TUGAS AKHIR – IF184802

MEREPRESENTASIKAN MAKNA KATA UNTUK METODE KLASIFIKASI NAÏVE BAYES, RANDOM FOREST, DAN SUPPORT VECTOR MACHINE DALAM STUDI KASUS KEMACETAN DI SURABAYA

RAKHMA RUFAIDA HANUM
NRP 05111540000161

Dosen Pembimbing
Abdul Munif, S.Kom., M.Sc.
Nurul Fajrin Ariyani, S.Kom., M.Sc.

Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2019

[Halaman ini sengaja dikosongkan]

TUGAS AKHIR – IF184802

MEREPRESENTASIKAN MAKNA KATA UNTUK METODE KLASIFIKASI NAÏVE BAYES, RANDOM FOREST, DAN SUPPORT VECTOR MACHINE DALAM STUDI KASUS KEMACETAN DI SURABAYA

RAKHMA RUFAIDA HANUM
NRP 05111540000161

Dosen Pembimbing
Abdul Munif, S.Kom., M.Sc.
Nurul Fajrin Ariyani, S.Kom., M.Sc.

Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2019

[Halaman ini sengaja dikosongkan]

UNDERGRADUATE THESIS – IF184802

REPRESENTING THE MEANING OF WORD FOR NAÏVE BAYES, RANDOM FOREST AND SUPPORT VECTOR MACHINE CLASSIFICATION METHOD IN CASE STUDY TRAFFIC IN SURABAYA

RAKHMA RUFAIDA HANUM
NRP 05111540000161

Supervisors
Abdul Munif, S.Kom., M.Sc.
Nurul Fajrin Ariyani, S.Kom., M.Sc.

Department of Informatics
Faculty of Information and Communication Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2019

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

MEREPRESENTASIKAN MAKNA KATA UNTUK METODE KLASIFIKASI NAÏVE BAYES, RANDOM FOREST, DAN SUPPORT VECTOR MACHINE DALAM STUDI KASUS KEMACETAN DI SURABAYA

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Manajemen Informasi
Program Studi S-1 Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh:
RAKHMA RUFAIDA HANUM
NRP. 05111540000161

Disetujui oleh Pembimbing Tugas Akhir:

1. Abdul Munif., S.Kom., M.Sc. .....
NIP. 19860823 201504 1 004 (Pembimbing 1)

2. Nurul Fajrin A., S.Kom., M.Sc. .....
NIP. 19860722 201504 2 005 (Pembimbing 2)

SURABAYA
JANUARI, 2019

[Halaman ini sengaja dikosongkan]

MEREPRESENTASIKAN MAKNA KATA UNTUK METODE KLASIFIKASI NAÏVE BAYES, RANDOM FOREST, DAN SUPPORT VECTOR MACHINE DALAM STUDI KASUS KEMACETAN DI SURABAYA

Nama Mahasiswa	:	Rakhma Rufaida Hanum
NRP	:	05111540000161
Jurusan	:	Informatika, FTIK-ITS
Dosen Pembimbing 1	:	Abdul Munif, S.Kom., M.Sc.
Dosen Pembimbing 2	:	Nurul Fajrin Ariyani, S.Kom., M.Sc.

ABSTRAK

Twitter merupakan media sosial berbasis microblogging yang memungkinkan pengguna untuk memposting tulisan pendek yang dikenal dengan istilah tweet. Pengguna dapat menuliskan informasi kemacetan. Pada referensi Tugas Akhir sebelumnya, pembobotan kata untuk klasifikasi dilakukan dengan metode TF-IDF (Term Frequency-Inverse Document Frequency).

Dalam tugas akhir ini, dilakukan representasi makna kata ke dalam bentuk vektor menggunakan Word2vec. Pembobotan kata didapatkan dari hasil kedekatan vektor kata dengan vektor kata "macet" hasil dari training Word2vec. Semakin tinggi skor kedekatan, maka semakin mirip makna kedua kata tersebut. Klasifikasi tweet kemacetan di Surabaya dilakukan menggunakan dataset tweet yang berasal dari twitter. Data tersebut diambil dari akun @LMSurabaya dan akun @sits_dishubsby. Data tersebut akan diklasifikasi menggunakan metode klasifikasi Naïve Bayes, Random Forest dan Linear Support Vector Machine (SVM) menggunakan PySpark.

Dari hasil evaluasi didapatkan akurasi terbaik metode Random Forest yaitu 84.74%. Hasil akurasi yang didapat tidak lebih baik dibandingkan dengan TF-IDF sebesar 95.90%.

Kata kunci: klasifikasi, kemacetan, naïve bayes, random forest, svm, twitter, word2vec.

[Halaman ini sengaja dikosongkan]

REPRESENTING THE MEANING OF WORD FOR NAÏVE BAYES, RANDOM FOREST AND SUPPORT VECTOR MACHINE CLASSIFICATION METHOD IN CASE STUDY TRAFFIC IN SURABAYA

Student's Name : Rakhma Rufaida Hanum
Student's ID : 05111540000161
Department : Informatics, Faculty of ICT-ITS
Supervisor 1 : Abdul Munif, S.Kom., M.Sc.
Supervisor 2 : Nurul Fajrin Ariyani, S.Kom., M.Sc.

ABSTRACT

Twitter is a microblogging based social media that allows users to post short posts known as tweets. Users can write events around them, for example is traffic information. In the previous Final Project reference, the weighting of words for classification was calculated using the TF-IDF (Term Frequency-Inverse Document Frequency) method.

In this final project, representation of word meanings into vector shapes using Word2vec are performed. Term weighting obtained from the result of proximity of the word vector with the vector of word "macet" obtained from the result of Word2vec training. The higher the proximity score, the more similar the meaning of the two words. The classification of traffic jams in Surabaya is done by using a dataset of tweets originating from twitter.The data is taken from @LMSurabaya and @sits_dishubsby twitter account. The data will be classified using the Naïve Bayes, Random Forest and Linear Support Vector Machine (SVM) classification method using PySpark

After classification using Naive Bayes, Random Forest, and SVM is done, the result show that Random Forest got the best overall accuracy point, 84,74%. The result obtained is not as good as TF-IDF where 95,90% accuracy point obtained.

Keywords: classification, naïve bayes, random forest, traffic, twitter,svm, word2vec.

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Puji syukur saya sampaikan kepada Allah SWT yang Maha Kuasa karena berkat rahmat-Nya saya dapat melaksanakan Tugas Akhir yang berjudul:

"**MEREPRESENTASIKAN MAKNA KATA UNTUK METODE KLASIFIKASI NAