavelet coefficient*
1.	252.36	9.	64.11
2.	213.34	10.	115.37
3.	660.65	11.	1021.92
4.	1646.74	12.	1373.39
5.	1341.25	13.	702.35
6.	1500.92	14.	523.71
7.	368.42	15.	1272.77
8.	515.16	16.	2270.77



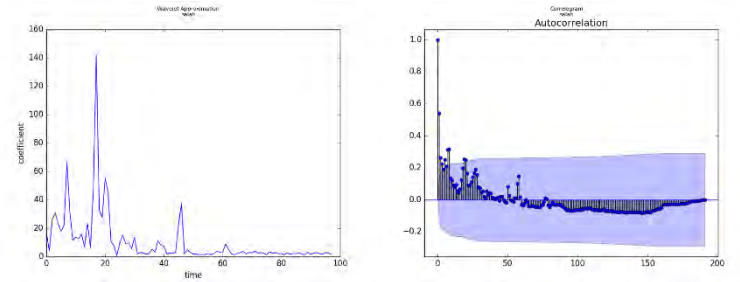
Gambar 0.1 Correlogram dan wavelet kata "sanusi" yang merupakan contoh kata yang dianggap tidak penting karena berulang secara periodik dan memiliki nilai *autocorrelation 31* dengan *confidence boundary* 0.20



Gambar 0.2 Correlogram dan Wavelet kata "tangkap" yang merupakan contoh kata yang dianggap tidak penting karena berulang secara periodik dan memiliki nilai *autocorrelation 29* dengan *confidence boundary* 0.20



Gambar 0.3 Correlogram dan Wavelet kata "korupsi" " yang merupakan contoh kata yang dianggap penting karena memiliki nilai *autocorrelation* yang rendah yaitu 13 dengan *confidence boundary* 0.20



Gambar 0.4 Correlogram dan Wavelet kata "salah" " yang merupakan contoh kata yang dianggap penting karena memiliki nilai *autocorrelation* yang rendah yaitu 10 dengan *confidence boundary* 0.20

Tabel WF kata kunci berdasarkan lokasi Indonesia (1)

KATA	TF	KATA	TF	KATA	TF
mall	77	gedung	52	long	45
centre	73	morning	51	panas	45
ayam	72	nasi	51	goodnight	45
plaza	70	kedai	51	malam	44
international	69	family	50	hospital	44
petaling	63	goreng	50	sman	44
coffee	63	putri	50	ikan	44

KATA	TF	KATA	TF	KATA	TF
bakso	63	salon	50	studio	44
pusat	62	chicken	50	salah	43
bangun	61	orang	49	kasih	43
airport	61	jakarta	49	hahahahaha	43
food	60	siang	49	resort	43
bakar	60	nonton	49	sini	43
kuala	59	pizza	49	market	43
lumpur	59	jabat	49	sushi	43
cafe	57	waroeng	49	sakit	42
nemenin	57	pustaka	49	kalo	42
makan	56	afood	48	muka	42
nner	56	burger	48	putrajaya	42
america	55	balikpapan	48	kantor	42
jaya	54	hiring	48	center	42
night	54	suka	47	bank	42
bismillah	54	susah	47	kampus	42
shopping	54	civil	47	universiti	42
sleep	54	sultan	47	tanah	42
captain	54	bandara	47	karaoke	42
aeon	54	universitas	46	indonesia	41
negeri	53	layan	46	hahahaha	41
watching	53	gitu	46	lupa	41
cream	53	langor	45	tengok	41
pagi	52	kena	45	lagu	41
esok	52	pantai	45	ajar	41

Tabel WF kata kunci berdasarkan lokasi Indonesia (2)

KATA	TF	KATA	TF	KATA	TF
makassar	41	hati	37	medan	35
university	41	restoran	37	sore	35
rvice	41	pergi	37	hehehe	35
bangsa	41	takde	37	hall	35
hidup	40	mesti	37	library	35

