TWITTER *TEXT MINING* UNTUK INFORMASI GEMPA BUMI MENGGUNAKAN TF-IDF DI INDONESIA

SKRIPSI

Oleh:

RIJAALUL FATTAH
NIM. 09650040

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2016

TWITTER *TEXT MINING* UNTUK INFORMASI GEMPA BUMI MENGGUNAKAN TF-IDF DI INDONESIA

SKRIPSI

Oleh:

RIJAALUL FATTAH
NIM. 09650040

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM MALANG
2016

HALAMAN PENGAJUAN

TWITTER *TEXT MINING* UNTUK INFORMASI GEMPA BUMI MENGGUNAKAN TF-IDF DI INDONESIA

SKRIPSI

Diajukan kepada:
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang
Untuk Memenuhi Salah Satu Persyaratan Dalam
Memperoleh Gelar Sarjana Komputer (S.Kom)

Oleh:

RIJAALUL FATTAH
NIM. 09650040

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM MALANG
2016

HALAMAN PERSETUJUAN

TWITTER *TEXT MINING* UNTUK INFORMASI GEMPA BUMI MENGGUNAKAN TF-IDF DI INDONESIA

SKRIPSI

Oleh:

RIJAALUL FATTAH
NIM. 09650040

Telah Diperiksa dan Disetujui untuk Diuji:
Tanggal : 1 Juli 2016

Dosen Pembimbing I	Dosen Pembimbing II
Dr. M. Amin Hariyadi, M.T	**Ainatul Mardhiyah, M. Cs**
NIP. 19670118 200501 1 001	**NIPT. 20100301 2 147**

Mengetahui,
Ketua Jurusan Teknik Informatika

Dr. Cahyo Crysdian
NIP. 19740424 200901 1 008

HALAMAN PENGESAHAN

TWITTER *TEXT MINING* UNTUK INFORMASI GEMPA BUMI MENGGUNAKAN TF-IDF DI INDONESIA

SKRIPSI

Oleh:

RIJAALUL FATTAH
NIM. 09650040

Telah Dipertahankan di Depan Dewan Penguji Skripsi dan Dinyatakan Diterima
Sebagai Salah Satu Persyaratan Untuk Memperoleh Gelar Sarjana Komputer
Tanggal 15 Juli 2016

Susunan Dewan Penguji			Tanda Tangan
1. Penguji Utama	:	A'la Syauqi, M. Kom NIP. 19771201 200801 1 007	( )
2. Ketua Penguji	:	Dr. Cahyo Crysdian NIP. 19740424 200901 1 008	( )
3. Sekretaris Penguji	:	Dr. M. Amin Hariyadi, M.T NIP. 19670118 200501 1 001	( )
4. Anggota Penguji	:	Ainatul Mardhiyah, M. Cs NIPT. 20100301 2 147	( )

Mengetahui,
Ketua Jurusan Teknik Informatika

Dr. Cahyo Crysdian
NIP. 19740424 200901 1 008

HALAMAN PERSEMBAHAN

Assalamuaikum Warahmatullahi Wabarakaatuh

Puji syukur kepada Sang Khalik yang Maha Berkehendak, Allah *Subhanahu Wa Ta'ala*. Atas Rahmat dan Karunia-Nya saya bisa menyelesaikan skripsi ini dengan baik. Sholawat dan salam semoga tetap tercurahkan kepada Nabi Muhammad *Shallallahu 'alaihi Wa Sallam* yang sekaligus sebagai suri tauladan buat umat manusia. Semoga saya termasuk hamba kepercayaan Tuhan dan umat yang selalu mengikuti Rasulnya. *Aamiin*.

Skripsi ini penulis persembahkan untuk ibunda Musringatun dan ayahanda Pratiknyo, atas segala kepercayaan yang diberikan kepada penulis untuk dapat menempuh studi di perguruan tinggi dan menyelesaikan skripsi, dan juga yang selalu menjadi motivasi dalam menyelesaiakan skripsi ini. Merekalah orang yang paling berpengaruh dalam hidup penulis.

Dengan selesainya masa studi ini, semoga penulis dapat menjadi anak yang berbakti kepada orang tua dan ilmu yang didapatkan bermanfaat.

PERNYATAAN KEASLIAN TULISAN

Saya yang bertanda tangan di bawah ini:

Nama	:	Rijaalul Fattah
NIM	:	09650040
Jurusan	:	Teknik Informatika
Fakultas	:	Sains dan Teknologi
Judul Penelitian	:	Twitter *Text Mining* untuk Informasi Gempa Bumi Menggunakan TF-IDF di Indonesia

Menyatakan dengan sebenarnya bahwa tugas akhir/skripsi yang saya tulis ini benar-benar merupakan hasil karya saya sendiri, bukan merupakan pengambil alihan data, tulisan atau pikiran orang lain yang saya akui sebagai hasil tulisan atau pikiran saya sendiri, kecuali dengan mencantumkan sumber cuplikan pada daftar pustaka. Apabila dikemudian hari terbukti atau dapat dibuktikan tugas akhir/skripsi ini hasil jiplakan, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Malang, 30 Juni 2016

Yang membuat pernyataan,

Rijaalul Fattah
NIM. 09650040

KATA PENGANTAR

Assalaamu'alaikum Warahmatullaahi Wabaarakaatuh

Syukur alhamdulilah penulis haturkan kepada Allah SWT atas rahmat, taufik serta hidayah-Nya, sehingga penulis mampu menyelesaikan peyusunan skripsi ini sebagai salah satu syarat untuk memperoleh gelar sarjana dalam bidang teknik informatika di Fakultas Sains dan Teknologi, Universitas Islam Negeri Maulana Malik Ibrahim Malang.

Penulis menyadari adanya banyak keterbatasan yang penulis miliki dalam proses penyusunan skripsi ini, sehingga penulis banyak mendapat bimbingan dan arahan dari berbagai pihak. Untuk itu ucapan terima kasih yang sebesar-besarnya dan penghargaan setinggi-tingginya penulis sampaikan terutama kepada :

1. Bapak Dr. M. Amin Hariyadi, M.T, selaku pembimbing dalam skripsi ini yang telah memberikan bimbingan dan pengarahan dalam proses penyelesaian skripsi ini.
2. Ibu Ainatul Mardhiyah, M.Cs, selaku pembimbing dalam skripsi ini yang telah memberikan bimbingan dan pengarahan dalam proses penyelesaian skripsi ini.
3. Ibu, Bapak dan seluruh keluarga besar yang selalu memberikan do'a dan motivasi dalam penyelesaian skripsi ini.
4. Prof. Dr. H. Mudjia Rahardjo, M.Si., selaku Rektor Universitas Islam Negeri Maulana Malik Ibrahim Malang.

5. Dr. Hj. Bayyinatul Muchtaromah., drh., M.Si selaku Dekan Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang.
6. Bapak Dr. Cahyo Crysdian, selaku Ketua Jurusan Teknik Informatika Universitas Islam Negeri Maulana Malik Ibrahim Malang.
7. Bapak Zainal Abidin, M.Kom, selaku dosen wali memberikan bimbingan dan pengarahan dalam skripsi ini.
8. Segenap sivitas akademika Jurusan Teknik Informatika, terutama seluruh dosen, terima kasih atas segenap ilmu dan bimbingannya.
9. Seluruh teman-teman Jurusan Teknik Informatika khususnya angkatan 2009.
10. Semua pihak dan seluruh alam semesta, penulis ucapkan terima kasih yang sebesar-besarnya.

Sebagai penutup, penulis menyadari dalam skripsi ini masih banyak kekurangan dan jauh dari sempurna, untuk itu penulis selalu menerima segala kritik dan saran dari pembaca. Harapan penulis, semoga karya ini bermanfaat bagi kita semua.

Wasslaamu'alaikum Warahmatullahi Wabarakaatuh

Malang, 30 Juni 2016
Hormat Saya,

Rijaalul Fattah
NIM. 09650040

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR SOURCE CODE

ABSTRAK

Fattah, Rijaalul. 2016. **Twitter *Text mining* untuk Informasi Gempa Bumi Menggunakan TF-IDF di Indonesia**. Skripsi. Jurusan Teknik Informatika Fakultas Sains Dan Teknologi. Universitas Islam Negeri Maulana Malik Ibrahim, Malang.

Pembimbing : (I) Dr. M. Amin Hariyadi, M.T (II) Ainatul Mardhiyah, M.Cs

Kata Kunci: Twitter, *Text mining*, Gempa Bumi, TF-IDF.

Text mining atau Penambangan Teks adalah proses yang dilakukan oleh komputer untuk mendapatkan sesuatu yang baru, atau menemukan kembali informasi yang tersirat secara implisit.

Penelitian ini membahas mengenai penambangan teks kemudian disaring untuk mendapatkan informasi mengenai gempa bumi di Indonesia. Tujuan yang ingin didapatkan adalah memperoleh informasi tentang terjadinya gempa bumi dari media sosial secara akurat. Aplikasi ini dibangun menggunakan bahasa Python sebagai bahasa pemrogramannya. Untuk uji cobanya dilakukan dengan men-*stream web* Twitter menggunakan kata kunci "gempa" kemudian di-*crawl.* Dokumen yang dihasilkan dari proses stream twitter API dilakukan penghitungan bobot tf-idf.

Dari hasil uji coba pada proses uji coba, didapatkan bahwa proses stream menghasilkan 520 dokumen *tweet* dengan kata kunci gempa. Kemudian oleh aplikasi dokumen tersebut dihitung kemunculan *term* gempa pada setiap dokumen. Dari 520 dokumen *tweet* yang dihasilkan, frekuensi rata-rata tweet yang dihasilkan dalam satu menit adalah 1 tweet dengan bobot tf-idf 0.036. Frekuensi kemunculan *term* divisualkan dalam bentuk *chart time series* yang menunjukkan kemunculan *term* gempa pada tiap menit.

ABSTRACT

Fattah, Rijaalul. 2016. **Twitter Text Mining for Earthquake Information using TF-IDF in Indonesia**. Thesis. Informatic Engineering. Faculty of Science and Technology Islamic State University Maulana Malik Ibrahim o Malang.
Lecturer : (I) Dr. M. Amin Hariyadi, M.T (II) Ainatul Mardhiyah, M.Cs

Key Words: Twitter, *Text Mining*, Earth quake, TF-IDF.

Text Mining is the process that performed by a computer to get something new, or find back information that implied implicitly.

This research discussed about text mining then filtered it to get information of the earth quake in Indonesia. The goal is to get obtain information about the earthquake of social media accurately . This application is build on Python as the programming language. The test process is by doing some twitter stream using keyword "gempa" then crawl it. Documents produced from the twitter API stream process count of tf-idf weight.

From the trial at process trial, got that process stream produce 520 documents tweets keyword earthquake. Then by the application of the documents calculated the term earthquake in each document. From the 520 documents of the tweet that produced, the average frecuency of tweets generated in one minute is one tweet with tf-idf weight of 0.036. Frequency of occurrence visualized term in the chart time series showing the term earthquake in every minutes.

مُلَخَّص

فتاح, رجالل. 2016. تويتر النص التءدين لمءلومات الزلازل باستخدام TF-IDFالاءسر اءيل في اندونسيا. مقال. كلية الهندسة المعلوماتية في الجامعات الإسلامية العلوم والتكنولوجيا الدولة مولانا مالك إبراهيم مالا غ.

مؤدّب. (I) Dr. M. Amin Hariyadi, M.T (II) Ainatul Mardhiyah, M.Cs

---

كلمات البحت: توىتر والتءدين الينص, TF-IDFالاسر اديلي

التعدين النص هو تنفذ العلميه بالكمبيوتر لتحصل ما الجديد أو لتحصل سر البيانات مرة ضمنا.

هذاالبحوث يفيض التعدين النص ثم يرشح لتحصل البيانات عن الزلزال في الأندونيسي والهدف ليحصل البيانات وقعت الزلزال عن وسائل الاجتماعي مضبوطا. استمارة النجمعات تقام بمستعملة اللغة الفيطون python كما لغة برمجتها فتجربتها تنفذ بالتيارات stream المواقع التغريد web-twitter و مستخدمها الكلمة الطنانة "زلزال" ثم مزواحفها crawl.

ونتيجة تجربتها لعملية التجربة توجد أنها عملية التيارات تنتاج 520 ورقة التغريد بكلمة الزلزال من كل الأوراق. و ترددات termالطنانة "زلزال". ثم استمارة الورقة تحسب انبثاق المصطلح التي تدل انبثاق المصطلح chart time series الاننثاق المصطلح متصور في شكل الوقت سلسلة الرسم الزلزال كل دقيقها

BAB I

PENDAHULUAN

1.1 Latar Belakang

Internet dan media sosial kini sudah menjadi arus baru bagi masyarakat untuk mendapatkan informasi tentang apa saja. Di Indonesia media sosial juga marak digunakan oleh masyarakat. Twitter merupakan salah satu media sosial penyedia layanan *micro blogging* bagi para penggunanya. Saat ini pengguna Twitter di Indonesia mencapai 50 juta pengguna.

Setiap *tweet* atau cuitan pengguna ini mengandung data yang apabila dikumpulkan dapat diolah menjadi informasi. Pengguna Twitter selalu cepat dalam mengabarkan suatu peristiwa atau kejadian, misalnya gempa bumi. Indonesia juga merupakan Negara yang berada di atas pertemuan antara lempengan dua benua yang menjadikannya Negara rawan gempa bumi.

Selama ini informasi mengenai gempa bumi atau bencana alam lainnya selalu disampaikan oleh Badan Meteorologi Klimatologi dan Geofisika atau BMKG. Kehadiran media sosial memberikan alternatif baru bagi masyarakat untuk mendapatkan informasi tentang gempa bumi atau bencana alam lainnya dengan mudah. Bahkan masyarakat pengguna twitter dapat saling bertukar informasi mengenai berbagai informasi, salah satu yang pasti, gempa bumi. Kebiasaan pengguna sosial media seperti ini lah yang kadang dianggap tidak terlalu penting untuk disikapi, mereka dianggap terlalu banyak bicara, bahkan disaat terjadi bencana. Namun, apabila dipelajari lebih lanjut, kebiasaan berbagi kabar melalui

sosial media twitter ini dapat dimanfaatkan juga. Misalkan untuk informasi terjadinya gempa bumi dari seluruh Indonesia. Karena setiap *tweet* yang mereka sampaikan merupakan data yang dapat diolah. Dan apabila yang menyampaikan kabar lebih banyak, maka data bias dikatakan valid. Dalam hal ini apabila terjadi gempa dan banyak orang yang mengabarkan tentang terjadinya gempa di twitter, itu berarti memang benar-benar terjadi gempa bumi ytang dirasakan oleh banyak orang.

Dalam al-Qur'an juga mengabarkan bahwa akan terjadi bencana alam yang dahsyat untuk dihadapi oleh umat manusia, oleh karena itu, kita sebagai manusia harus senantiasa siap akan segala hal yang akan terjadi. Informasi mengenai gempa bumi ini bias kita manfaatkan sebagai persiapan menghadapi bencana alam. Seperti yang tertuang pada surah Al-Zalzalah :

إِذَا زُلۡزِلَتِ ٱلۡأَرۡضُ زِلۡزَالَهَا ۝١ وَأَخۡرَجَتِ ٱلۡأَرۡضُ أَثۡقَالَهَا ۝٢ وَقَالَ ٱلۡإِنسَٰنُ مَا لَهَا ۝٣

Apabila bumi digoncangkan dengan goncangannya, dan bumi telah mengeluarkan beban-beban beratnya, dan manusia bertanya: "Apa (yang terjadi) baginya?"

Surah **الزلزلة** memiliki arti "goncangan", goncangan yang dimaksudkan adalah gempa bumi. Dijelaskan dalam tafsir Al-Misbah volume 15 bahwa surah ini berbicara tentang awal terjadinya hari kemudian. Gempa bumi adalah peristiwa bergetarnya bumi akibat pelepasan energi di dalam bumi secara tiba-tiba yang

ditandai dengan patahnya lapisan batuan pada kerak bumi. Akumulasi energi penyebab terjadinya gempabumi dihasilkan dari pergerakan lempeng-lempeng tektonik. Energi yang dihasilkan dipancarkan kesegala arah berupa gelombang gempabumi sehingga efeknya dapat dirasakan sampai ke permukaan bumi.

Twitter *text mining* merupakan sebuah sistem yang dapat menggali data dari twitter untuk kemudian diolah dan dijadikan informasi tentang terjadinya gempa bumi di Indonesia. Melalui Twitter *text mining* dapat memberikan informasi mengenai gempa bumi di Indonesia bagi masyarakat secara cepat.

1.2 Rumusan Masalah

Seberapa akurasi Term Frequency – Inverse Document Frequency (TF-IDF) apabila digunakan untuk *text minig*?

1.3 Batasan Masalah

Batasan masalah yang akan dibahas dalam penyelesaian penelitian yang dilakukan ini adalah :

1. Data yang digunakan adalah *tweet* dari pengguna Twitter yang menyebutkan tentang gempa bumi dan nama kabupaten/kota di Indonesia.
2. Pengolahan data pada sistem ini menggunakan metode Term Frequency, Inverse Document Frequency (TF-IDF).
3. Sistem dibangun menggunakan bahasa pemrograman Python.

1.4 Tujuan Penelitian

Adapun tujuan penelitian ini yaitu membuat sebuah sistem yang dapat menggali data dari twitter untuk kemudian diolah dan dijadikan informasi tentang terjadinya gempa bumi di Indonesia.

1.5 Manfaat Penelitian

Manfaat pembuatan aplikasi ini adalah dapat digunakan sebagai sumber informasi mengenai gempa bumi di Indonesia bagi masyarakat.

1.6 Metode Penelitian

1. Analisis Data

Pada tahap ini, peneliti melakukan proses analisis kebutuhan secara keseluruhan dari sistem yang akan dibangun. Penelitian ini menggunakan data dari twit pengguna Twitter yang menyebutkan tentang terjadinya gempa bumi di suatu daerah di Indonesia dan akan diolah dalam sistem penambangan teks yang akan menghasilkan informasi gempa bumi yang diharapkan.

2. Perancangan (Desain Sistem)

Pada tahap ini merupakan tahap perancangan sistem berdasarkan spesifikasi yang telah ditentukan sebelumnya. Perancangan sistem ini terfokus pada struktur data, arsitektur software, dan algoritma yang digunakan.

3. Pembuatan Kode Program

Pembuatan kode program ini menggunakan *Notepadqq* sebagai media pengkodean dan menggunakan bahasa pemrograman Python.

4. Uji Coba Aplikasi

Uji coba kali ini digunakan untuk menguji apakah hasil dari aplikasi ini sesuai yang diharapkan. Nilai yang terbentuk dikatakan valid jika sesuai dengan rule-rule yang sudah ditentukan.

5. Penulisan Laporan

Penulisan laporan digunakan untuk membuat dokumentasi berupa laporan yang berisi seluruh proses penelitian yang telah dilakukan. Laporan ini diharapkan bermanfaat dan mempermudah untuk penelititan terkait selanjutnya.

1.7 Sistematika Penulisan

Dalam penulisan skripsi ini, secara keseluruhan terdiri dari lima bab yang masing-masing bab disusun dengan sistematika sebagai berikut:

BAB I PENDAHULUAN

Bab ini merupakan bagian awal, bab ini berisi latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian dan sistematika penulisan laporan.

BAB II TINJAUAN PUSTAKA

Bab ini berisi tentang teori-teori yang berhubungan dengan permasalahan yang diangkat dari penelitian ini, teori-teori tersebut antara lain *text mining* dan tf-idf

BAB III ANALISIS DAN PERANCANGAN

Bab ini menjelaskan tentang analisis dan perancangan tentang twitter *text mining* dan tf-idf yang meliputi tahapan penelitian, tahapan pembuatan sistem, dan pembuatan program.

BAB IV HASIL DAN PEMBAHASAN

Bab ini berisi tentang penjelasan keseluruan sistem yang telah dibuat dan hasil pengujian yang sudah dilakukan sesuai dengan studi kasus.

BAB V PENUTUP

Bab terakhir berisi kesimpulan berdasarkan hasil yang telah dicapai dari pembahasan. Selain itu juga berisi saran yang diharapkan dapat digunakan sebagai bahan pertimbangan bagi peneliti selanjutnya yang akan mengembangkan penelitian serupa.

BAB II

TINJAUAN PUSTAKA

2.1 *Text mining*

Text mining (penambangan teks) adalah penambangan yang dilakukan oleh komputer untuk mendapatkan sesuatu yang baru, sesuatu yang tidak diketahui sebelumnya atau menemukan kembali informasi yang tersirat secara implisit, yang berasal dari informasi yang diekstrak secara otomatis dari sumber-sumber data teks yang berbeda-beda (Feldman & Sanger, 2007). *Text mining* merupakan teknik yang digunakan untuk menangani masalah klasifikasi, *clustering*, *information extraction* dan *information retrival* (Berry & Kogan, 2010).

Pada dasarnya proses kerja dari *text mining* banyak mengapdopsi dari penelitian *Data Mining* namun yang menjadi perbedaan adalah pola yang digunakan oleh *text mining* diambil dari sekumpulan bahasa alami yang tidak terstruktur sedangkan dalam *Data Mining* pola yang diambil dari *database* yang terstruktur (Han & Kamber, 2006). Tahap-tahap *text mining* secara umum adalah *text preprocessing* dan *feature selection* (Feldman & Sanger 2007, Berry & Kogan 2010) . Dimana penjelasan dari tahap-tahap tersebut adalah sebagai berikut :

2.1.1 Text Preprocessing

Tahap *text preprocessing* adalah tahap awal dari *text mining*. Tahap ini mencakup semua rutinitas, dan proses untuk mempersiapkan data yang akan digunakan pada operasi *knowledge discovery* sistem *text mining* (Feldman &

Sanger, 2007). Tindakan yang dilakukan pada tahap ini adalah *toLowerCase*, yaitu mengubah semua karakter huruf menjadi huruf kecil dan *Tokenizing* yaitu proses penguraian deskripsi yang semula berupa kalimat-kalimat menjadi kata-kata dan menghilangkan delimiter-delimiter seperti tanda titik (.), koma (,), spasi dan karakter angka yang ada pada kata tersebut (Weiss et al, 2005).

2.1.2 Feature Selection

Tahap seleksi fitur (*feature selection*) bertujuan untuk mengurangi dimensi dari suatu kumpulan teks, atau dengan kata lain menghapus kata-kata yang dianggap tidak penting atau tidak menggambarkan isi dokumen sehingga proses pengklasifikasian lebih efektif dan akurat (Do et al, 2006., Feldman & Sanger, 2007., Berry & Kogan 2010). Pada tahap ini tindakan yang dilakukan adalah menghilangkan *stopword* (*stopword removal*) dan *stemming* terhadap kata yang berimbuhan (Berry & Kogan 2010., Feldman & Sanger 2007).

Stopword adalah kosakata yang bukan merupakan ciri (kata unik) dari suatu dokumen (Dragut et al. 2009). Misalnya “di”, “oleh”, “pada”, “sebuah”, “karena” dan lain sebagainya. Sebelum proses *stopword removal* dilakukan, harus dibuat daftar *stopword* (*stoplist*). Jika termasuk di dalam *stoplist* maka kata-kata tersebut akan dihapus dari deskripsi sehingga kata-kata yang tersisa di dalam deskripsi dianggap sebagai kata-kata yang mencirikan isi dari suatu dokumen atau *keywords*. Daftar kata *stopword* di penelitian ini bersumber dari Tala (2003).

Setelah melalui proses *stopword removal* tindakan selanjutnya adalah yaitu proses *stemming*. *Stemming* adalah proses pemetaan dan penguraian berbagai bentuk (*variants*) dari suatu kata menjadi bentuk kata dasarnya *(stem)*

(Tala, 2003). Tujuan dari proses *stemming* adalah menghilangkan imbuhan-imbuhan baik itu berupa prefiks, sufiks, maupun konfiks yang ada pada setiap kata. Jika imbuhan tersebut tidak dihilangkan maka setiap satu kata dasar akan disimpan dengan berbagai macam bentuk yang berbeda sesuai dengan imbuhan yang melekatinya sehingga hal tersebut akan menambah beban *database*. Hal ini sangat berbeda jika menghilangkan imbuhan-imbuhan yang melekat dari setiap kata dasar, maka satu kata dasar akan disimpan sekali walaupun mungkin kata dasar tersebut pada sumber data sudah berubah dari bentuk aslinya dan mendapatkan berbagai macam imbuhan. Karena bahasa Indonesia mempunyai aturan morfologi maka proses *stemming* harus berdasarkan aturan morfologi bahasa Indonesia.

Berdasarkan penelitian sebelumnya, ada beberapa algoritma stemming yang bisa digunakan untuk *stemming* bahasa Indonesia diantaranya algoritma *confix-stripping*, algoritma Porter *stemmer* bahasa Indonesia, algoritma Arifin dan Sutiono, dan Algorima Idris (Tala 2003, Agusta 2009, Asian et al 2005, Adriani et al 2007). Dimana, Algoritma *confix-stripping* adalah algoritma yang akurat dalam *stemming* bahasa Indonesia (Tala 2003, Agusta 2009, Asian et al 2005, Adriani et al 2007).

2.2 Stemming

Stemming dilakukan untuk mengubah kata berimbuhan menjadi kata dasarnya. Misalnya ‘berlari’ menjadi ‘lari’, ‘pemrosesan’ menjadi ‘proses’ dan seterusnya. Dengan demikian, kualitas informasi meningkat. Kualitas informasi yang dimaksud adalah hubungan antar kata itu sendiri, misalnya ‘memberi’,

‘diberi’, ‘memberikan’, yang semula adalah kata yang berbeda, dengan adanya stemming, kata tersebut menjadi ‘beri’, sehingga ada hubungan antara satu sama lain. Selain itu, space yang digunakan untuk penyimpanan juga menjadi lebih kecil. Terdapat 5 langkah pembangunan inverted index, yaitu (Manning, 2009):

1. **Penghapusan format dan markup dari dalam dokumen**

   Jika dokumen yang digunakan bukan berupa teks murni maka tahap ini dilakukan. Karena dokumen teks yang biasanya kita lihat berupa format non teks seperti html, pdf atau dalam bentuk word. Format-format ini mengharuskan sebuah teks dilengkapi unsur-unsur tambahan untuk dapat menghasilkan tampilan yang friendly dimata kita. Informasi-informasi itu dihilangkan karena dianggap tidak perlu dan tidak mencerminkan isi sebuah dokumen teks.

2. **Pemisahan rangkaian kata (*tokenization*)**

   Proses tokenizing adalah proses pemotongan string masukan berdasarkan tiap kata yang menyusunnya. Pada prinsipnya proses ini memisahkan setiap kata yang menyusun dokumen. Pada umumnnya setiap kata terindentifikasi dengan kata yang lain oleh karakter spasi, sehingga proses tokenizing mengandalkan karakter spasi pada dokumen untuk pemisahan kata. Pada proses tokenizing biasanya juga ditambahkan informasi jumlah kemunculan kata pada kalimat tersebut.

3. **Penyaringan (*filtration*)**

   Proses Filtering adalah proses pengambilan kata-kata yang

dianggap penting atau mempunyai makna saja. Pada proses ini kata-kata yang dianggap tidak mempunyai makna seperti kata sambung akan dihilangkan. Pada proses ini biasanya digunakan daftar stopword yang tersimpan dalam suatu tabel basis data, yang nantinya digunakan sebagai acuan penghilangan kata. Daftar stopword berbeda untuk setiap bahasanya.

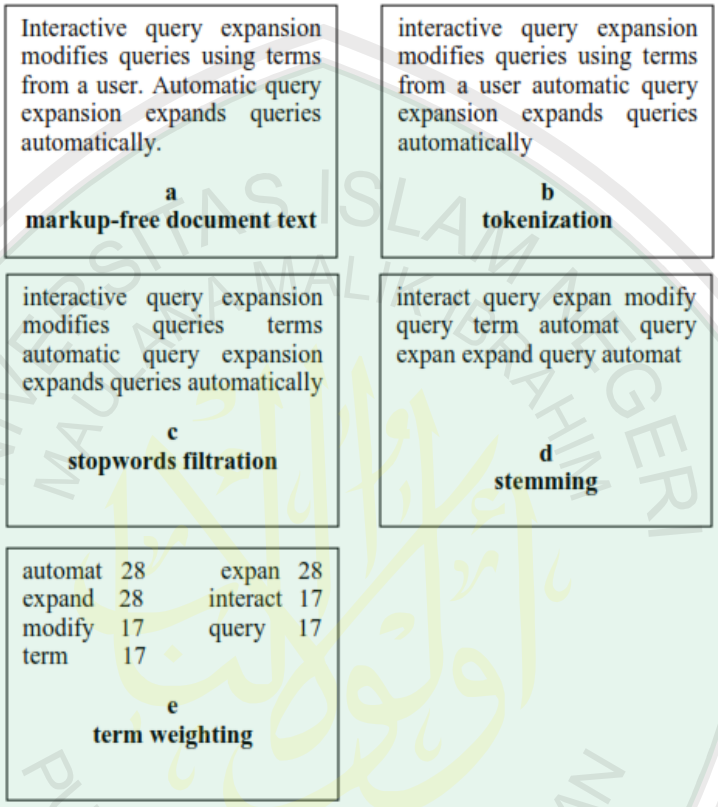
4. Konversi *term* ke bentuk dasar (*stemming*)

Stemming adalah proses konversi *term* ke bentuk umumnya, sebagaimana dijelaskan sebelumnya. Dokumen dapat pula diekspansi dengan mencarikan sinonim bagi term-term tertentu di dalamnya. Sinonim adalah kata-kata yang mempunyai pengertian serupa tetapi berbeda dari sudut pandang morfologis. Seperti *stemming*, operasi ini bertujuan menemukan suatu kelompok kata terkait. Akan tetapi sinonim bekerja berdasarkan pada *thesaurus*, tidak berbagi-pakai term stem. Jika pengguna memasukkan query "heart disease" maka query diekspansi untuk mengakomodasi semua sinonim dari *disease* seperti *ailment*, *complication*, *condition*, *disorder*, *fever*, *ill*, *illness*, *infirmity*, *malady*, *sickness*, dan lain-lain (Cios, 2007).

5. Pemberian bobot terhadap term (*weighting*)

Dimulai dengan perhitungan jumlah kata dalam setiap dokumen, yang kemudian akan dihitung menggunakan skema pembobotan yang dikehendaki. Setiap *term* diberikan bobot sesuai dengan skema pembobotan yang dipilih, apakah pembobotan lokal, global atau

kombinasi keduanya. Banyak aplikasi menerapkan pembobotan kombinasi berupa perkalian bobot lokal *term frequency* dan *global inverse document frequency*, ditulis tf .idf.



Gambar 2. 1 Contoh lima tahap indexing berbasis content
(Sumber : komputasi.wordpress.com, 2010)

2.3 Stopword

Stopword merupakan kata-kata yang sangat sering muncul dalam dokumen. Stopword ini tidak bisa dikatakan sebagai kata-kata yang tidak berpengaruh terhadap proses kategorisasi. Misalnya adalah kata penghubung seperti ‘dan’, ‘atau’, ‘kemudian’, dan seterusnya. Selain itu, kata depan juga merupakan stopword yang tidak memiliki arti penting bagi dokumen itu sendiri. Oleh karena itu, stopword harus dihilangkan. Penghilangan stopword ini

dilakukan berdasarkan kamus kata tertentu yang disebut dengan database stopword.

2.4 TF-IDF

Sistem Temu Kembali Informasi berhadapan dengan pencarian informasi yang sesuai dengan *query* pengguna dari koleksi dokumen. Koleksi dokumen tersebut terdiri dari dokumen-dokumen yang beragam panjangnya dengan kandungan *term* yang berbeda pula. Hal yag perlu diperhatikan dalam pencarian informasi dari koleksi dokumen yang heterogen adalah pembobotan *term*. *Term* dapat berupa kata, frase atau unit hasil *indexing* lainnya dalam suatu dokumen yang dapat digunakan untuk mengetahui konteks dari dokumen tersebut. Karena setiap kata memiliki tingkat kepentingan yang berbeda dalam dokumen, maka untuk setiap kata tersebut diberikan sebuah indikator, yaitu *term weight*. (Zafikri, 2010)

Term weighting atau pembobotan *term* sangat dipengaruhi oleh hal-hal berikut ini (Mandala, 2004):

1. *Term Frequency* (tf) *factor*, yaitu faktor yang menentukan bobot *term* pada suatu dokumen berdasarkan jumlah kemunculannya dalam dokumen tersebut. Nilai jumlah kemunculan suatu kata (*term frequency*) diperhitungkan dalam pemberian bobot terhadap suatu kata. Semakin besar jumlah kemunculan suatu *term* (tf tinggi) dalam dokumen, semakin besar pula bobotnya dalam dokumen atau akan memberikan nilai kesesuian yang semakin besar.
2. *Inverse Document Frequency* (idf) *factor*, yaitu pengurangan dominansi *term* yang sering muncul di berbagai dokumen. Hal ini diperlukan karena *term* yang

banyak muncul di berbagai dokumen, dapat dianggap sebagai *term* umum (*common term*) sehingga tidak penting nilainya. Sebaliknya faktor kejarangmunculan kata (*term scarcity*) dalam koleksi dokumen harus diperhatikan dalam pemberian bobot. Menurut Mandala (dalam Witten, 1999) 'Kata yang muncul pada sedikit dokumen harus dipandang sebagai kata yang lebih penting (*uncommon tems*) daripada kata yang muncul pada banyak dokumen. Pembobotan akan memperhitungkan faktor kebalikan frekuensi dokumen yang mengandung suatu kata (*inverse document frequency*). Hal ini merupakan usulan dari George Zipf. Zipf mengamati bahwa frekuensi dari sesuatu cenderung kebalikan secara proposional dengan urutannya.