KATA	TF	KATA	TF	KATA	TF
masuk	40	malas	37	sihatan	35
fakultas	40	breakfast	37	smlm	35
barat	40	sarap	37	cerah	35
nunggu	40	office	37	main	34
sentral	40	sabah	37	pulak	34
tidur	39	tangerang	37	banget	34
dont	39	central	37	cakap	34
sayang	39	teknologi	37	kedah	34
city	39	good	36	seremban	34
kurang	39	biar	36	senang	34
harini	39	habis	36	bebek	34
cheras	39	bahagia	36	muhammadiyah	34
islam	39	jugak	36	tomyam	34
wisma	39	sarawak	36	shake	34
east	39	school	36	cari	33
pisang	39	wisata	36	tang	33
ramen	39	sibuk	36	mata	33
denpasar	39	benci	36	gila	33
puskesmas	39	manado	36	sehat	33
awake	39	nama	35	holiday	33
singapore	38	hehe	35	sepang	33
lunch	38	baby	35	tani	33
gunung	38	mati	35	wangsa	33
bilang	38	station	35	daerah	33
kantin	38	start	35	mara	33
cendol	38	kangen	35	johor	32

Tabel WF kata kunci berdasarkan lokasi Indonesia (3)

KATA	TF	KATA	TF	KATA	TF
libur	32	sains	31	island	29
bahru	32	fest	31	hahahahahaha	29
work	32	shah	30	nasional	29
awak	32	rindu	30	didikan	29

KATA	TF	KATA	TF	KATA	TF
pakai	32	malaysia	30	fitness	29
sabar	32	lepas	30	giant	29
park	32	sekolah	30	politeknik	29
ching	32	bikin	30	surakarta	29
ruang	32	guys	30	books	29
tomorrow	32	sedih	30	donuts	29
mandiri	32	suruh	30	time	28
teknik	32	hope	30	taman	28
building	32	negara	30	terima	28
institut	32	taknak	30	ople	28
selamat	31	ngapain	30	tido	28
tunggu	31	samarinda	30	surabaya	28
harap	31	wake	30	baca	28
jumpa	31	baso	30	yogyakarta	28
sembilan	31	institute	30	benda	28
sungai	31	laka	29	bogor	28
malang	31	bandar	29	mana	28
town	31	sikit	29	timur	28
lepak	31	house	29	hahahah	28
villa	31	weekend	29	baik	28
utara	31	waktu	29	kenal	28
convention	31	duit	29	senyum	28
menara	31	dunia	29	tangan	28
subuh	31	terminal	29	takpe	28
hukum	31	skrg	29	depok	28
petang	31	hahah	29	steak	28
campus	31	resto	29	golden	28
jobs	31	sedap	29	kompleks	28

Tabel WF kata kunci berdasarkan lokasi Indonesia (4)

KATA	TF	KATA	TF	KATA	TF
engineering	28	pasal	26	udara	25
early	28	room	26	spbu	25

KATA	TF	KATA	TF	KATA	TF
datar	28	rest	26	wolf	25
pontianak	28	court	26	posted	24
skin	28	ekonomi	26	beli	24
char	28	smkn	26	birthday	24
kerja	27	matahari	26	life	24
anak	27	science	26	warung	24
today	27	industri	26	pulau	24
wkwk	27	kaji	26	jawa	24
bareng	27	bandung	25	kuat	24
tweet	27	cerita	25	bagus	24
hmmm	27	cantik	25	final	24
dengar	27	antan	25	pastu	24
emang	27	kaya	25	duduk	24
blok	27	baju	25	paper	24
sian	27	mothers	25	feat	24
wilayah	27	exam	25	dear	24
sunway	27	girl	25	buta	24
tonight	27	sate	25	modus	24
rsud	27	comel	25	manhattan	24
balai	27	motor	25	reuni	24
klcc	27	federal	25	happymothersday	24
masyarakat	27	babi	25	kota	23
laboratorium	27	cuba	25	restaurant	23
akademi	27	kenyang	25	study	23
komunikasi	27	business	25	sweet	23
gdnight	27	assalamualaikum	25	lancar	23
moga	26	store	25	ilmu	23
subang	26	summer	25	kuliah	23
huhu	26	terjun	25	perempuan	23
nice	26	opening	25	amin	23

Tabel WF kata kunci berdasarkan lokasi Indonesia (5)