Metode TF-IDF merupakan metode pembobotan *term* yang banyak digunakan sebagai metode pembanding terhadap metode pembobotan baru. Pada metode ini, perhitungan bobot *term t* dalam sebuah dokumen dilakukan dengan mengalikan nilai *Term Frequency* dengan *Inverse Document Frequency.*

Pada *Term Frequency* (tf), terdapat beberapa jenis formula yang dapat digunakan yaitu (Mandala, 2004):

1. tf biner (*binery tf*), hanya memperhatikan apakah suatu kata ada atau tidak dalam dokumen, jika ada diberi nilai satu, jika tidak diberi nilai nol
2. tf murni (raw tf), nilai tf diberikan berdasarkan jumlah kemunculan suatu kata di dokumen. Contohnya, jika muncul lima kali maka kata tersebut akan bernilai lima.
3. tf logaritmik, hal ini untuk menghindari dominansi dokumen yang mengandung sedikit kata dalam *query*, namun mempunyai frekuensi yang tinggi.

$$tf = 1 + log\ (tf) \qquad (1)$$

4. tf normalisasi, menggunakan perbandingan antara frekuensi sebuah kata dengan jumlah keseluruhan kata pada dokumen.

$$\text{tf} = 0.5 + 0.5 \text{ x } \left(\frac{tf}{\max tf}\right) \qquad (2)$$

Inverse Document Frequency (idf) dihitung dengan menggunakan formula

$$idf_j = log\ (D\ /df_j) \qquad (3)$$

dimana

D adalah jumlah semua dokumen dalam koleksi

Df_j adalah jumlah dokumen yang mengandung term t_j

Menurut Defeng (dalam Robertson, 2004) 'Jenis formula yang akan digunakan untuk perhitungan *term frequency* (tf) yaitu tf murni (*raw tf*). Dengan demikian rumus umum untuk TF-IDF adalah penggabungan dari formula perhitungan *raw tf* dengan formula *idf* (rumus 2.3) dengan cara mengalikan nilai *term frequency (tf)* dengan nilai *inverse document frequency (idf)* :

$$w_{ij} = tf_{ij} \times idf_j \qquad (4)$$

$$w_{ij} = tf_{ij} \times \log\left(\frac{D}{df_{ij}}\right) \qquad (5)$$

Keterangan :

w_{ij} adalah bobot *term* t_j terhadap *dokumen* d_i

tf_{ij} adalah jumlah kemunculan *term* t_j dalam dokumen d_i

D adalah jumlah semua dokumen yang ada dalam *database*

df_j adalah jumlah dokumen yang mengandung *term* t_j (minimal ada satu kata yaitu *term* t_j)

Berdasarkan rumus 2.4, berapapun besarnya nilai tf_{ij} , apabila D = df_j maka akan didapatkan hasil 0 (nol) untuk perhitungan *idf*. Untuk itu dapat ditambahkan nilai 1 pada sisi *idf*,' sehingga perhitungan bobotnya menjadi sebagai berikut:

$$w_{ij} = tf_{ij} \times (\log|\left(\frac{D}{df_j}\right) + 1) \qquad (6)$$

berikut ini diberikan contoh perhitungan bobot dokumen terhadap *query* yang diberikan pengguna, dengan menggunakan metode pembobotan TF-IDF di atas:

pengguna memberikan *query* : gold silver truck

sehingga didapatkan *query terms* (**Q**)*:*

- gold
- silver
- truck

dalam koleksi dokumen terdapat:

dokumen 1 (**d**1) = Shipment of gold damaged in a fire.

dokumen 2 (**d**2) = Delivery of silver arrived in a silver truck.

dokumen 3 (**d**3) = Shipment of gold arrived in a truck

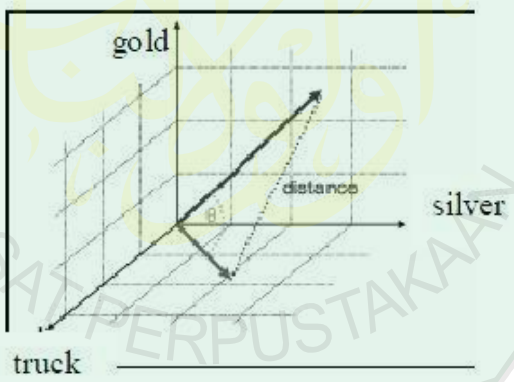
Jadi total jumlah dokumen dalam koleksi (**D**) = 3

Untuk setiap *query* dan dokumen dalam koleksi, dilakukan pemotongan string berdasarkan tiap kata yang menyusunnya, menghilangkan tanda baca, angka dan *stopword:*

Setelah melalui proses ini, maka kata *of, in***,** dan *a* pada ketiga dokumen dihapus lalu di-*stemming* sehingga didapatkan *term-term (documents terms)* sebagai berikut**:**

- ship
- damage
- deliver
- arrive
- gold
- fire
- silver
- truck

Pada tahap ini tiap dokumen diwujudkan sebagai sebuah vektor dengan elemen sebanyak *term query* yang terdapat dalam tiap dokumen yang berhasil dikenali dari tahap ekstraksi dokumen sebelumnya. Vektor tersebut beranggotakan bobot dari setiap *term query* yang dihitung berdasarkan metode TF-IDF



Gambar 2. 2 Representasi *Term Query* pada Ruang Vektor
(Sumber : Mandala, 2004)

Fungsi metode ini adalah untuk mencari representasi nilai dari tiap dokumen dalam koleksi. Dari sini akan dibentuk suatu vektor antara dokumen dan *query* yang ditentukan oleh nilai bobot *term query* dalam dokumen. Semakin besar nilai perhitungan bobot yang diperoleh maka semakin tinggi tingkat similaritas dokumen terhadap *query*. Contohnya untuk perhitungan bobot (w) *term*

query silver dalam dokumen2 (**d**2) = Delivery of silver arrived in a silver truck, yaitu: jumlah kemunculan *term* silver dalam dokumen 2 (**d**2) adalah sebanyak dua kali (tf = 2), total dokumen yang ada di koleksi sebanyak tiga dokumen (**D**=3), dari ketiga dokumen dalam koleksi, *term* silver muncul pada dokumen 2 (**d**2), sehingga total dokumen yang mengandung *term* silver adalah satu dokumen (df = 1), sehingga dapat diperoleh nilai bobot *term* silver pada dokumen 2 (**d**2)

$$w_{ij} = tf_{ij} \times (\log\left(\frac{D}{df_j}\right) + 1)$$

$$w_{ij} = 2 * (\log\left(\frac{3}{1}\right) + 1)$$

$$w_{ij} = 2 * (0.477 + 1)$$

$$w_{ij} = 2.954$$

Dengan demikian dapat diperoleh nilai bobot (w) untuk setiap *term* pada *query* dalam masing-masing dokumen:

Tabel 2. 1 Perhitungan Pembobotan TF-IDF *Term Query* dalam Setiap Dokumen (Sumber : Zafikri, 2010)

	tf			**df**	**D/df**	**IDF**	**IDF+1**	**W = tf* (IDF+1)**		
Q	**d1**	**d2**	**d3**					**d1**	**d2**	**d3**
gold	1	0	1	2	1.5	0.176	1.176	1.176	0	1.176
silver	0	2	0	1	3	0.477	0	0	2.954	0
truck	1	1	1	2	1.5	0.176	0	0	1.176	1.176
								Sum(d1)	Sum(d2)	Sum(d3)
				Nilai Bobot setiap Dokumen =				1.176	4.130	2.352

2.5 Bahasa Pemrograman Python

Python adalah bahasa pemrograman interpretatif multiguna dengan filosofi perancangan yang berfokus pada tingkat keterbacaan kode. Python diklaim sebagai bahasa yang menggabungkan kapabilitas, kemampuan, dengan sintaksis kode yang sangat jelas, dan dilengkapi dengan fungsionalitas pustaka standar yang besar serta komprehensif.

Python mendukung multi paradigma pemrograman, utamanya (tidak dibatasi) pada pemrograman berorientasi objek, pemrograman imperatif, dan pemrograman fungsional. Salah satu fitur yang tersedia pada python adalah sebagai bahasa pemrograman dinamis yang dilengkapi dengan manajemen memori otomatis. Seperti halnya pada bahasa pemrograman dinamis lainnya, python umumnya digunakan sebagai bahasa skrip meski pada praktiknya penggunaan bahasa ini lebih luas mencakup konteks pemanfaatan yang umumnya tidak dilakukan dengan menggunakan bahasa skrip. Python dapat digunakan untuk berbagai keperluan pengembangan perangkat lunak dan dapat berjalan di berbagai platform sistem operasi.

2.6 Twitter

Twitter adalah sebuah situs web yang dimiliki dan dioperasikan oleh Twitter Inc., yang menawarkan jaringan sosial berupa mikroblog sehingga memungkinkan penggunanya untuk mengirim dan membaca pesan *Tweets* (Twitter, 2013). Mikroblog adalah salah satu jenis alat komunikasi online dimana pengguna dapat memperbarui status tentang mereka yang sedang memikirkan dan

melakukan sesuatu, apa pendapat mereka tentang suatu objek atau fenomena tertentu. *Tweets* adalah teks tulisan hingga 140 karakter yang ditampilkan pada halaman profil pengguna. *Tweets* bisa dilihat secara publik, namun pengirim dapat membatasi pengiriman pesan ke daftar teman-teman mereka saja. Pengguna dapat melihat *Tweets* pengguna lain yang dikenal dengan sebutan pengikut (*follower).*

Tidak seperti Facebook, LinkedIn, dan MySpace, Twitter merupakan sebuah jejaring sosial yang dapat digambarkan sebagai sebuah graph berarah (Wang, 2010), yang berarti bahwa pengguna dapat mengikuti pengguna lain, namun pengguna kedua tidak diperlukan untuk mengikutinya kembali. Kebanyakan akun berstatus publik dan dapat diikuti tanpa memerlukan persetujuan pemilik.

Semua pengguna dapat mengirim dan menerima *Tweets* melalui situs Twitter, aplikasi eksternal yang kompatibel (telepon seluler), atau dengan pesan singkat (SMS) yang tersedia di negara-negara tertentu (Twitter, 2013). Pengguna dapat menulis pesan berdasarkan topik dengan menggunakan tanda # (*hashtag*). Sedangkan untuk menyebutkan atau membalas pesan dari pengguna lain bisa menggunakan tanda @.

Pesan pada awalnya diatur hanya mempunyai batasan sampai 140 karakter disesuaikan dengan kompatibilitas dengan pesan SMS, memperkenalkan singkatan notasi dan slang yang biasa digunakan dalam pesan SMS. Batas karakter 140 juga meningkatkan penggunaan layanan memperpendek URL seperti bit.ly, goo.gl, dan tr.im, dan jasa hosting konten, seperti Twitpic, Tweephoto, memozu.com dan NotePub untuk mengakomodasi multimedia isi dan teks yang

lebih panjang daripada 140 karakter (Twitter, 2013). Twitter menggunakan bit.ly untuk memperpendek otomatis semua URL yang dikirim-tampil. Fitur yang terdapat dalam Twitter, antara lain:

1. **Laman Utama (*Home)***

   Pada halaman utama kita bisa melihat *Tweets* yang dikirimkan oleh orang-orang yang menjadi teman kita atau yang kita ikuti (*following*).

2. **Profil (*Profile*)**

   Pada halaman ini yang akan dilihat oleh seluruh orang mengenai profil atau data diri serta *Tweets* yang sudah pernah kita buat.

3. ***Followers***

   Pengikut adalah pengguna lain yang ingin menjadikan kita sebagai teman. Bila pengguna lain menjadi pengikut akun seseorang, maka *Tweets* seseorang yang ia ikuti tersebut akan masuk ke dalam halaman utama.

4. ***Following***

   Kebalikan dari pengikut, *following* adalah akun seseorang yang mengikuti akun pengguna lain agar *Tweets* yang dikirim oleh orang yang diikuti tersebut masuk ke dalam halaman utama.

5. ***Mentions***

   Biasanya konten ini merupakan balasan dari percakapan agar sesama pengguna bisa langsung menandai orang yang akan diajak bicara.

6. ***Favorite***

   Tweets ditandai sebagai favorit agar tidak hilang oleh halaman sebelumnya.

7. **Pesan Langsung (*Direct Message*)**

   Fungsi pesan langsung lebih bisa disebut SMS karena pengiriman pesan langsung di antara pengguna.

8. ***Hashtag***

   Hashtag "#" yang ditulis di depan topik tertentu agar pengguna lain bisa mencari topik yang sejenis yang ditulis oleh orang lain juga

9. ***List***

   Pengguna Twitter dapat mengelompokkan ikutan mereka ke dalam satu grup sehingga memudahkan untuk dapat melihat secara keseluruhan para nama pengguna (*username*) yang mereka ikuti (*follow*).

10. **Topik Terkini (*Trending Topic*)**

    Topik yang sedang banyak dibicarakan banyak pengguna dalam suatu waktu yang bersamaan.

2.7 Gempa Bumi di Indonesia

Gempa bumi adalah peristiwa bergetarnya bumi akibat pelepasan energi di dalam bumi secara tiba-tiba yang ditandai dengan patahnya lapisan batuan pada kerak bumi. Akumulasi energi penyebab terjadinya gempabumi dihasilkan dari pergerakan lempeng-lempeng tektonik. Energi yang dihasilkan dipancarkan kesegala arah berupa gelombang gempabumi sehingga efeknya dapat dirasakan sampai ke permukaan bumi.

Menurut teori lempeng tektonik, permukaan bumi terpecah menjadi beberapa lempeng tektonik besar. Lempeng tektonik adalah segmen keras kerak

bumi yang mengapung diatas astenosfer yang cair dan panas. Oleh karena itu, maka lempeng tektonik ini bebas untuk bergerak dan saling berinteraksi satu sama lain. Daerah perbatasan lempeng-lempeng tektonik, merupakan tempat-tempat yang memiliki kondisi tektonik yang aktif, yang menyebabkan gempa bumi, gunung berapi dan pembentukan dataran tinggi. Teori lempeng tektonik merupakan kombinasi dari teori sebelumnya yaitu: Teori Pergerakan Benua (Continental Drift) dan Pemekaran Dasar Samudra (Sea Floor Spreading).

Lapisan paling atas bumi, yaitu litosfir, merupakan batuan yang relatif dingin dan bagian paling atas berada pada kondisi padat dan kaku. Di bawah lapisan ini terdapat batuan yang jauh lebih panas yang disebut mantel. Lapisan ini sedemikian panasnya sehingga senantiasa dalam keadaan tidak kaku, sehingga dapat bergerak sesuai dengan proses pendistribusian panas yang kita kenal sebagai aliran konveksi. Lempeng tektonik yang merupakan bagian dari litosfir padat dan terapung di atas mantel ikut bergerak satu sama lainnya. Ada tiga kemungkinan pergerakan satu lempeng tektonik relatif terhadap lempeng lainnya, yaitu apabila kedua lempeng saling menjauhi (spreading), saling mendekati(collision) dan saling geser (transform).

Jika dua lempeng bertemu pada suatu sesar, keduanya dapat bergerak saling menjauhi, saling mendekati atau saling bergeser. Umumnya, gerakan ini berlangsung lambat dan tidak dapat dirasakan oleh manusia namun terukur sebesar 0-15cm pertahun. Kadang-kadang, gerakan lempeng ini macet dan saling mengunci, sehingga terjadi pengumpulan energi yang berlangsung terus sampai pada suatu saat batuan pada lempeng tektonik tersebut tidak lagi kuat menahan

gerakan tersebut sehingga terjadi pelepasan mendadak yang kita kenal sebagai gempa bumi.

Indonesia merupakan daerah rawan gempabumi karena dilalui oleh jalur pertemuan 3 lempeng tektonik, yaitu: Lempeng Indo-Australia, lempeng Eurasia, dan lempeng Pasifik.

Lempeng Indo-Australia bergerak relatip ke arah utara dan menyusup kedalam lempeng Eurasia, sementara lempeng Pasifik bergerak relatip ke arah barat.

Jalur pertemuan lempeng berada di laut sehingga apabila terjadi gempabumi besar dengan kedalaman dangkal maka akan berpotensi menimbulkan tsunami sehingga Indonesia juga rawan tsunami.

Belajar dari pengalaman kejadian gempabumi dan tsunami di Aceh, Pangandaran dan daerah lainnya yang telah mengakibatkan korban ratusan ribu jiwa serta kerugian harta benda yang tidak sedikit, maka sangat diperlukan upaya-upaya mitigasi baik ditingkat pemerintah maupun masyarakat untuk mengurangi resiko akibat bencana gempabumi dan tsunami.

Mengingat terdapat selang waktu antara terjadinya gempa bumi dengan tsunami maka selang waktu tersebut dapat digunakan untuk memberikan peringatan dini kepada masyarakat sebagai salah satu upaya mitigasi bencana tsunami dengan membangun Sistem Peringatan Dini Tsunami Indonesia (Indonesia Tsunami Early Warning System / Ina-TEWS).

BAB III

ANALISIS DAN PERANCANGAN SISTEM

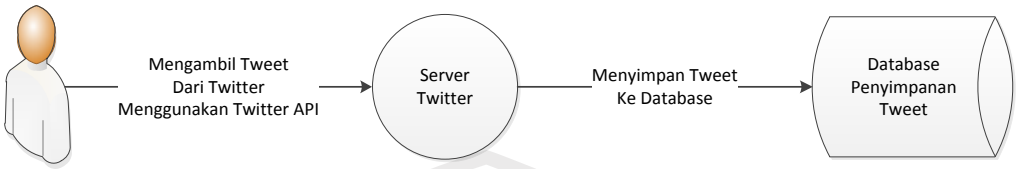
Pada bab ini akan dijelaskan tentang analisa dan perancangan sistem dari aplikasi Twitter *text mining* untuk informasi gempa bumi menggunakan tf-idf di Indonesia. Perancangan dan analisis sistem dilakukan berdasarkan kebutuhan dengan tujuan untuk memperoleh informasi mengenai kejadian gempa bumi di Indonesia.

3.1 Analisis Data

3.1.1 Data *Tweet*

Dalam penelitian ini, data yang digunakan adalah data *tweet* dari pengguna twitter. Data *Tweet* dperoleh dengan memanfaatkan API yang disediakan oleh Twitter. Dengan memanfaatkan API tersebut dibangunlah sebuah aplikasi untuk mengambil data *Tweet* tersebut dari Twitter kemudian disimpan ke dalam Database.

Pada saat pengumpulan data, peneliti menggunakan Twitter API *Search*, kemudian memasukkan keyword-keyword yang berhubungan dengan gempa bumi dan nama daerah di seluruh Indonesia. Proses pengambilan data *tweet* dimulai dari pengguna mengambil data *tweet* sesuai dengan kata kunci dari Twitter menggunakan Twitter API Search. Kemudian server Twitter akan memberikan *tweet* sesuai dengan kata kunci selanjutnya data *tweet* dapat diunduh dan disimpan ke dalam data base. Skema dari proses pengambilan data *tweet* ini seperti pada gambar 3.1



Gambar 3. 1 Skema dari proses pengambilan data *tweet*

3.1.2 Data Stopword

Data *stopword* didapat dari jurnal Tala (2003) dimana datanya berjumlah 753 data dan dari *Tweet-tweet* yang digunakan dalam penelitian. Data *stopword* di dalam *database*. Data *stopword* digunakan saat *script* stopwords dijalankan. Rancangan tabel *stopword* dapat dilihat pada Tabel 3.1.

Tabel 3. 1 Tabel Stopword

Id_stopword	Stopword
1	ada
2	adanya
3	adalah
4	adapun
5	agak
6	agaknya
7	agar
8	akan
9	akankah

3.1.3 Data Keyword

Data keyword didapat dari kata “gempa”, “gempa bumi”, dan nama daerah Kabupaten/kota di seluruh Indonesia. Data keyword disimpan di dalam dokumen berekstensi .txt, dokumen ini akan di-*load* saat *script* twitter streaming API dijalankan. Rancangan data keyword ditulis pada dokumen teks berekstensi .txt dapat dilihat pada gambar 3.2.

gempa bumi jawa timur bangkalan banyuwangi blitar bojonegoro bondowoso gresik jember jombang kediri lamongan lumajang madiun magetan malang mojokerto ngnanjuk ngawi pacitan pamekasan pasuruan ponorogo probolinggo sampang sidoarjo situbondo sumenep trenggalek tuban tulungagung batu Surabaya jawa tengah banjarnegara banyumas batang blora boyolali brebes cilacap demak grobogan jepara karanganyar kebumen kendal klaten kudus magelang pati pekalongan pemalang purbalingga purworejo rembang semarang sragen tegal temanggung wonogiri wonosobo magelang salatiga surakarta

Gambar 3. 2 Data Keyword

3.2 Analisis Sistem

Analisis sistem bertujuan untuk mengidentifikasi permasalahan-permasalahan yang ada pada sistem yang meliputi perangkat lunak (*software*), pengguna (*user*) serta hasil analisis terhadap sistem dan elemen-elemen yang terkait. Analisis ini diperlukan sebagai dasar bagi tahapan perancangan sistem.

3.2.1 Stemming

Berdasarkan algoritma *confix stripping* langkah-langkah proses *stemming* adalah sebagai berikut :

1. Kata yang belum di-*stemming* dibandingkan ke dalam *database* kamus kata dasar. Jika ditemukan, maka kata tersebut diasumsikan sebagai kata

dasar dan algoritma berhenti. Jika kata tidak sesuai dengan kata dalam kamus, lanjut ke langkah 2.

2. Jika kata di-*input* memiliki pasangan awalan-akhiran “be-lah”,”be-an”,”me-i”,”di-i”,”pe-i”, atau “te-i” maka langkah *stemming* selanjutnya adalah 5, 3, 4, 5, 6, tetapi jika kata yang di-*input* tidak memiliki pasangan awalan-akhiran tersebut, langkah *stemming* berjalan normal yaitu 3, 4, 5, 6, 7.
3. Hilangkan partikel dan kata ganti kepunyaan. Pertama hilangkan partikel (“-lah”, “-kah”, “-tah”, “-pun”). Setelah itu hilangkan juga kata ganti kepunyaan (“-ku”, “-mu”, atau “-nya”). Contoh : kata “bajumulah”, proses *stemming* pertama menjadi “bajumu” dan proses stemming kedua menjadi “baju”. Jika kata “baju” ada di dalam kamus maka algoritma berhenti. Sesuai dengan model imbuhan, menjadi :

   [[[AW+]AW+]AW+] Kata Dasar [+AK]

4. Hilangkan juga akhiran (“-i”,”-an”, dan “-kan”), sesuai dengan model imbuhan, maka menjadi:

   [[[AW+]AW+]AW+] Kata Dasar

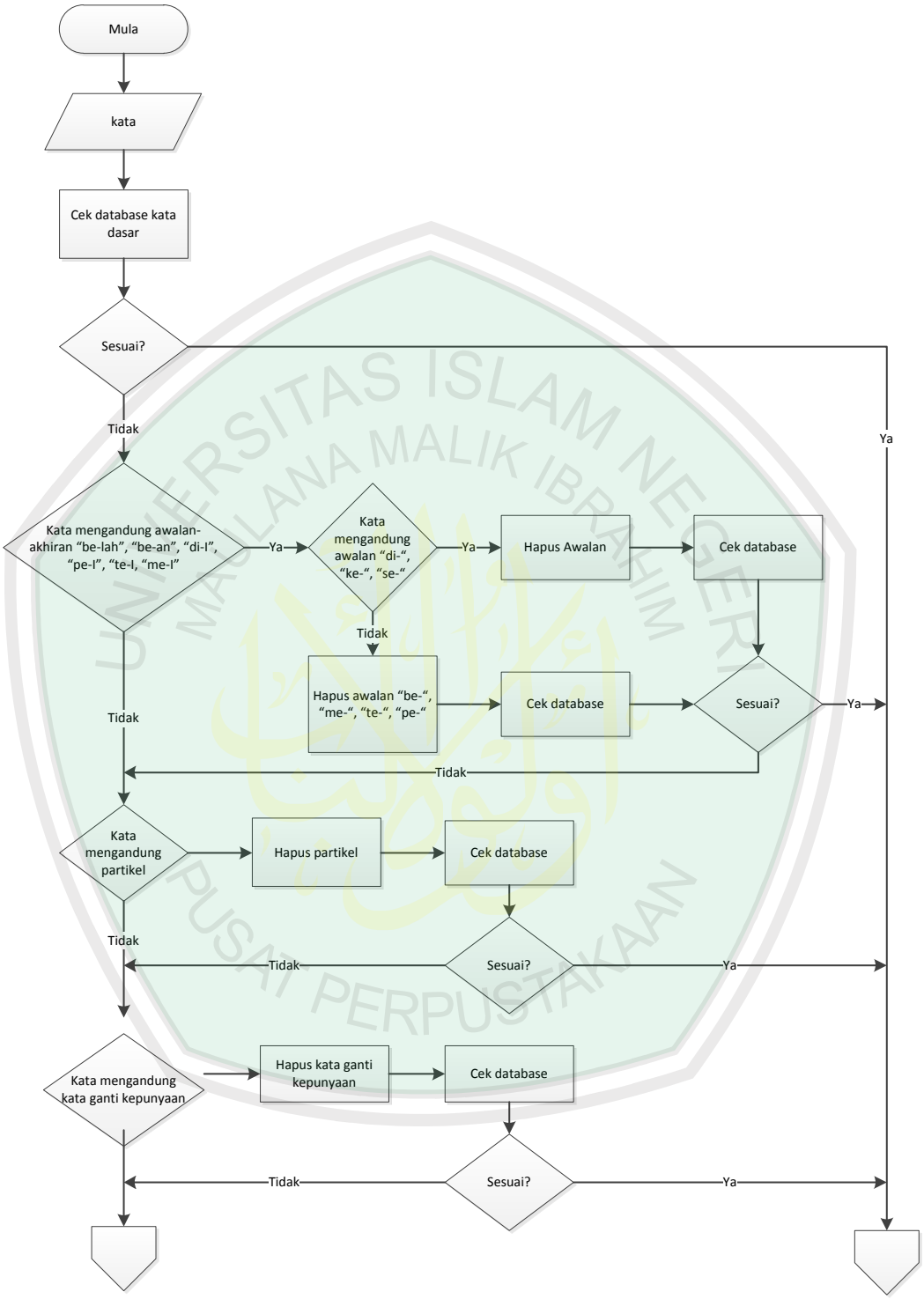
   Contoh: kata “membelikan” di-*stemming* menjadi ”membeli”, jika tidak ada dalam *database* kata dasar maka dilakukan proses penghilangan awalan.
5. Penghilangan awalan (“be-“, ”di-“, ”ke-“, ”me-”, ”pe-”, ”se-”, dan “te-“) mengikuti langkah-langkah berikut:

   a. Algoritma akan berhenti jika:

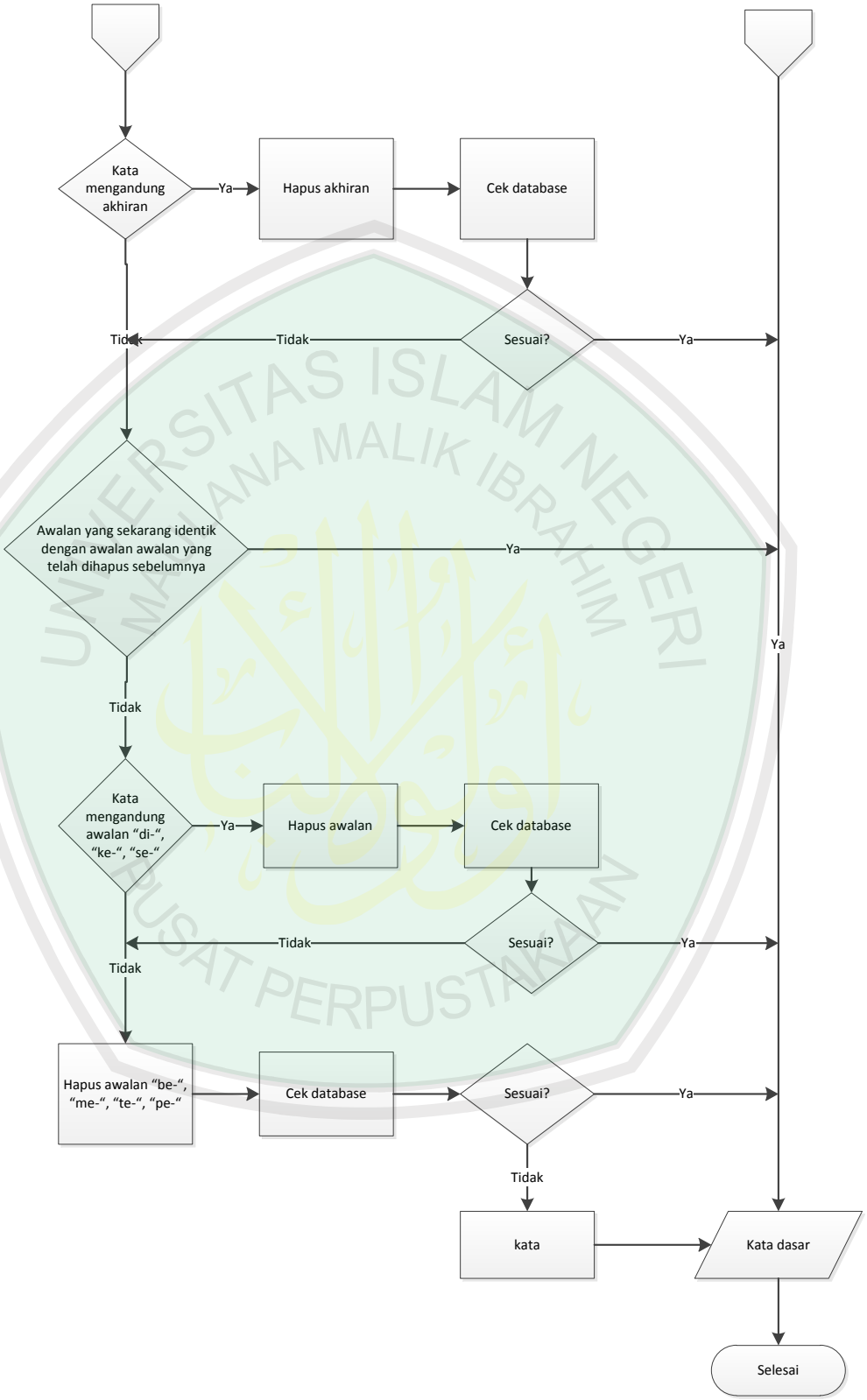
i. Awalan diidentifikasi bentuk sepasang imbuhan yang tidak diperbolehkan dengan akhiran (berdasarkan tabel 2.1) yang dihapus pada langkah 3.
ii. Diidentifikasi awalan yang sekarang identik dengan awalan yang telah dihapus sebelumnya atau,
iii. Kata tersebut sudah tidak memiliki awalan.

b. Identifikasi jenis awalan dan peluruhannya bila diperlukan.jenis awalan ditentukan dengan aturan dibawah ini.
i. Jika awalan dari kata adalah “di-“, “ke-“, atau “se-“ maka awalan dapat langsung dihilangkan.
ii. Hapus awalan “te-“, “be-“, “me-“, atau “pe-“ yang menggunakan aturan peluruhan. Sebagai contoh kata “menangkap”, setelah menghilangkan awalan “me-“ maka kata yang didapat adalah “nangkap”. Karena kata “nangkap” tidak ditemukan dalam database kata dasar maka karakter “n” diganti dengan karakter “t” sehingga dihasilkan kata “tangkap” dan kata “tangkap” merupakan kata yang sesuai dengan kata yang ada di database kata dasar, maka algoritma berhenti.

6. Jika semua langkah gagal, maka kata yang diuji pada algoritma ini dianggap sebagai kata dasar. *Flowchart* dari proses *stemming* adalah seperti pada Gambar 3.3 :



Gambar 3. 3 *Flowchart* proses *stemming* (Manulu, 2014)



Gambar 3.3 *Flowchart* proses *stemming* (lanjutan) (Manulu, 2014)

3.2.2 Contoh Penggunaan TF-IDF

3.2.2.1 Menghitung Term Frequency (TF)

Term frequency (tf) merupakan frekuensi kemunculan *term* (t) pada dokumen (d). berikut ini merupakan sebuah contoh yang terdapat pada paragraf:

> Saya sedang belajar menghitung tf-idf. Tf-idf merupakan frekuensi kemunculan term pada dokumen. Langkah awal perhitungan tersebut adalah menghitung tf, kemudian menghitung df dan idf. Langkah terakhir menghitung nilai tf-idf. Mari kita belajar!

Gambar 3. 4 Contoh Kalimat

Tiap kalimat dianggap sebagai dokumen. Untuk menentukan nilai tf, setiap kalimat pada tiap dokumen ditandai :

"Saya sedang belajar menghitung tf-idf."

"Tf-idf merupakan frekuensi kemunculan term pada dokumen."

"Langkah awal perhitungan tersebut adalah menghitung tf, kemudian menghitung df dan idf."

"Langkah terakhir menghitung nilai tf-idf."

"Mari kita belajar!"

Tabel 3. 2 Term Frequency

Term (t)	D1 (dokumen 1)	D2	D3	D4	D5
Akhir	0	0	0	1	0
Awal	0	0	1	0	0
Belajar	1	0	0	0	1
Dokumen	0	1	0	0	0
Frekuensi	0	1	0	0	0
Hitung	1	0	3	1	0
Idf	1	1	1	1	0

Term (t)	D1 (dokumen 1)	D2	D3	D4	D5
Kita	0	0	0	0	1
Langkah	0	0	1	1	0
Muncul	0	1	0	0	0
Saya	1	0	0	0	0
Term	0	1	0	0	0
Tf	1	1	1	1	0
df	0	0	1	0	0

3.2.2.2 Menghitung Document Frequency (DF)

Document frequency (df) adalah banyaknya dokumen dimana suatu *term* (t) muncul. Soal yang sama dengan sebelumnya pada sub bab 3.2.2.1 contoh menghitung tf, untuk menentukan nilai df dapat seperti pada Tabel 3.3

Tabel 3. 3 Tabel Document Frequency

Term (t)	df
Akhir	1
Awal	1
Belajar	2
Dokumen	1
Frekuensi	1
Hitung	3
Idf	4
Kita	1
Langkah	2
Muncul	1
Saya	1
Term	1
Tf	4

Term (t)	df
Df	1

3.2.2.3 Menghitung Inverse Document Frequency (IDF)

Berdasarkan perhitungan sebelumnya pada sub bab 3.2.2.2 tentang penghitungan tf dan df, untuk mendapatkan nilai idf dapat menggunakan persamaan :

$$idf = \log(\frac{N}{df})$$

Tabel 3. 4 Tabel IDF

Term (t)	df	idf
Akhir	1	1
Awal	1	1
Belajar	2	½=0.5
Dokumen	1	1
Frekuensi	1	1
Hitung	3	1/3=0.3
Idf	4	¼=0.25
Kita	1	1
Langkah	2	½=0.5
Muncul	1	1
Saya	1	1
Term	1	1
Tf	4	¼=0.25
Df	1	1

Apabilai nilai yang diasumsikan N=1000 maka akan didapatkan nilai idf seperti pada Tabel 3.5

Tabel 3. 5 IDF apabila N=1000

Term (t)	df	idf
Akhir	1	log(1000/1)=3
Awal	1	log(1000/1)=3
Belajar	2	log(1000/2)=2.70
Dokumen	1	log(1000/1)=0.602
Frekuensi	1	log(1000/1)=0.602
Hitung	3	log(1000/3)=2.52
Idf	4	log(1000/4)=2.40
Kita	1	log(1000/1)=0.602
Langkah	2	log(1000/2)= 2.70
Muncul	1	log(1000/1)=0.602
Saya	1	log(1000/1)=0.602
Term	1	log(1000/1)=0.602
Tf	4	log(1000/4)=2.40
Df	1	log(1000/1)=0.602

3.2.2.4 Menghitung TF-IDF

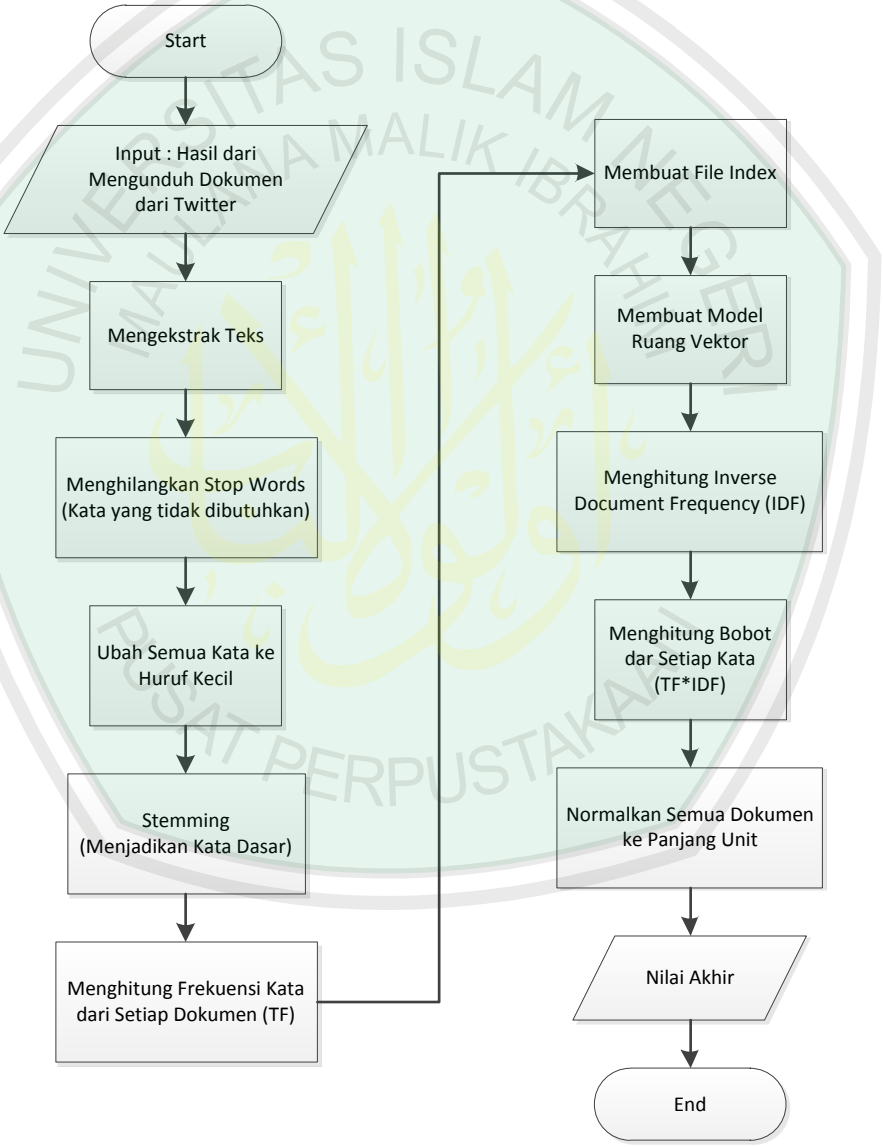
Untuk mendapatkan nilai TF-IDF dapat mengalikan nilai tf dengan idf. Dengan doal yang sama dengan penghitungan sebelumnya, penghitungan TF-IDF dapat menghasilkan nilai seperti pada Tabel 3.6