KATA	TF	KATA	TF	KATA	TF
test	23	solo	22	shop	21
nangis	23	butuh	22	mulu	21
song	23	suara	22	faculty	21
nyanyi	23	square	22	penyet	21
lembang	23	papa	22	village	21
uang	23	true	22	angkringan	21
budaya	23	kumpul	22	tidoq	21
starbucks	23	rame	22	fakulti	21
unit	23	malioboro	22	martabak	21
wisuda	23	boat	22	kampoeng	21
cinema	23	highlands	22	photo	20
suria	23	interview	22	bang	20
kawah	23	inti	22	kawan	20
pesta	23	camat	22	teman	20
pratama	23	aktivitas	22	stay	20
leni	23	provinsi	22	muhammad	20
woke	23	alam	21	kolej	20
geprek	23	rumah	21	kakak	20
supper	23	mama	21	cuti	20
semangat	22	kampung	21	late	20
tinggal	22	terengganu	21	syukur	20
kopi	22	besok	21	follow	20
puncak	22	twitter	21	manis	20
lapar	22	manusia	21	sahabat	20
feel	22	mood	21	cute	20
hilang	22	heart	21	pikir	20
maaf	22	bilik	21	kasi	20
hang	22	diam	21	lewat	20
adik	22	garden	21	sapa	20
nampak	22	territory	21	masak	20
alor	22	sape	21	hate	20

KATA	TF	KATA	TF	KATA	TF
ayah	22	graha	21	alun	20

Tabel 0.3 Frekuensi Kemunculan Kata "pagi"

No	Interval Waktu	TF	No	Interval Waktu	TF
1.	2016-05-08 00:00:00	69	13.	2016-05-08 12:00:00	44
2.	2016-05-08 01:00:00	61	14.	2016-05-08 13:00:00	29
3.	2016-05-08 02:00:00	59	15.	2016-05-08 14:00:00	20
4.	2016-05-08 03:00:00	44	16.	2016-05-08 15:00:00	17
5.	2016-05-08 04:00:00	82	17.	2016-05-08 16:00:00	28
6.	2016-05-08 05:00:00	307	18.	2016-05-08 17:00:00	24
7.	2016-05-08 06:00:00	577	19.	2016-05-08 18:00:00	20
8.	2016-05-08 07:00:00	617	20.	2016-05-08 19:00:00	12
9.	2016-05-08 08:00:00	561	21.	2016-05-08 20:00:00	14
10.	2016-05-08 09:00:00	265	22.	2016-05-08 21:00:00	24
11.	2016-05-08 10:00:00	138	23.	2016-05-08 22:00:00	18
12.	2016-05-08 11:00:00	45	24.	2016-05-08 23:00:00	44

Tabel 0.4 Frekuensi kemunculan kata "minggu"

No	Interval Waktu	TF	No	Interval Waktu	TF
1.	2016-05-08 00:00:00	72	13.	2016-05-08 12:00:00	59

No	Interval Waktu	TF	No	Interval Waktu	TF
2.	2016-05-08 01:00:00	40	14.	2016-05-08 13:00:00	59
3.	2016-05-08 02:00:00	24	15.	2016-05-08 14:00:00	59
4.	2016-05-08 03:00:00	9	16.	2016-05-08 15:00:00	45
5.	2016-05-08 04:00:00	13	17.	2016-05-08 16:00:00	45
6.	2016-05-08 05:00:00	37	18.	2016-05-08 17:00:00	43
7.	2016-05-08 06:00:00	100	19.	2016-05-08 18:00:00	51
8.	2016-05-08 07:00:00	186	20.	2016-05-08 19:00:00	35
9.	2016-05-08 08:00:00	201	21.	2016-05-08 20:00:00	31
10.	2016-05-08 09:00:00	125	22.	2016-05-08 21:00:00	23
11.	2016-05-08 10:00:00	118	23.	2016-05-08 22:00:00	30
12.	2016-05-08 11:00:00	80	24.	2016-05-08 23:00:00	32

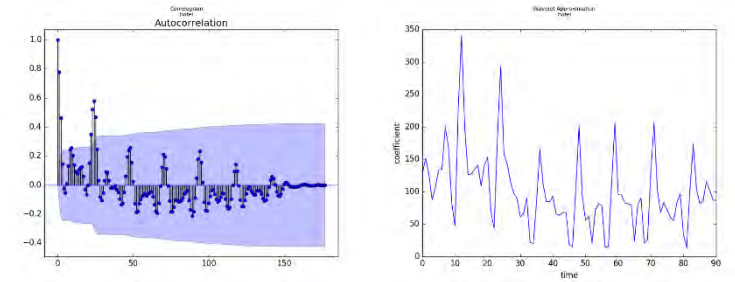
Tabel 0.5 Hasil *wavelet coefficient* kata "pagi"

No	Nilai *Wavelet coefficient*	No	Nilai *Wavelet coefficient*
1.	89.33	13.	32.67
2.	96.20	14.	100.50
3.	72.06	15.	61.13
4.	147.01	16.	138.84
5.	782.21	17.	841.17
6.	736.06	18.	797.68
7.	180.06	19.	229.62