Tabel 3. 6 Penghitungan nilai TF-IDF

Term (t)	D1 (dokumen 1)	D2	D3	D4	D5	idf	TF*IDF				
							D1	D2	D3	D4	D5
Akhir	0	0	0	1	0	log(4/1)=0.602	0	0	0	0.602	0
Awal	0	0	1	0	0	log(4/1)=0.602	0	0	0.602	0	0
Belajar	1	0	0	0	1	log(4/2)=0.301	0.301	0	0	0	0.301
Dokumen	0	1	0	0	0	log(4/1)=0.602	0	0.602	0	0	0
Frekuensi	0	1	0	0	0	log(4/1)=0.602	0	0.602	0	0	0
Hitung	1	0	3	1	0	log(4/3)=0.125	0.125	0	0.375	0.125	0
Idf	1	1	1	1	0	log(4/4)=0	0	0	0	0	0
Kita	0	0	0	0	1	log(4/1)=0.602	0	0	0	0	0.602
Langkah	0	0	1	1	0	log(4/2)=0.301	0	0	0.301	0.301	0
Muncul	0	1	0	0	0	log(4/1)=0.602	0	0.602	0	0	0
Saya	1	0	0	0	0	log(4/1)=0.602	0.602	0	0	0	0
Term	0	1	0	0	0	log(4/1)=0.602	0	0.602	0	0	0
Tf	1	1	1	1	0	log(4/4)=0	0	0	0	0	0
Df	0	0	1	0	0	log(4/1)=0.602	0	0	0.602	0	0

3.2.2.5 Flowchart TF-IDF

Seluruh proses penghitungan nilai TF-IDF dapat digambarkan dalam flowchart pada Gambar 3.5 proses pertama adalah start/mulai. Kemudian data input yang didapatkan dari mengunduh dokumen stream dari Twitter. Dokumen ini kemudian dilakukan proses ekstrak teks.



Gambar 3. 5 Flowchart TF-IDF

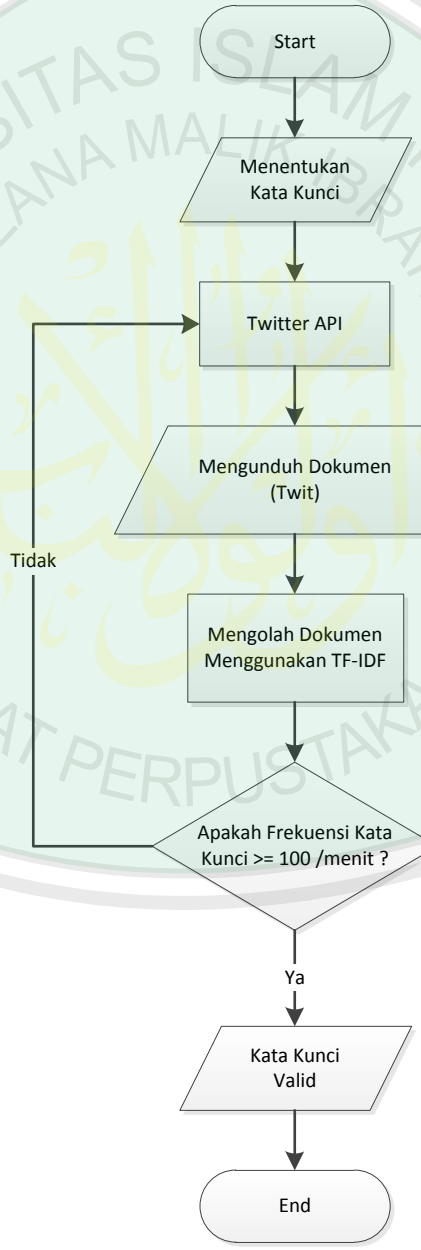
Kata yang tidak dibutuhkan akan dihilangkan pada saat tahap stopwords kemudian seluruh huruf pada setiap kata dirubah menjadi huruf kecil. Proses berikutnya adalah stemming, yakni merubah kata yang memiliki awalan atau akhiran menjadi kata dasar. Setelah stemming selesai dilakukan, penghitungan frekuensi kata dari setiap dokumen (TF) bisa dimulai. Hasil dari penghitungan frekuensi kata ini dubuatkan file index. Dari file index dibuatkan model ruang vektor. Proses selanjutnya adalah menghitung Inverse Document Frequency (IDF). Setelah melakukan penghitungan IDF dapat dilanjutkan ke proses selanjutnya yakni menghitung bobot dari setiap kata (TF-IDF). Apabila sudah didapatkan nilai akhir. Seluruh proses TF-IDF selesai.

3.3 Perancangan Sistem

3.3.1 Flowchart Sistem

Alur kerja sistem Twitter *text mining* untuk informasi gempa bumi menggunakan TF-IDF secara umum ditunjukkan gambar 3.4 berikut. Penjelasan dari alur tersebut yang pertama adalah memulai kemudian menentukan kata kunci, yakni “gempa”. Kemudian aplikasi berjalan dan meminta kepada Twitter API untuk dicarikan seluruh *tweet* yang terdapat kata “gempa” di dalamnya. Seluruh *tweet* yang terdapat kata “gempa” di-*stream* dan dikumpulkan untuk kemudian diunduh dan disimpan ke dalam dokumen. Dokumen yang baru didapatkan inilah yang akan diolah menggunakan TF-IDF untuk mendapatkan bobot *term.* Setelah bobot *term* didapatkan, proses selanjutnya adalah apakah frekuensi kata kunci lebih dari atau sama dengan 100 *tweet* per menit. Apabila didapatkan 100 *tweet*

dalam satu menit, maka dapat diinformasikan bahwa sedang terjadi gempa bumi di suatu daerah di Indonesia dan dapat dirasakan oleh banyak orang. Apabila tidak mencapai 100 *tweet* dalam satu menit yang terjadi sebaliknya. Proses ini akan terus berjalan berulang untuk terus mendapatkan informasi dari *tweet* tentang gempa.



Gambar 3. 6 Flowchart Sistem

3.3.2 Definisi Flowchart Sistem

Flowchart merupakan alur kerja yang dibuat untuk memudahkan dalam memahami desain sistem yang akan dibuat. Definisi dari flowchart dapat dilihat pada Tabel 3.7

Tabel 3. 7 Definisi Flowchart Sistem

Chart	Deskripsi
Menentukan kata kunci	Kata kunci yang digunakan adalah kata "gempa" dan nama daerah Kabupaten / kota di Indonesia
Twitter API	Sistem akan mencari *tweet* yang sesuai dengan kata kunci memanfaatkan API yang disediakan oleh Twitter
Mengunduh dokumen (*tweet*)	*Tweet* yang sesuai dengan kata kunci akan diunduh dan dikumpulkan dalam dokumen
Mengolah dokumen menggunakan TF-IDF	Dokumen berisikan *tweet* dengan kata kunci yang telah ditentukan diolah menggunakan TF-IDF untuk mendapatkan nilai frekuensi kata
Apakah frekuensi kata kunci >= 100 / menit?	Apabila frekuensi kata yang muncul lebih dari atau sama dengan 100 kata per menit, maka kata kunci valid, jika tidak maka sistem akan mengunduh *tweet* lagi dari Twitter API
Kata kunci valid	Kata kunci valid maka dapat disimpulkan telah terjadi gempa bumi di suatu daerah di Indonesia.

3.4 Perancangan Tampilan Antarmuka

Perancangan antarmuka digunakan untuk menggambarkan tampilan antarmuka. Pada sistem twitter *text mining* untuk informasi gempa bumi menggunakan TF-IDF di Indonesia ini nantinya akan menampilkan sebuah grafik yang menggambarkan frekuensi *tweet* yang dituliskan oleh pengguna setiap menit. Gambar 3.7 ini adalah rancangan antarmuka dari aplikasi yang direncanakan.

Grafik ini nantinya akan menjelaskan berapa banyak *tweet* atau frekuensi *tweet* pada tiap menitnya. Sumbu y adalah frekuensi, sedangkan sumbu x adalah waktu.

Gambar 3. 7 Desain antarmuka

BAB IV

HASIL DAN PEMBAHASAN

Pada bab ini pembahasan mengenai hasil uji coba Twitter *text mining* untuk informasi gempa bumi menggunakan TF-IDF di Indonesia yang telah dirancang dan dibuat. Uji coba dilakukan untuk mengetahui apakah sistemnya dapat berjalan sebagaimana mestinya dengan lingkungan uji coba yang telah dilakukan.

4.1 Implementasi Sistem

Uji coba sistem ini melalui beberapa tahap, yang pertama adalah implementasi stream twitter API. Pada tahap awal ini sistem akan melakukan stream terhadap web twitter dengan kata kunci "gempa" sebagai penyaring. Kemudian hasil dari stream yang didapatkan akan diolah ke tahap selanjutnya, yakni tokenisasi dan *stopword* sebelum akhirnya sampai pada tahap Term Frequency Inverse Document Frequency (TF-IDF).

Hasil dari implementasi tahap TF-IDF akan divisualisasikan ke dalam bentuk grafik *chart time series* agar dapat dengan mudah memahami informasi yang disampaikan. Pada grafik ini akan menampilkan frekuensi dari *term* "gempa" dalam tiap menit. Selain itu, dari implementasi ini dapat diketahui seberapa akurat data yang dihasilkan agar dapat mengetahui apakah data yang dighasilkan layak untuk dijadikan sebagai informasi terjadinya gempa bumi.

4.1.1 Implementasi Stream Twitter API

Sistem ini mengandalkan twitter search API sebagai sumber data text yang akan diolah. Untuk mendapatkan *tweet,* peneliti menggunakan aplikasi stream twitter API dengan bahasa pemrograman Python. Cara kerja aplikasi ini adalah dengan menjalankan source code di terminal. Dibutuhkan ckey dan atoken sebagai autentifikasi untuk dapat mengakses stream twitter yang disediakan oleh API search. Dalam source code ini juga dituliskan kata kunci "gempa" kemudian melakaukan pengunduhan *tweet* dengan kata kunci "gempa". Seluruh *tweet* yang telah diunduh disimpan dala dokumen .json untuk dilakukan proses berikutnya. Berikut ini adalah source code implementasi stream twitter API pada twitter *text mining* untuk informasi gempa bumi.

Langkah pertama yang dilakuakn dalam tahap ini adalah aplikasi akan berkoneksi dan berhubungan dengan Twitter API Search. Agar dapat mengakses dibutuhkan ckey dan atoken sebagai autentifikasi untuk dapat mengakses stream twitter yang disediakan oleh API search. Kode hak akses atau ckey dan atoken dituliskan pada *script* Twitter streaming. Adapun isi dari *script* tersebut sebagai berikut:

```
ckey = 'wmSMYOFxQ867iGfvoaEBdAOpg'
csecret =
'dvJTua6me7cXVjUgyneWru6tiqX70wtPNMWmZlXGyOUeGa691W'
atoken = '61111680-
N5KCNpLzSz5f7bhOnZeXtVzTmFOdnWlvBq5o09GT0'
asecret = 'snYPccETaazso21XowfusBZKxcz8uQ2mxrjgoHYb8BIFv'
```

Source Code 4. 1 Pengaturan Hak Akses API Twitter

Seluruh kode hak akses didapatkan langsung dari aplikasi twitter. Untuk mendapatkan hak akses ini harus terdaftar sebagai pengguna Twitter kemudian mengajukan permintaan hak akses dengan mengisi form pembuatan aplikasi.

Setelah pengajuan pembuatan aplikasi disetujui, Twitter akan memberikan hak akses berupa kode.

```
twitter_stream = Stream(auth, MyListener(args.data_dir,
args.query))
    twitter_stream.filter(track=["gempa", "#gempa"])
```

Source Code 4. 2 Filter Gempa

Untuk mendapatkan *tweet* sesuai dengan kata kunci yang dibutuhkan. Pada akhir *script* ditambahkan filter atau penyaring berisikan kata kunci yakni "gempa".

```
import tweepy
from tweepy import Stream
from tweepy import OAuthHandler
from tweepy.streaming import StreamListener
import time
import argparse
import string
import json

ckey = 'wmSMYOFxQ867iGfvoaEBdAOpg'
csecret =
'dvJTua6me7cXVjUgyneWru6tiqX70wtPNMWmZlXGyOUeGa691W'
atoken = '61111680-
N5KCNpLzSz5f7bhOnZeXtVzTmFOdnWlvBq5o09GT0'
asecret = 'snYPccETaazso21XowfusBZKxcz8uQ2mxrjgoHYb8BIFv'

def get_parser():
    """Get parser for command line arguments."""
    parser = argparse.ArgumentParser(description="Twitter
Downloader")
    parser.add_argument("-q",
                        "--query",
                        dest="query",
                        help="Query/Filter",
                        default='-')
    parser.add_argument("-d",
                        "--data-dir",
                        dest="data_dir",
                        help="Output/Data Directory")
    return parser

#class listener (StreamListener):

class MyListener(StreamListener):
```

```
    """Custom StreamListener for streaming data."""

    def __init__(self, data_dir, query):
        query_fname = format_filename(query)
        self.outfile = "%s/stream_%s.json" % (data_dir,
query_fname)

    def on_data(self, data):
        try:
            with open(self.outfile, 'a') as f:
                f.write(data)
                print(data)
                return True
        except BaseException as e:
            print("Error on_data: %s" % str(e))
            time.sleep(5)
        return True

    def on_error(self, status):
        print(status)
        return True

def format_filename(fname):
    """Convert file name into a safe string.

    Arguments:
        fname -- the file name to convert
    Return:
        String -- converted file name
    """
    return ''.join(convert_valid(one_char) for one_char in
fname)

def convert_valid(one_char):
    """Convert a character into '_' if invalid.

    Arguments:
        one_char -- the char to convert
    Return:
        Character -- converted char
    """
    valid_chars = "-_.%s%s" % (string.ascii_letters,
string.digits)
    if one_char in valid_chars:
        return one_char
    else:
        return '_'

@classmethod
def parse(cls, api, raw):
    status = cls.first_parse(api, raw)
```

```
    setattr(status, 'json', json.dumps(raw))
    return status

if __name__ == '__main__':
    parser = get_parser()
    args = parser.parse_args()
    auth = OAuthHandler(ckey, csecret)
    auth.set_access_token(atoken, asecret)
    api = tweepy.API(auth)

    twitter_stream = Stream(auth, MyListener(args.data_dir,
args.query))
    twitter_stream.filter(track=["gempa", "#gempa"])
```

Source Code 4. 3 Streaming Twitter API

Dari *source code* stream twitter API yang dijalankan, didapatkan informasi lengkap dari twitter berupa teks antara lain berisikan informasi mengenai *text* yakni tentang *tweet* itu sendiri, *crerated_at* merupakan tanggal *tweet* ditulis, *favourite_count* dan *retweet_count* adalah jumlah favorit dan re*tweet, lang* adalah bahasa yang digunakan, *id* identitas *tweet*, *place coordinates geo* merupakan lokasi dari mana *tweet* ditulis apabila tersedia, *user* merupakan profil lengkap penulis *tweet* atau pengguna twitter. Seluruh teks disimpan dalam file .json.


```
{
    "contributors": null,
    "truncated": false,
    "text": "Gempa Super Dahsyat Prediksi Akan Terjadi
https://t.co/LybhSrk3N0",
    "is_quote_status": false,
    "in_reply_to_status_id": null,
    "id": 744189258911481856,
    "favorite_count": 0,
    "source": "<a href=\"http://www.facebook.com/twitter\"
rel=\"nofollow\">Facebook</a>",
    "retweeted": false,
    "coordinates": null,
    "timestamp_ms": "1466263520349",
    "entities": {
        "user_mentions": [],
        "symbols": [],
        "hashtags": [],
        "urls": [
            {
                "url": "https://t.co/LybhSrk3N0",
```

```
                "indices": [
                    42,
                    65
                ],
                "expanded_url": "http://fb.me/18eJuWC93",
                "display_url": "fb.me/18eJuWC93"
            }
        ]
    },
    "in_reply_to_screen_name": null,
    "id_str": "744189258911481856",
    "retweet_count": 0,
    "in_reply_to_user_id": null,
    "favorited": false,
    "user": {
        "follow_request_sent": null,
        "profile_use_background_image": false,
        "default_profile_image": false,
        "id": 725777857629147136,
        "verified": false,
        "profile_image_url_https":
"https://pbs.twimg.com/profile_images/725778801351757824/9G
S90kwp_normal.jpg",
        "profile_sidebar_fill_color": "000000",
        "profile_text_color": "000000",
        "followers_count": 4,
        "profile_sidebar_border_color": "000000",
        "id_str": "725777857629147136",
        "profile_background_color": "000000",
        "listed_count": 0,
        "profile_background_image_url_https":
"https://abs.twimg.com/images/themes/theme1/bg.png",
        "utc_offset": null,
        "statuses_count": 1167,
        "description": null,
        "friends_count": 1,
        "location": null,
        "profile_link_color": "19CF86",
        "profile_image_url":
"http://pbs.twimg.com/profile_images/725778801351757824/9GS
90kwp_normal.jpg",
        "following": null,
        "geo_enabled": false,
        "profile_banner_url":
"https://pbs.twimg.com/profile_banners/725777857629147136/1
461874171",
        "profile_background_image_url":
"http://abs.twimg.com/images/themes/theme1/bg.png",
        "name": "Cantika Indriyani",
        "lang": "id",
        "profile_background_tile": false,
```

```
        "favourites_count": 0,
        "screen_name": "CantikaIndriyan",
        "notifications": null,
        "url": null,
        "created_at": "Thu Apr 28 20:05:00 +0000 2016",
        "contributors_enabled": false,
        "time_zone": null,
        "protected": false,
        "default_profile": false,
        "is_translator": false
    },
    "geo": null,
    "in_reply_to_user_id_str": null,
    "possibly_sensitive": false,
    "lang": "in",
    "created_at": "Sat Jun 18 15:25:20 +0000 2016",
    "filter_level": "low",
    "in_reply_to_status_id_str": null,
    "place": null
}
```


Source Code 4. 4 Hasil Streaming Twitter

Teks Source Code 4.4 berisikan seluruh informasi yang berkaitan dengan *tweet*. Dalam penelitian *text mining* ini. Data yang akan diambil adalah data *tweet* teks dari pengguna.