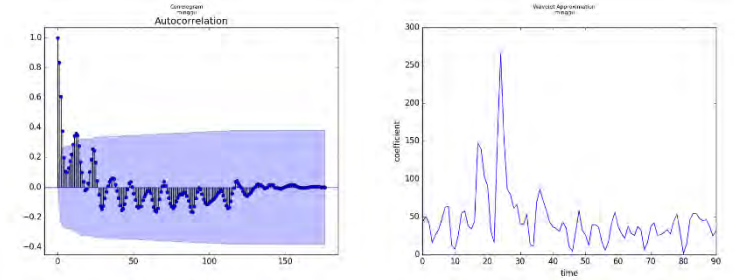
No	Nilai *Wavelet coefficient*	No	Nilai *Wavelet coefficient*
8.	52.12	20	41.46
9.	27.78	21.	29.12
10.	35.76	22.	35.39
11.	26.35	23.	31.23
12.	21.78	24.	18.17

Tabel 0.6 *Wavelet coefficient* kata "minggu"

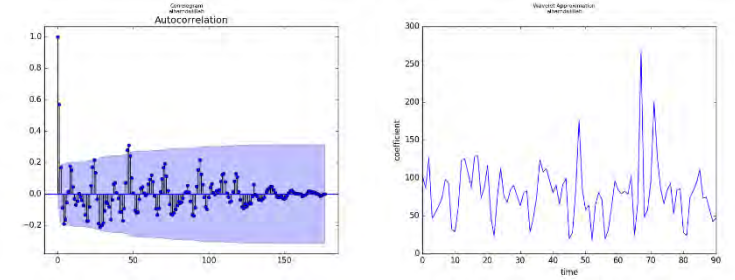
No	**Nilai *Wavelet coefficient***	**No**	**Nilai *Wavelet coefficient***
1.	63.41	13.	39.99
2.	96.31	14.	52.54
3.	30.68	15.	13.82
4.	16.43	16.	11.11
5.	151.82	17.	68.83
6.	265.22	18.	86.45
7.	153.80	19.	71.21
8.	86.46	20	58.25
9.	79.32	21.	42.92
10.	61.57	22.	36.76
11.	65.59	23.	34.62
12.	40.72	24.	30.44



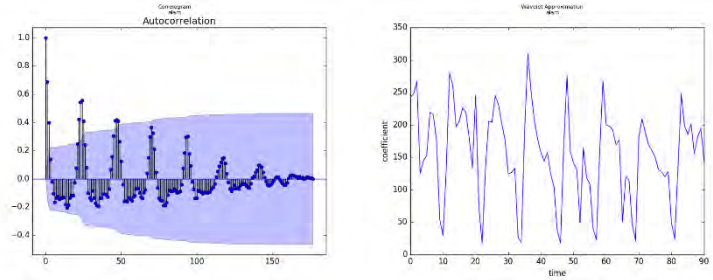
Gambar 0.5 Correlogram dan Wavelet kata "hotel" yang dianggap penting karena *lag – lag* berada didalam batas *confidence boundary* 0.20 dengan nilai *autocorrelation* 16.



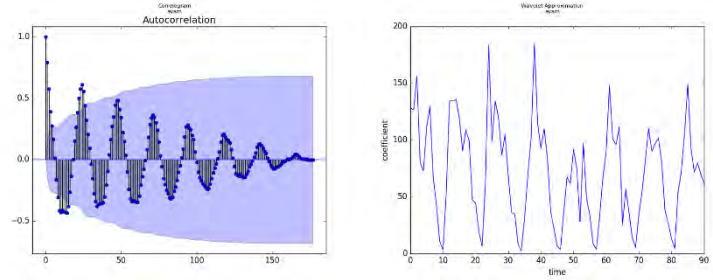
Gambar 0.6 Correlogram dan Wavelet kata "minggu" yang dianggap penting karena *lag – lag* berada didalam batas *confidence boundary* 0.20 dengan nilai *autocorrelation* 12.