4.1.2 Implementasi Tokenization

Tokenization digunakan untuk memilah bagian mana saja yang dibutuhkan untuk diolah selanjutnya karena data yang dihasilkan dari streaming twitter sangat banyak dan tidak semuanya dibutuhkan dalam penelitian ini. Dalam penelitian ini, bagian yang dibutuhkan adalah *tweet* text dari pengguna, sehingga bagian yang akan diambil hanya tweet yang dituliskan dan disampaikan oleh pengguna pada akun media sosial twitter mereka. Pada source code 4.5 merupakan baris perintah untuk melakukan tokenisasi dalam Python. *Term* yang tidak dibuthkan akan disingkirkan dalam proses ini.

```
def preprocess(s, lowercase=False):
    tokens = tokenize(s)
    if lowercase:
        tokens = [token if emoticon_re.search(token) else
token.lower() for token in tokens]
    return tokens
```

Source Code 4. 5 Baris Perintah Tokenisasi

Dokumen .json harus di-*load* terlebih dahulu sebelum melakukan proses tokenisasi. Pada `tokens = preprocess(tweet['text'])` merupakan kode untuk melakukan tokenisasi pada teks *tweet.* Kemudian adalah proses output dari tokenisasi.Berikut ini adalah source code tokenization.

```
with open('stream_1.json', 'r') as f:
            for line in f:
                tweet = json.loads(line)
                tokens = preprocess(tweet['text'])
                print(json.dumps(tokens))
```

Source Code 4. 6 Load Dokumen Tokenisasi

Berikut ini adalah keseluruhan source code tokensisasi dalam bahasa pemrograman Python.

```
from nltk.tokenize import word_tokenize
import re
import json

emoticons_str = r"""
    (?:
        [:=;] # Eyes
        [oO\-]? # Nose (optional)
        [D\)\]\(\]/\\OpP] # Mouth
    )"""

regex_str = [
    emoticons_str,
    r'<[^>]+>', # HTML tags
    r'(?:@[\w_]+)', # @-mentions
    r"(?:\#+[\w_]+[\w\'_\-]*[\w_]+)", # hash-tags
    r'http[s]?://(?:[a-z]|[0-9]|[$-_@.&+]|[!*\(\),]|(?:%[0-
9a-f][0-9a-f]))+', # URLs

    r'(?:(?:\d+,?)+(?:\.?\d+)?)', # numbers
    r"(?:[a-z][a-z'\-_]+[a-z])", # words with - and '
    r'(?:[\w_]+)', # other words
    r'(?:\S)' # anything else
```

```
]

tokens_re = re.compile(r'('+'|'.join(regex_str)+')',
re.VERBOSE | re.IGNORECASE)
emoticon_re = re.compile(r'^'+emoticons_str+'$', re.VERBOSE
| re.IGNORECASE)


def tokenize(s):
    return tokens_re.findall(s)

def preprocess(s, lowercase=False):
    tokens = tokenize(s)
    if lowercase:
        tokens = [token if emoticon_re.search(token) else
token.lower() for token in tokens]
    return tokens

with open('stream_1.json', 'r') as f:
            for line in f:
                tweet = json.loads(line)
                tokens = preprocess(tweet['text'])
                print(json.dumps(tokens))
```

Source Code 4. 7 Tokenisasi

4.1.3 Implementasi Stopword

Implementasi stopword digunakan untuk membuang kata-kata yang tidak bermakna dan tidak dibutuhkan untuk penghitungan tf-idf. Berikuti ini adalah source code stopword.

```
import operator
from nltk.corpus import stopwords
import string
from nltk.tokenize import word_tokenize
import re
import json
from collections import Counter

emoticons_str = r"""
    (?:
        [:=;] # Eyes
        [oO\-]? # Nose (optional)
        [D\)\]\(\]/\\OpP] # Mouth
    )"""
```

```
regex_str = [
    emoticons_str,
    r'<[^>]+>', # HTML tags
    r'(?:@[\w_]+)', # @-mentions
    r"(?:\#+[\w_]+[\w\'_\-]*[\w_]+)", # hash-tags
    r'http[s]?://(?:[a-z]|[0-9]|[$-_@.&+]|[!*\(\),]|(?:%[0-
9a-f][0-9a-f]))+', # URLs

    r'(?:(?:\d+,?)+(?:\.?\d+)?)', # numbers
    r"(?:[a-z][a-z'\-_]+[a-z])", # words with - and '
    r'(?:[\w_]+)', # other words
    r'(?:\S)' # anything else
]

tokens_re = re.compile(r'('+'|'.join(regex_str)+')',
re.VERBOSE | re.IGNORECASE)

punctuation = list(string.punctuation)
stop = stopwords.words('english') + punctuation + ['rt',
'via']

def tokenize(s):
    return tokens_re.findall(s)

def preprocess(s, lowercase=False):
    tokens = tokenize(s)
    if lowercase:
        tokens = [token if emoticon_re.search(token) else
token.lower() for token in tokens]
    return tokens

fname = 'stream_backup.json'
with open(fname, 'r') as f:
    count_all = Counter()
    for line in f:
        tweet = json.loads(line)
        # Create a list with all the terms
        #terms_all = [term for term in
preprocess(tweet['text'])]
        terms_stop = [term for term in
preprocess(tweet['text']) if term not in stop]
        # Update the counter
        count_all.update(terms_stop)
    # Print the first 5 most frequent words
    print(count_all.most_common(5))
```

Source Code 4. 8 Script Implementasi Stopword

4.1.4 Implementasi Term Frequency

Pada tahap ini dokumen tweet akan dihitung seberapa banyak frekuensi *term* atau kata yang muncul dalam sebuah dokumen. Dalam hal ini kata yang akan dihitung frekuensinya adalah kata ”gempa”. Selain kata “gempa” juga bisa ditampilkan 10 kata lain dengan frekuensi kemunculan tertinggi. Untuk memulai penghitungan frekuansi diawali dengan *load* dokumen json ke dalam *script*.

```
fname = 'tokenise.json'
with open(fname, 'r') as f:
    count_all = Counter()
    for line in f:
   tweet = json.loads(line)
```

Source Code 4. 9 Load Dokumen

Proses selanjutnya adalah meghitung dan menampilkan hasil dari penghitungan frekuensi *term* di dalam dokumen. Sepuluh *term* dengan frekuensi kemunculan tertinggi akan ditampilkan.

```
# Create a list with all the terms
        terms_all = [term for term in
preprocess(tweet['text'])]
        # Update the counter
        count_all.update(terms_all)
    # Print the first 10 most frequent words
print(count_all.most_common(10))
```

Source Code 4. 10 Menghitung Frekuensi Term

Berikut ini adalah implementasi penuh proses Term Frequnecy dalam bahasa pemrograman Python.

```
import operator
import json
import string
from nltk.corpus import stopwords
from collections import Counter

fname = 'tokenise.json'
with open(fname, 'r') as f:
    count_all = Counter()
    for line in f:
        tweet = json.loads(line)
```

```
            # Create a list with all the terms
            terms_all = [term for term in
    preprocess(tweet['text'])]
            # Update the counter
            count_all.update(terms_all)
        # Print the first 10 most frequent words
        print(count_all.most_common(10))

    punctuation = list(string.punctuation)
    stop = stopwords.words('english') + punctuation +
    ['rt', 'via']

    terms_stop = [term for term in
    preprocess(tweet['text']) if term not in stop]

    # Count terms only once, equivalent to Document
    Frequency
    terms_single = set(terms_all)
    # Count hashtags only
    terms_hash = [term for term in
    preprocess(tweet['text'])
                if term.startswith('#')]
    # Count terms only (no hashtags, no mentions)
    terms_only = [term for term in
    preprocess(tweet['text'])
                if term not in stop and
             not term.startswith(('#', '@'))]
```

Source Code 4. 11 Term Frequency

4.1.5 Implementasi TF-IDF

Implementasi TF-IDF ini merupakan tahap terkahir sebelum data yang dihasilkan dapat ditampilkan. Dokumen yang akan dihitung bobotnya dimasukkan ke dalam script. Berikut ini adalah *source code* untuk menjalankan metode TF-IDF dalam sistem twitter *text mining* untuk informasi gempa bumi menggunakan TF-IDF di Indonesia.

```
from __future__ import division, unicode_literals
import math
from textblob import TextBlob as tb
import json

def tf(word, blob):
    return blob.words.count(word) / len(blob.words)
```

```
def n_containing(word, bloblist):
    return sum(1 for blob in bloblist if word in blob)

def idf(word, bloblist):
    return math.log(len(bloblist) / (1 + n_containing(word,
bloblist)))

def tfidf(word, blob, bloblist):
    return tf(word, blob) * idf(word, bloblist)

document1 = tb("""["gempa", "Super", "Dahsyat", "Preksi",
"Akan", " ", ":   Lybhk3N0"]
["gempa", "berkekuatan", "M", "4.5", "-", "27", "", "NW",
"of", "Ayaviri", ",", "Peru", ":", "DYFI", "?", "-",
"ITime", "20 ", "-", "06", "-", "18", "15", ":", "06", ":",
"58", "UTC", "20 ", ".", ".", ".", ":  m4ogfSnKZi", "Wasp
", "bencana", "!"]
["Papua", "guncang", "gempa", "4,7", "", "-", "Okezone", ":
sTp2FTsgcN", "#Yahukimo"]
["1", ".", "Faktor", "Alam", "murni", "penyebab",
"utamanya", " lah", "alam", "itu", "senri", ",", "Cont",
";", "gempa", "bumi", ",", "tsunami", ",", "b i", ",",
"dan", "letusan", "gunung", "berapi", ".",
"#Bencanajateng"]
["Tiap", "kali", " ", "kreta", "lewat", ",", "lantai", "2",
"serasa", "goyang", ".", ".", "Klo", "pas", "lg", "gak",
"konsen", "kira", "gempa", ".", ".", " ud83d ude2f", "
ud83d ude2f", "Buset", "dah", ".", ".", "Resiko", "rumah",
"deket", "rel", "ni", "ya", ".", ".", " ud83d ude2c"]
["gempa", "Aceh", "Desember", "2004,", "melepaskan",
"energi", "sebesar", "23.3", "megaton", "TNT", "atau",
"setara", "dengan", "1502", "bom", "atom", "Hiroshima"]
["Pisang", "pisang", "pisang", " ud83c udf4c", " ud83c
udf4c", " ud83c udf4c", "(", "with", "Pisang", "at",
"Monumen", "gempa", "sumbar", ")", " u2014", ":
4wrxXyR2nf"]""")

Document2 = tb("""["Eventually", "nggak", "bahas",
"phobia", "gempa", "Mitsurugi", ".", ".", ".", "piyeeee",
".", ".", ".", "apa", "minggu", "depan", ".", ".", ".",
"?"]
["Beb", ",", "", "kangen", "nih", "|", "kangen", "sama",
"", "ya", "neng", "?", "|", "bkn", ",", "", "kangen",
"sama", "kumisku", "dulu", "|", "Wekk", "*", "lalu",
"gempa", "*"]
["@jinnyxxpark", "@iGojunhoe", "@laclice", "gempa", "bumi",
"ya", "jin"]
["", "@infoBMKG", ":", "#gempa", "rasakan", " ", ":",
"4.6", "", ",", "18", "-", "Jun", "-", "20 ", "13", ":",
"44", ":", "40", "", "(", "Pusat", "gempa", "", "darat",
```

```
"31", "", "Barat", "laut", "Kotaagung", ")", ",", "", ":",
"6", "", "#BMKG"]
["The", "latest", "Seputar", "gempa", "!", ":
s8xxdqePLA", "via", "@ndorokng"]
["@jungyeio", "ayo", "yein", " ud83d ude04", "temenan",
"yang", "aweet", "ya", "sampe", "rpw", "gempa", " ", "g"]
["", "@BeritaBodor", ":", "Telah", " ", "gempa",
"berkekuatan", "9,9", "", "kedalaman", "hati", "jomblo", "
", "karena", "bergesernya", "lempeng", "perasaan", "dari",
"mantan"]
["Misal", ":", "negara", "XXX", "guncang", "gempa", "dan",
"0000", "penduduk", "meninggal", "dunia", "-", "-", "&",
"gt", ";", "oh", "wajar", ".", "Negara", "XXX", "kan",
"negara", "kafir", "jd", "azab", "Tuhan", ".", "#yakeles"]
["", "@NakLuah", ":", "Phone", "vibrate", "dekat", "rumah",
":", " ", "perasan", ".", "Phone", "vibrate", "dekat",
"sekolah", ":", "gempa", "bumi", "6.7", "skala", "ricter",
"."]
["Kamu", "merasa", "gempa", "gak", "?", "?", "?", " ",
"kamu", "yang", "mengguncang", "hatiku", "sih", ".",
"@Nala_JKT48"]
["", "@pasagmerapi", ":", "Indonesia", "super", "market",
"bencana", ",", "brbagai", "ancaman", "bencana", " ", ",",
"gempa", "bumi", ",", "tsunami", ",", "tanah", "longsor",
",", "banjir", ",", "letusan", "g", ".", "api", "dll", ",",
" "]
["", "@HauraMujahidah", ":", "gempa", "bumi", "15", "saat",
".", "Boleh", "buatkan", "bangunan", "re ", "Manusia",
"lari", "lintang", "pukang", "Ramai", " ", "tahu", "hala",
"tuju", "Agaknya", ",", "kiama", "  "]
["", "@pasagmerapi", ":", "Indonesia", "super", "market",
"bencana", ",", "brbagai", "ancaman", "bencana", " ", ",",
"gempa", "bumi", ",", "tsunami", ",", "tanah", "longsor",
",", "banjir", ",", "letusan", "g", ".", "api", "dll", ",",
"  "]
["@cheonsommi", "pdhl", "gwa", "tu", "nanyain", "lo",
"kena", "gempa", "apa", "nggak", "hmzzz"]
["@myoomina_", "gempa", "apaan", "?", "?", "?", "?",
"kapan", "gempa", "?", "?", "?"]
["@cheonsommi", "lombok", "kan", "habis", "gempa", "!",
"!", "lombok", "mananya", "bali", "sih", "ga", "ngei"]
["Papua", "guncang", "gempa", "4,7", "", "-", "Okezone", ":
JONr2KkWWF", "#Yahukimo"]""")

document3 = tb("""["", "@pasagmerapi", ":", "Indonesia",
"super", "market", "bencana", ",", "brbagai", "ancaman",
"bencana", " ", ",", "gempa", "bumi", ",", "tsunami", ",",
"tanah", "longsor", ",", "banjir", ",", "letusan", "g",
".", "api", "dll", ",", "  "]
["", "@pasagmerapi", ":", "Indonesia", "super", "market",
"bencana", ",", "brbagai", "ancaman", "bencana", " ", ",",
```

```
"gempa", "bumi", ",", "tsunami", ",", "tanah", "longsor",
",", "banjir", ",", "letusan", "g", ".", "api", "dll", ",",
"  "]
["@myoomina_", "sotaw", "abis", "mana", " ", "gempa",
"chingu"]
["", "@pasagmerapi", ":", "Indonesia", "super", "market",
"bencana", ",", "brbagai", "ancaman", "bencana", " ", ",",
"gempa", "bumi", ",", "tsunami", ",", "tanah", "longsor",
",", "banjir", ",", "letusan", "g", ".", "api", "dll", ",",
"  "]
[" ud83d ude1f", ":   czC1ccOs"]
["", "@NakLuah", ":", "Phone", "vibrate", "dekat", "rumah",
":", " ", "perasan", ".", "Phone", "vibrate", "dekat",
"sekolah", ":", "gempa", "bumi", "6.7", "skala", "ricter",
"."]
["Lah", "tibo", "lo", "siko", ",", "Gan", "?", "(", "at",
"Monumen", "gempa", "sumbar", ")", " u2014", ":
PTA8kAXweA"]
["", "@pasagmerapi", ":", "Indonesia", "super", "market",
"bencana", ",", "brbagai", "ancaman", "bencana", " ", ",",
"gempa", "bumi", ",", "tsunami", ",", "tanah", "longsor",
",", "banjir", ",", "letusan", "g", ".", "api", "dll", ",",
"  "]
["kamu", "itu", "kayak", "lempengan", "bumi", "ya", ",",
"geser", "kit", "aja", "bisa", "gempa-in", "hati", "", "."]
["gempa", "berkekuatan", "M", "4.8", "-", "Southwest",
"Inan", "Ridge", ":", "DYFI", "?", "-", "ITime", "20 ", "-
", "06", "-", "18", "14", ":", "52", ":", "38", "UTC", "20
", "-", "0", ".", ".", ".", ":   Oe1EOSukOe", "Wasp ",
"bencana", "!"]
["", "@infopapuan", ":", "gempa", "Darat", "", "",
"guncang", "Papua", "-", ":   2M4K9ZGNzv", ":
E0O79ysx6r", "#Paniai"]
["", "@NakLuah", ":","Phone", "vibrate", "dekat", "rumah",
":", " ", "perasan", ".", "Phone", "vibrate", "dekat",
"sekolah", ":", "gempa", "bumi", "6.7", "skala", "ricter",
"."]"""")

bloblist = [document1, document2, document3]
for i, blob in enumerate(bloblist):
    print("Top words in document {}".format(i + 1))
    scores = {word: tfidf(word, blob, bloblist) for word in
blob.words}
    sorted_words = sorted(scores.items(), key=lambda x:
x[1], reverse=True)
    for word, score in sorted_words[:10]:
        print("Word: {}, TF-IDF: {}".format(word,
round(score, 5)))
```

Source Code 4. 12 Implementasi TF-IDF

4.2 Pembahasan Uji Coba

Pada sub bab ini akan dijelaskan mengenai analisis hasil pengujian sistem yang berfungsi untuk mengetahui kinerja dari program dalam melakukan proses tf-idf. Pengujian yang dilakukan pada sistem ini adalah pengujian *testing*. Pengujian sistem yang dilakukan adalah dengan menjalankan source code python di dalam terminal linux. Kemudian dari proses ini lah didapatkan *tweet* mengenai gempa bumi dari twitter API, yang kemudian diolah menggunakan tf-idf.