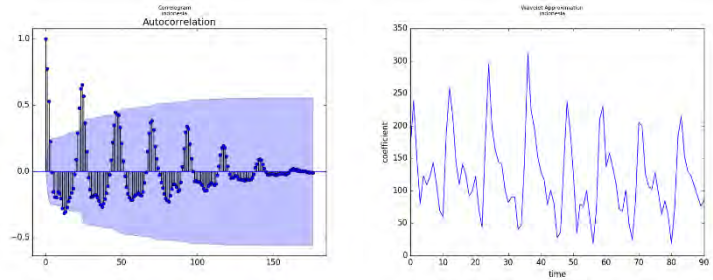
Gambar 0.7 Correlogram dan Wavelet kata "hotel" yang dianggap penting karena *lag – lag* berada didalam batas *confidence boundary* 0.20 dengan nilai *autocorrelation* 8.



Gambar 0.8 Correlogram dan Wavelet kata “alam” yang dianggap tidak penting karena *lag – lag* berada pada correlogram menunjukkan fluktuasi dari data secara periodik dengan nilai *autocorrelation* 21.



Gambar 0.9 Correlogram dan Wavelet kata “ayam” yang dianggap tidak penting karena *lag – lag* berada pada correlogram menunjukkan fluktuasi dari data secara periodik dengan nilai *autocorrelation* 72.



Gambar 0.10 Correlogram dan Wavelet kata “indonesia” yang dianggap tidak penting nilai *autocorrelation* 41

BAB VI
KESIMPULAN DAN SARAN

Bab ini berisikan kesimpulan yang dapat diambil dari hasil uji coba yang telah dilakukan. Selain kesimpulan, terdapat juga saran yang ditujukan untuk pengembangan perangkat lunak nantinya.

6.1 Kesimpulan

Kesimpulan yang diperoleh berdasarkan uji coba dan evaluasi yang telah dilakukan pada tugas akhir antara lain :

1. Penggunaan *keyword* yang terlalu spesifik tidak cocok untuk implementasi pemilihan kata kunci Twitter menggunakan *autocrrelation wavelet coefficient*
2. Penentuan lokasi geografis Indonesia yang terlalu luas mengakibatkan isu yang didapat terlalu banyak dan keterkaitan antar kata menjadi tidak jelas.
3. Penentuan *confidence boudary* yang paling baik adalah 0.2 berdasarkan pengamatan manual sesuai dengan nilai batas kepercayaan 95% pada *correlogram* dengan batas nilai *autocorrelation* adalah 17.
4. Nilai *silhouette* yang terbaik ditunjukkan pada hasil ekstraksi *trending issue* dengan pembuangan kata *trivial* yaitu 0.36322 dengan rata – rata nilai *silhouette* adalah 0.35708 yang mengindikasikan bahwa hasil klaster yang dihasilkan sudah cukup baik.
5. Ringkasan yang dihasilkan berdasarkan ekstraksi *trending issue* lebih baik tanpa melakukan pembuangan kata *trivial* dengan nilai rata – rata rouge yaitu 0.30199. Sedangkan nilai rouge yang dihasilkan berdasarkan ekstraksi *trending issue* dengan pembuangan kata *trivial* adalah 0.28816.

6.2 Saran

Saran yang diberikan terkait pengembangan pada Tugas Akhir ini adalah:

1. Penggunaan *keyword* yang lebih umum untuk mendapatkan isu lebih dianjurkan sebagai contoh kata kunci ‘sanusi’ merupakan kata kunci kasus korupsi, sehingga sebaiknya *keyword* yang digunakan adalah ‘korupsi’.
2. Penentuan lokasi pada proses *crawling* sebaiknya ditentukan berdasarkan lokasi yang lebih kecil sehingga kata yang masuk tidak beragam dan terlalu banyak.
3. Ringkasan berdasarkan *trending issue* akan lebih baik jika kata kunci *non trivial* yang didapat dilakukan berdasarkan penggunaan saran 1 dan saran 2 sekaligus.

DAFTAR PUSTAKA

[1] D. R. Radev, E. Hovy dan K. McKeown, "Introduction to the Special Issue on Summarization," *Computational Linguistics,* vol. 28, no. 4, pp. 339-408, December, 2002.

[2] N. Hayatin, C. Fatichah dan D. Purwitasari, "Pembobotan Kalimat Berdasarkan Fitur Berita dan Trending Issue untuk Peringkasan Multi Dokumen Berita," *Jurnal Ilmiah Teknologi Informasi,* vol. 13, pp. 152-159, 2015.

[3] R. S. Perdana, C. Fatichah dan D. Purwitasari, "Pemilihan Kata Kunci untuk Deteksi Kejadian Trivial pada Dokumen Twitter Menggunakan Autocorrelation Wavelet Analysis," *Jurnal Ilmiah Teknologi Informasi,* vol. 13, pp. 152-159, Juli, 2015.

[4] D. Kim, D. Kim, S. Kim, M. Jo dan E. Hwang, "SNS-based Issue Detection and Related News Summarization Scheme," dalam *ICUIMC'14 Proceedings of the 8th International Conference on Ubiquitous Information Management and Communication*, Siem Reap, Cambodia, 2014.

[5] A. Z. Arifin dan S. A. N, "Klasifikasi Dokumen Berita Kejadian Berbahasa Indonesia dengan Algoritma Single Pass Clustering," dalam *Proceeding of Seminar on Intelligent Technology and Its Applications (SITIA), Teknik Elektro, Institut Teknologi Sepuluh Nopember*, 2002.

[6] D. J. Foster, C. C. Mosher dan S. Hassanzadeh, "Wavelet transform methods for geophysical applications," dalam *Mathematical Methods in Geophysical Imaging*, December , 1993.

[7] J. Rafiee dan P. Tse, "Use of autocorrelation of wavelet coefficients for fault diagnosis," *Mechanical Systems and Signal Processing,* vol. 23, no. 5, pp. 1554-1572, Juli, 2009.

[8] M. Wheelwright Makridakis, "Metode dan Aplikasi peramalan," 1999.

[9] L. R. Kaufman, "Clustering by Means Medoids," *Statistical Data Analysis Based on the L1-Norm and Related Methode,* 1987.

[10] G. Salton dan C. Buckley, "TERM WEIGHTING APPROACHES IN AUTOMATIC TEXT RETRIEVAL," *Information Processing & Management2,* vol. 24, pp. 513-523, 1988.

[11] K. Sarkar, "Sentence Clustering-based Summarization of Multiple Text Documents," *Internasional Journal of Computing Science and Communication Technologies,* vol. 2, pp. 325 - 335, 2009.

[12] R. Ferreira, L. d. S. Cabral, R. D. Lins, G. P. e. Silva, F. Freitas, G. D. Cavalcanti, R. Lima, S. J. Simske dan Luciano, "Assessing sentence scoring techniques for extractive text summarization," *Expert Systems with Applications,* vol. 40, no. 14, pp. 5755-5764, Oktober, 2013.

[13] P. J. Rousseeuw, "Silhouette : a Graphical aid to the Interpretation and Validation of Cluster Analysis," *Journal of Computational and Applied Mathematics 20,* pp. 53 - 63, 1987.

[14] C.-Y. 2. ". Lin, "ROUGE: a Package for Automatic Evaluation of Summaries," dalam *In Proceedings of the Workshop on Text Summarization Branches Out (WAS 2004)*, Barcelona, Spain, 2004.

[15] MongoDB, "Introduction to MongoDB," MongoDB, 2008-2016. [Online]. Available: https://docs.mongodb.com/manual/introduction/. [Diakses 25 June 2016].

[16] MongoDB, Inc, "What is NoSQL," MongoDB, Inc, 2016. [Online]. Available: https://www.mongodb.com/nosql-explained?jmp=footer. [Diakses 20 June 2016].

BIODATA PENULIS

Oshi Prahtiwi Gusman merupakan anak dari Bapak Badarman dan Ibu Dra. Gusneli. Lahir di Pekanbaru pada tanggal 18 Januari 1994. Penulis menempuh pendidikan formal dimulai dari TK Riga PLN Pekanbaru (1999-2000), SD 001 Rintis Lima Puluh Pekanbaru (2000-2006), SMP Negeri 4 Pekanbaru (2006-2009), SMA Negeri 8 Peknabaru (2009-2012) dan S1 Teknik Informatika ITS (2012-2016). Bidang studi yang diambil oleh penulis pada saat berkuliah di Teknik Informatika ITS adalah Komputasi Cerdas dan Visi (KCV). Penulis aktif dalam organisasi seperti Ikatan Pelajar Mahasiswa Riau (2012-2016), Himpunan Mahasiswa Teknik Computer-Informatika (2013-2014) dan Volly TC (2013-2014). Penulis juga aktif dalam berbagai kegiatan kepanitiaan yaitu SCHEMATICS 2013 divisi REEVA dan SCHEMATICS 2014 divisi HUMAS. Penulis memiliki hobi travelling dan menyukai hal baru. Penulis dapat dihubungi melalui email: oshigusman@gmail.com