Pengujian dilakukan dengan menjalankan *stream* twitter API selama 6 jam 45 menit, yakni mulai pukul 21.00 hingga 03.45 WIB yang menghasilkan dokumen *tweet* dengan *term* "gempa" sebanyak 520 *tweet*. Dokumen yang dihasilkan dari *stream* ini lah yang diuji coba lebih lanjut. Hasil dari penghitungan tf-idf ditampilkan pada Tabel 4.1 yang menunjukkan *term* "gempa" memiliki bobot 0.036 dari keseleuruhan dokumen.

Tabel 4. 1 Hasil TF-IDF

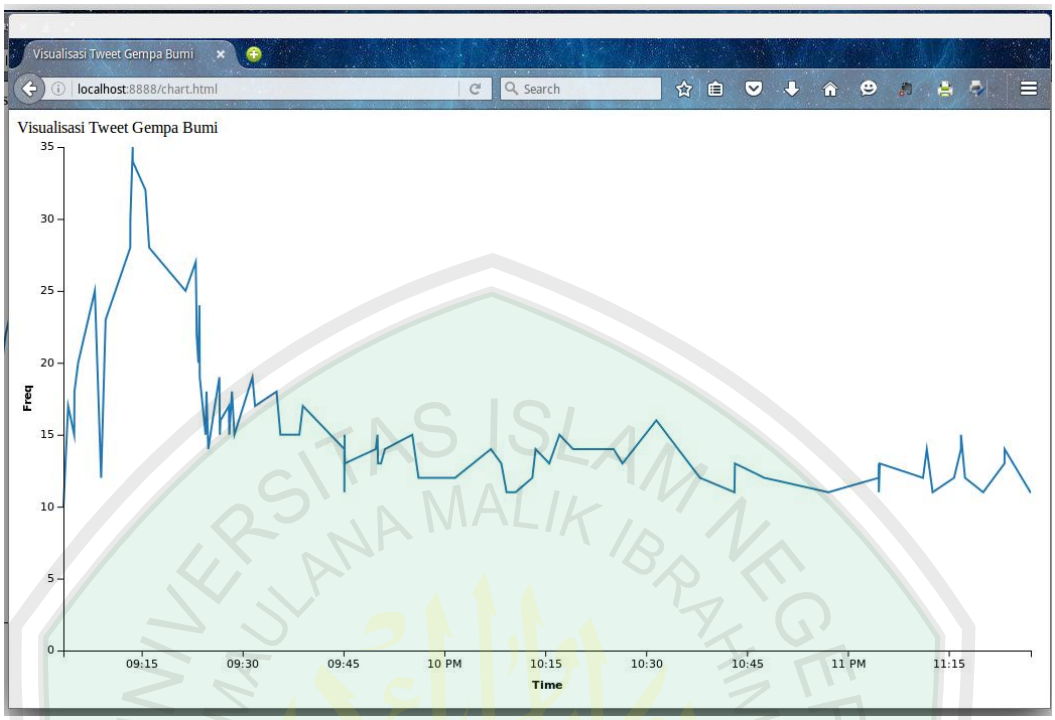
Term	TF-IDF
gempa	0.036
purworejo	0.009
bumi	0.007
guncang	0.007
longsor	0.002
jateng	0.002
wilayah	0.002
sumbar	0.002
dampak	0.002
goyang	0.002

Setelah dilakukan penghitungan tf –idf dan tinjauan statistik, data yang dihasilkan belum layak untuk dijadikan informasi terjadinya gempa bumi. Karena

dari proses *stream* twitter API yang berjalan selama 6 jam 45 menit dan menghasilkan 520 dokumen *tweet*, rata-rata *tweet* yang menyebutkan *term* tentang "gempa" adalah 1 tweet tiap menitnya. Nilai ini didapatkan dari menghitung nilai mean pada dokumen tweet yang dihasilkan. Nilai mean yang dihaslikan adalah 1.29 dan nilai median yang dihasilkan adalah 1. Hal ini bisa terjadi karena pada tiap menit selama stream twitter API berjalan, bisa ada satu *tweet* atau beberapa *tweet*, bahkan bisa tidak ada *tweet* sama sekali. Dalam penelitian ini, dapat dikatan terjadinya gempa apabila dalam satu waktu terdapat 100 tweet dalam satu menit. Kareana apabila ada 100 *tweet* yang menyampaikan tentang gempa secara hampir bersamaan dalam satu menit, dapat dirasionalkan bahwa ada 100 orang yang merasakan terjadinya gempa bumi.

4.3 Pembahasan Hasil

Berdasarkan hasil uji coba, penggunaan tf-idf untuk *text mining* menghasilkan data yang presisi dan teruji. Hal ini terbukti dengan data yang dihasilkan dapat menampilkan frekuensi kemunculan *term*. Tetapi data yang dihasilkan belum layak untuk dijadikan sebagai informasi terjadinya gempa bumi. Karena nilai yang dihasilkan masih jauh dari nilai patokan terjadinya gempa bumi, yakni 100 tweet dalam satu menit. Data hasil pengujian juga dapat divisualkan. Agar hasil pengujian mudah dipahami, maka data akan ditampilkan dalam bentuk grafik.



Gambar 4. 1 Grafik Time Chart Tweet Gempa Bumi

4.4 Integrasi Nilai Islam

Penelitian ini juga membahas mengenai integrasi nilai islam. Surah Al-Zalzalah ayat 1 – 3 dikutip dari Al-qur'an sebagai ayat yang sesuai dengan masalah yang dibahas, yakni tentang terjadinya gempa bumi dan bencana alam di kemudian hari.

إِذَا زُلْزِلَتِ ٱلْأَرْضُ زِلْزَالَهَا ﴿١﴾ وَأَخْرَجَتِ ٱلْأَرْضُ أَثْقَالَهَا ﴿٢﴾ وَقَالَ ٱلْإِنسَٰنُ مَا لَهَا ﴿٣﴾

Apabila bumi digoncangkan dengan goncangannya, dan bumi telah mengeluarkan beban-beban beratnya, dan manusia bertanya: "Apa (yang terjadi) baginya?"

Surah **الزلزلة** memiliki arti "goncangan", goncangan yang dimaksudkan adalah gempa bumi. Dijelaskan dalam tafsir Al-Misbah volume 15 bahwa surah ini berbicara tentang awal terjadinya hari kemudian. Allah berfirman: *Apabila* – dan itu pasti terjadi – *bumi digoncangkan dengan goncangannya* yang dahsyat yang hanya terjadi sekali dalam kedahsyatan seperti itu, *dan* persada *bumi* di seluruh penjurunya tanpa kecuali *telah mengeluarkan beban-beban berat* yang dakiandung-*nya*, baik manusia yang telah mati maupun barang tambang yang dipendamnya atau apapun selainnya *dan* ketika itu *manusia* yang sempat mengalaminya *bertanya* – dalam hatinya – keheranan: *"Apa* yang terjadi *baginya* sehingga dia bergoncang demikian dahsyat dan mengeluarkan isi perutnya*?"*

Kata *idza* digunakan al-Qur'an untuk sesuatu yang pasti akan terjadi, berbeda dengan kata *in* yang biasa digunakan untuk sesuatu yang belum atau jarang terjadi, dan berbeda pula dengan *lau* yang digunakan untuk mengandaikan sesuatu yang mustahil terjadi. Dengan demikian ayat di atas mengisyaratkan kepastian terjadinya goncangan bumi yang diuraikan ini.

Pengulangan kata *al-ardh / bumi* pada ayat kedua mengisyaratkan bahwa goncangan dan pengeluaran isi perut bumi itu terjadi di seluruh wilayah bumi kecuali, dan ini adalah salah satu yang membedakan antara goncangan atau gempa yang terjadi selama ini, karena gempa tersebut hanya terjadi pada wilayah terbatas dari bumi.

Gempa bumi adalah peristiwa bergetarnya bumi akibat pelepasan energi di dalam bumi secara tiba-tiba yang ditandai dengan patahnya lapisan batuan pada kerak bumi. Akumulasi energi penyebab terjadinya gempa bumi dihasilkan dari

pergerakan lempeng-lempeng tektonik. Energi yang dihasilkan dipancarkan kesegala arah berupa gelombang gempabumi sehingga efeknya dapat dirasakan sampai ke permukaan bumi.

Selama ini informasi mengenai gempa bumi atau bencana alam lainnya selalu disampaikan oleh Badan Meteorologi Klimatologi dan Geofisika atau BMKG. Kehadiran media sosial memberikan pesaing baru bagi BMKG dalam menginformasikan suatu peristiwa gempa atau pun bencana, mereka adalah masyarakat yang selalu ingin berkabar tentang apa yang masyarakat alami. Twitter merupakan salah satu media sosial penyedia layanan *micro blogging* bagi para penggunanya. Pengguna Twitter selalu cepat dalam mengabarkan suatu peristiwa atau kejadian, misalnya gempa bumi. Bahkan kabar ini bisa lebih cepat sampai kepada masyarakat lain atau pemerintah dari pada informasi yang dirilis oleh BMKG.

Twitter *text mining* merupakan sebuah sistem yang dapat menggali data dari twitter untuk kemudian diolah dan dijadikan informasi tentang terjadinya gempa bumi di Indonesia. Melalui Twitter *text mining* dapat memberikan informasi mengenai gempa bumi di Indonesia bagi masyarakat secara cepat.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Setelah melakukan analisis dan pengujian yang dilakukan pada bab sebelumnya dan berdasarkan rumusan masalah pada bab pertama, data yang dihasilkan menunjukkan bahwa Term Frequncy – Inverse Document Frequency (TF-IDF) dapat digunakan untuk *text mining*. Adapun data yang dihasilkan data yang dihasilkan dari proses *stream* twitter API selama 6 jam 45 menit sebanyak 520 dokumen *tweet*. Dari 520 dokumen yang dihalsikan, rata-rata frekuensi tweet yang didapatkan dalam satu menit adalah 1 tweet dengan bobot tf-idf 0.036. Sehingga bisa diambil kesimpulan bahwa tf-idf dapat digunakan untuk *text mining* tetapi data tweet yang dihasilkan belum layak untuk dijadikan sebagai informasi terjadinya gempa bumi.

Sistem ini memang masih memiliki kekurangan pada keakuratan kata yang dihasilkan karena belum dapat mendeteksi apakah *tweet* tersebut merupakan *tweet* yang benar-benar mengabarkan tentang gempa bumi atau bukan.

5.2 Saran

Berdasarkan kesimpulan di atas, masih banyak kekurangan dalam penelitian aplikasi penghitungan frekuensi untuk informasi gempa bumi ini. Peneliti menyarankan pengembangan penelitian lebih lanjut pada sisitem untuk tidak hanya mengabarkan gempa bumi saja, namun juga peringatan dan informasi lokasi terjadinya gempa bumi yang lebih akurat.

DAFTAR PUSTAKA

Berry, M.W. & Kogan, J. 2010. *Text mining Aplication and theory*. WILEY : United Kingdom.

BMKG. 2014. *Gempabumi, Apakah Gempa bumi itu?* http://www.bmkg.go.id/BMKG_Pusat/Gempabumi_-_Tsunami/ Gempabumi.bmkg diakses tanggal 5 Juni 2016

Bonzanini, Marco. 2015. *Mining Twitter Data with Python (Part 1: Collecting data).* https://marcobonzanini.com/2015/03/02/mining-twitter-data-with-python-part-1/ Diakses tanggal 6 Juni 2016

Dragut, E., Fang, F., Sistla, P., Yu, S. & Meng, W. 2009. *Stop Word and Related Problems in Web Interface Integration*. http://www.vldb.org/pvldb/2/ vldb09-384.pdf. Diakses tanggal 21 Juni 2016.

Feldman, R & Sanger, J. 2007. *The Text mining Handbook : Advanced Approaches in Analyzing Unstructured Data*. Cambridge University Press : New York.

Han, J & Kamber, M. 2006. *Data Mining: Concepts and Techniques Second Edition.* Morgan Kaufmann publisher : San Francisco.

Ilyas, Husni. 2010. *Unified Messaging System Information Retrieval & Klasifikasi Teks.* https://komputasi.wordpress.com/2010/01/22/unified-messaging-system-information-retrieval-klasifikasi-teks/ Diakses tanggal 28 Juni 2016

Mandala R. 2004. *Bahan Kuliah Sistem Temu Kembali Informasi.* Institut Teknologi Bandung. Departemen Teknik Informatika.

Mandala, R. & Setiawan, Hendra. 2002. *Peningkatan Performansi sistem Temu-Kembali Informasi dengan Perluasan Query Secara Otomatis.* ITB : Bandung.

Manning, Christopher D, Ragnavan Prabhakar, Schutze, Hinrich. 2009. *Introduction to Information Retrieval*. Cambridge University Press

Manulu, Boy Utomo. 2014. *Analisis Sentimen pada Twitter Menggunakan Text mining*. USU : Medan.

Tala, Fadillah Z. 2003. *A Study of Stemming Efects on Information Retrieval in Bahasa Indonesia. Institute for Logic, Language and Computation Universiteit van Amsterdam The Netherlands.* http://www.illc.uva.nl/ Research/Reports/MoL-2003-02.text.pdf. Diakses tanggal 6 Juni 2016.

Twitter. 2013. *Twitter Support.* https://support.twitter.com/. Diakses tanggal 5 Juni 2016.

Wang, A. H. 2010. *Don't Follow Me: Twitter Spam Detection*. Proceedings of 5th International Conference on Security and Cryptography (SECRYPT) Athens 2010: pp. 1-10. California:IEEE.

Weiss, S.M., Indurkhya, N., Zhang, T., Damerau, F.J. 2005. *Text mining : Predictive Methods fo Analyzing Unstructered Information*. Springer : New York

Witten, E. H., Frank, E., & Hall, M. A. 2011. *Data Mining Practical Machine Learning Tools and Techniques Third Edition.* Burlington, MA, USA: Elsevier Inc.

Zafikri, Atika. 2010. *Implementasi Metode Term Frequency Inverse Document Frequency (TF-IDF) Pada Sistem Temu Kembali Informasi.* http://repository.usu.ac.id/handle/123456789/16465 Diakses tanggal 14 Juni 2016

Lampiran

1. twitterStreaming.py

```
#import requests.packages.urllib3
#requests.packages.urllib3.disable_warnings()

import tweepy
from tweepy import Stream
from tweepy import OAuthHandler
from tweepy.streaming import StreamListener
import time
import argparse
import string
#import config
import json

ckey = 'wmSMYOFxQ867iGfvoaEBdAOpg'
csecret =
'dvJTua6me7cXVjUgyneWru6tiqX70wtPNMWmZlXGyOUeGa691W'
atoken = '61111680-
N5KCNpLzSz5f7bhOnZeXtVzTmFOdnWlvBq5o09GT0'
asecret = 'snYPccETaazso21XowfusBZKxcz8uQ2mxrjgoHYb8BIFv'

def get_parser():
    """Get parser for command line arguments."""
    parser = argparse.ArgumentParser(description="Twitter
Downloader")
    parser.add_argument("-q",
                        "--query",
                        dest="query",
                        help="Query/Filter",
                        default='-')
    parser.add_argument("-d",
                        "--data-dir",
                        dest="data_dir",
                        help="Output/Data Directory")
    return parser

class MyListener(StreamListener):
    """Custom StreamListener for streaming data."""

    def __init__(self, data_dir, query):
        query_fname = format_filename(query)
        self.outfile = "%s/stream_%s.json" % (data_dir,
query_fname)

    def on_data(self, data):
        try:
            with open(self.outfile, 'a') as f:
                f.write(data)
```

```
                print(data)
                return True
        except BaseException as e:
            print("Error on_data: %s" % str(e))
            time.sleep(5)
        return True

    def on_error(self, status):
        print(status)
        return True

def format_filename(fname):
    """Convert file name into a safe string.

    Arguments:
        fname -- the file name to convert
    Return:
        String -- converted file name
    """
    return ''.join(convert_valid(one_char) for one_char in
fname)

def convert_valid(one_char):
    """Convert a character into '_' if invalid.

    Arguments:
        one_char -- the char to convert
    Return:
        Character -- converted char
    """
    valid_chars = "-_.%s%s" % (string.ascii_letters,
string.digits)
    if one_char in valid_chars:
        return one_char
    else:
        return '_'

@classmethod
def parse(cls, api, raw):
    status = cls.first_parse(api, raw)
    setattr(status, 'json', json.dumps(raw))
    return status

if __name__ == '__main__':
    parser = get_parser()
    args = parser.parse_args()
    auth = OAuthHandler(ckey, csecret)
    auth.set_access_token(atoken, asecret)
    api = tweepy.API(auth)

    twitter_stream = Stream(auth, MyListener(args.data_dir,
```

```
args.query))
    twitter_stream.filter(track=["gempa", "#gempa"])
#auth = OAuthHandler(ckey, csecret)
#auth.set_access_token(atoken, asecret)
#twitterStream = Stream(auth, listener())
#twitterStream.filter(track=["gempa"])
```

2. tokenise.py

```
from nltk.tokenize import word_tokenize
import re
import json

emoticons_str = r"""
    (?:
        [:=;] # Eyes
        [oO\-]? # Nose (optional)
        [D\)\]\(\]/\\OpP] # Mouth
    )"""

regex_str = [
    emoticons_str,
    r'<[^>]+>', # HTML tags
    r'(?:@[\w_]+)', # @-mentions
    r"(?:\#+[\w_]+[\w\'_\-]*[\w_]+)", # hash-tags
    r'http[s]?://(?:[a-z]|[0-9]|[$-
_@.&+]|[!*\(\),]|(?:%[0-9a-f][0-9a-f]))+', # URLs

    r'(?:(?:\d+,?)+(?:\.?\d+)?)', # numbers
    r"(?:[a-z][a-z'\-_]+[a-z])", # words with - and '
    r'(?:[\w_]+)', # other words
    r'(?:\S)' # anything else
]

tokens_re = re.compile(r'('+'|'.join(regex_str)+')',
re.VERBOSE | re.IGNORECASE)
emoticon_re = re.compile(r'^'+emoticons_str+'$',
re.VERBOSE | re.IGNORECASE)


def tokenize(s):
    return tokens_re.findall(s)

def preprocess(s, lowercase=False):
    tokens = tokenize(s)
    if lowercase:
        tokens = [token if emoticon_re.search(token) else
token.lower() for token in tokens]
    return tokens
```

```
with open('stream_backup.json', 'r') as f:
          for line in f:
              #line = f.readline()
              tweet = json.loads(line)
              tokens = preprocess(tweet['text'])
              #k = tokens
              #print(tokens)
              print(json.dumps(tokens))
```

3. stopwords.py

```
import operator
from nltk.corpus import stopwords
import string
from nltk.tokenize import word_tokenize
import re
import json
from collections import Counter

emoticons_str = r"""
    (?:
        [:=;] # Eyes
        [oO\-]? # Nose (optional)
        [D\)\]\(\]/\\OpP] # Mouth
    )"""

regex_str = [
    emoticons_str,
    r'<[^>]+>', # HTML tags
    r'(?:@[\w_]+)', # @-mentions
    r"(?:\#+[\w_]+[\w\'_\-]*[\w_]+)", # hash-tags
    r'http[s]?://(?:[a-z]|[0-9]|[$-
_@.&+]|[!*\(\),]|(?:%[0-9a-f][0-9a-f]))+', # URLs

    r'(?:(?:\d+,?)+(?:\.?\d+)?)', # numbers
    r"(?:[a-z][a-z'\-_]+[a-z])", # words with - and '
    r'(?:[\w_]+)', # other words
    r'(?:\S)' # anything else
]

tokens_re = re.compile(r'('+'|'.join(regex_str)+')',
re.VERBOSE | re.IGNORECASE)

punctuation = list(string.punctuation)
stop = stopwords.words('english') + punctuation + ['rt',
'via']

def tokenize(s):
    return tokens_re.findall(s)
```

```
def preprocess(s, lowercase=False):
    tokens = tokenize(s)
    if lowercase:
        tokens = [token if emoticon_re.search(token) else
token.lower() for token in tokens]
    return tokens

fname = 'stream_backup.json'
with open(fname, 'r') as f:
    count_all = Counter()
    for line in f:
        tweet = json.loads(line)
        # Create a list with all the terms
        #terms_all = [term for term in
preprocess(tweet['text'])]
        terms_stop = [term for term in
preprocess(tweet['text']) if term not in stop]
        # Update the counter
        count_all.update(terms_stop)
    # Print the first 5 most frequent words
    print(count_all.most_common(5))
```

4. termfrequency.py

```
import operator
from nltk.tokenize import word_tokenize
import re
import json
from collections import Counter

emoticons_str = r"""
    (?:
        [:=;] # Eyes
        [oO\-]? # Nose (optional)
        [D\)\]\(\]/\\OpP] # Mouth
    )"""

regex_str = [
    emoticons_str,
    r'<[^>]+>', # HTML tags
    r'(?:@[\w_]+)', # @-mentions
    r"(?:\#+[\w_]+[\w\'_\-]*[\w_]+)", # hash-tags
    r'http[s]?://(?:[a-z]|[0-9]|[$-
_@.&+]|[!*\(\),]|(?:%[0-9a-f][0-9a-f]))+', # URLs

    r'(?:(?:\d+,?)+(?:\.?\d+)?)', # numbers
    r"(?:[a-z][a-z'\-_]+[a-z])", # words with - and '
    r'(?:[\w_]+)', # other words
    r'(?:\S)' # anything else
]
```

```
tokens_re = re.compile(r'('+'|'.join(regex_str)+')',
re.VERBOSE | re.IGNORECASE)

def tokenize(s):
    return tokens_re.findall(s)

def preprocess(s, lowercase=False):
    tokens = tokenize(s)
    if lowercase:
        tokens = [token if emoticon_re.search(token) else
token.lower() for token in tokens]
    return tokens

fname = 'stream_1.json'
with open(fname, 'r') as f:
    count_all = Counter()
    for line in f:
        tweet = json.loads(line)
        # Create a list with all the terms
        terms_all = [term for term in
preprocess(tweet['text'])]
        # Update the counter
        count_all.update(terms_all)
    # Print the first 10 most frequent words
    print(count_all.most_common(10))
```

5. tfidf.py

```
from __future__ import division, unicode_literals
import math
from textblob import TextBlob as tb
import json

def tf(word, blob):
    return blob.words.count(word) / len(blob.words)

def n_containing(word, bloblist):
    return sum(1 for blob in bloblist if word in blob)

def idf(word, bloblist):
    return math.log(len(bloblist) / (1 + n_containing(word,
bloblist)))

def tfidf(word, blob, bloblist):
    return tf(word, blob) * idf(word, bloblist)

document1 = tb("""["gempa", "Super", "Dahsyat", "Preksi",
"Akan", " ", ":   Lybhk3N0"]
["gempa", "berkekuatan", "M", "4.5", "-", "27", "", "NW",
```

```
"of", "Ayaviri", ",", "Peru", ":", "DYFI", "?", "-",
"ITime", "20 ", "-", "06", "-", "18", "15", ":", "06", ":",
"58", "UTC", "20 ", ".", ".", ".", ":   m4ogfSnKZi", "Wasp
", "bencana", "!"]
["Papua", "guncang", "gempa", "4,7", "", "-", "Okezone", ":
sTp2FTsgcN", "#Yahukimo"]
["1", ".", "Faktor", "Alam", "murni", "penyebab",
"utamanya", " lah", "alam", "itu", "senri", ",", "Cont",
";", "gempa", "bumi", ",", "tsunami", ",", "b i", ",",
"dan", "letusan", "gunung", "berapi", ".", "#Bencanajateng"]
["Tiap", "kali", " ", "kreta", "lewat", ",", "lantai", "2",
"serasa", "goyang", ".", ".", "Klo", "pas", "lg", "gak",
"konsen", "kira", "gempa", ".", ".", " ud83d ude2f", " ud83d
ude2f", "Buset", "dah", ".", ".", "Resiko", "rumah",
"deket", "rel", "ni", "ya", ".", ".", " ud83d ude2c"]
["gempa", "Aceh", "Desember", "2004,", "melepaskan",
"energi", "sebesar", "23.3", "megaton", "TNT", "atau",
"setara", "dengan", "1502", "bom", "atom", "Hiroshima"]
["Pisang", "pisang", "pisang", " ud83c udf4c", " ud83c
udf4c", " ud83c udf4c", "(", "with", "Pisang", "at",
"Monumen", "gempa", "sumbar", ")", " u2014", ":
4wrxXyR2nf"]"""")

Document2 = tb(\"""["Eventually", "nggak", "bahas", "phobia",
"gempa", "Mitsurugi", ".", ".", ".", "piyeeee", ".", ".",
".", "apa", "minggu", "depan", ".", ".", ".", "?"]
["Beb", ",", "", "kangen", "nih", "|", "kangen", "sama", "",
"ya", "neng", "?", "|", "bkn", ",", "", "kangen", "sama",
"kumisku", "dulu", "|", "Wekk", "*", "lalu", "gempa", "*"]
["@jinnyxxpark", "@iGojunhoe", "@laclice", "gempa", "bumi",
"ya", "jin"]
["", "@infoBMKG", ":", "#gempa", "rasakan", " ", ":", "4.6",
"", ",", "18", "-", "Jun", "-", "20 ", "13", ":", "44", ":",
"40", "", "(", "Pusat", "gempa", "", "darat", "31", "",
"Barat", "laut", "Kotaagung", ")", ",", "", ":", "6", "",
"#BMKG"]
["The", "latest", "Seputar", "gempa", "!", ":   s8xxdqePLA",
"via", "@ndoroknq"]
["@jungyeio", "ayo", "yein", " ud83d ude04", "temenan",
"yang", "aweet", "ya", "sampe", "rpw", "gempa", " ", "g"]
["", "@BeritaBodor", ":", "Telah", " ", "gempa",
"berkekuatan", "9,9", "", "kedalaman", "hati", "jomblo", "
", "karena", "bergesernya", "lempeng", "perasaan", "dari",
"mantan"]
["Misal", ":", "negara", "XXX", "guncang", "gempa", "dan",
"0000", "penduduk", "meninggal", "dunia", "-", "-", "&",
"gt", ";", "oh", "wajar", ".", "Negara", "XXX", "kan",
"negara", "kafir", "jd", "azab", "Tuhan", ".", "#yakeles"]
["", "@NakLuah", ":", "Phone", "vibrate", "dekat", "rumah",
":", " ", "perasan", ".", "Phone", "vibrate", "dekat",
"sekolah", ":", "gempa", "bumi", "6.7", "skala", "ricter",
```

```
"."]
["Kamu", "merasa", "gempa", "gak", "?", "?", "?", " ",
"kamu", "yang", "mengguncang", "hatiku", "sih", ".",
"@Nala_JKT48"]
["", "@pasagmerapi", ":", "Indonesia", "super", "market",
"bencana", ",", "brbagai", "ancaman", "bencana", " ", ",",
"gempa", "bumi", ",", "tsunami", ",", "tanah", "longsor",
",", "banjir", ",", "letusan", "g", ".", "api", "dll", ",",
"  "]
["", "@HauraMujahidah", ":", "gempa", "bumi", "15", "saat",
".", "Boleh", "buatkan", "bangunan", "re ", "Manusia",
"lari", "lintang", "pukang", "Ramai", " ", "tahu", "hala",
"tuju", "Agaknya", ",", "kiama", "  "]
["", "@pasagmerapi", ":", "Indonesia", "super", "market",
"bencana", ",", "brbagai", "ancaman", "bencana", " ", ",",
"gempa", "bumi", ",", "tsunami", ",", "tanah", "longsor",
",", "banjir", ",", "letusan", "g", ".", "api", "dll", ",",
"  "]
["@cheonsommi", "pdhl", "gwa", "tu", "nanyain", "lo",
"kena", "gempa", "apa", "nggak", "hmzzz"]
["@myoomina_", "gempa", "apaan", "?", "?", "?", "?",
"kapan", "gempa", "?", "?", "?"]
["@cheonsommi", "lombok", "kan", "habis", "gempa", "!", "!",
"lombok", "mananya", "bali", "sih", "ga", "ngei"]
["Papua", "guncang", "gempa", "4,7", "", "-", "Okezone", ":
JONr2KkWWF", "#Yahukimo"]"""")

document3 = tb("""["", "@pasagmerapi", ":", "Indonesia",
"super", "market", "bencana", ",", "brbagai", "ancaman",
"bencana", " ", ",", "gempa", "bumi", ",", "tsunami", ",",
"tanah", "longsor", ",", "banjir", ",", "letusan", "g", ".",
"api", "dll", ",", "  "]
["", "@pasagmerapi", ":", "Indonesia", "super", "market",
"bencana", ",", "brbagai", "ancaman", "bencana", " ", ",",
"gempa", "bumi", ",", "tsunami", ",", "tanah", "longsor",
",", "banjir", ",", "letusan", "g", ".", "api", "dll", ",",
"  "]
["@myoomina_", "sotaw", "abis", "mana", " ", "gempa",
"chingu"]
["", "@pasagmerapi", ":", "Indonesia", "super", "market",
"bencana", ",", "brbagai", "ancaman", "bencana", " ", ",",
"gempa", "bumi", ",", "tsunami", ",", "tanah", "longsor",
",", "banjir", ",", "letusan", "g", ".", "api", "dll", ",",
"  "]
[" ud83d ude1f", ":   czC1ccOs"]
["", "@NakLuah", ":", "Phone", "vibrate", "dekat", "rumah",
":", " ", "perasan", ".", "Phone", "vibrate", "dekat",
"sekolah", ":", "gempa", "bumi", "6.7", "skala", "ricter",
"."]
["Lah", "tibo", "lo", "siko", ",", "Gan", "?", "(", "at",
"Monumen", "gempa", "sumbar", ")", " u2014", ":
```

```
PTA8kAXweA"]
["", "@pasagmerapi", ":", "Indonesia", "super", "market",
"bencana", ",", "brbagai", "ancaman", "bencana", " ", ",",
"gempa", "bumi", ",", "tsunami", ",", "tanah", "longsor",
",", "banjir", ",", "letusan", "g", ".", "api", "dll", ",",
"  "]
["kamu", "itu", "kayak", "lempengan", "bumi", "ya", ",",
"geser", "kit", "aja", "bisa", "gempa-in", "hati", "", "."]
["gempa", "berkekuatan", "M", "4.8", "-", "Southwest",
"Inan", "Ridge", ":", "DYFI", "?", "-", "ITime", "20 ", "-",
"06", "-", "18", "14", ":", "52", ":", "38", "UTC", "20 ",
"-", "0", ".", ".", ".", ":   Oe1EOSukOe", "Wasp ",
"bencana", "!"]
["", "@infopapuan", ":", "gempa", "Darat", "", "",
"guncang", "Papua", "-", ":   2M4K9ZGNzv", ":   E0O79ysx6r",
"#Paniai"]
["", "@NakLuah", ":", "Phone", "vibrate", "dekat", "rumah",
":", " ", "perasan", ".", "Phone", "vibrate", "dekat",
"sekolah", ":", "gempa", "bumi", "6.7", "skala", "ricter",
"."]"""")

bloblist = [document1, document2, document3]
for i, blob in enumerate(bloblist):
    print("Top words in document {}".format(i + 1))
    scores = {word: tfidf(word, blob, bloblist) for word in
blob.words}
    sorted_words = sorted(scores.items(), key=lambda x:
x[1], reverse=True)
    for word, score in sorted_words[:10]:
          print("Word: {}, TF-IDF: {}".format(word,
   round(score, 5)))
```

6. pandas_timechart.py

```
import operator
from nltk.corpus import stopwords
import string
from nltk.tokenize import word_tokenize
import re
import pandas
import json
import vincent
from collections import Counter

emoticons_str = r"""
    (?:
        [:=;] # Eyes
        [oO\-]? # Nose (optional)
        [D\)\]\(\]/\\OpP] # Mouth
    )"""
```

```
regex_str = [
    emoticons_str,
    r'<[^>]+>', # HTML tags
    r'(?:@[\w_]+)', # @-mentions
    r"(?:\#+[\w_]+[\w\'_\-]*[\w_]+)", # hash-tags
    r'http[s]?://(?:[a-z]|[0-9]|[$-
_@.&+]|[!*\(\),]|(?:%[0-9a-f][0-9a-f]))+', # URLs

    r'(?:(?:\d+,?)+(?:\.?\d+)?)', # numbers
    r"(?:[a-z][a-z'\-_]+[a-z])", # words with - and '
    r'(?:[\w_]+)', # other words
    r'(?:\S)' # anything else
]

tokens_re = re.compile(r'('+'|'.join(regex_str)+')',
re.VERBOSE | re.IGNORECASE)

punctuation = list(string.punctuation)
stop = stopwords.words('english') + punctuation + ['rt',
'via']

#word_freq = count_terms_only.most_common(20)
#labels, freq = zip(*word_freq)
#data = {'data': freq, 'x': labels}
#bar = vincent.Bar(data, iter_idx='x')
#bar.to_json('term_freq.json')

def tokenize(s):
    return tokens_re.findall(s)

def preprocess(s, lowercase=False):
    tokens = tokenize(s)
    if lowercase:
        tokens = [token if emoticon_re.search(token) else
token.lower() for token in tokens]
    return tokens

fname = 'stream_3.json'
with open(fname, 'r') as f:
   count_all = Counter()
   dates_gempa = []
   for line in f:
        tweet = json.loads(line)
        terms_stop = [term for term in
preprocess(tweet['text']) if term.startswith('G')]
        terms_hash = [term for term in
preprocess(tweet['text']) if term.startswith('G')]
        if 'Gempa' in terms_hash:
             dates_gempa.append(tweet['created_at'])
             count_all.update(terms_stop)
        word_freq = count_all.most_common(20)
```

```
        ones = [1]*len(dates_gempa)
# the index of the series
idx = pandas.DatetimeIndex(dates_gempa)
# the actual series (at series of 1s for the moment)
gempa = pandas.Series(ones, index=idx)

# Resampling / bucketing
per_minute = gempa.resample('1Min').sum().fillna(0)
time_chart = vincent.Line(gempa)
time_chart.axis_titles(x='Time', y='Freq')
time_chart.to_json('time_chart.json')
#print(count_all.most_common(5))
```

7. chart.html

```
<html>
<head>
    <title>Visualisasi Tweet Gempa Bumi</title>
    <script src="http://d3js.org/d3.v3.min.js"
charset="utf-8"></script>
    <script
src="http://d3js.org/topojson.v1.min.js"></script>
    <script
src="http://d3js.org/d3.geo.projection.v0.min.js"
charset="utf-8"></script>
    <script
src="http://trifacta.github.com/vega/vega.js"></script>
</head>
<body>
   Visualisasi Tweet Gempa Bumi
    <div id="vis"></div>
<script type="text/javascript">
// parse a spec and create a visualization view
function parse(spec) {
  vg.parse.spec(spec, function(chart) {
chart({el:"#vis"}).update(); });
}
parse("time_chart.json");
</script>
</body>
</html>
```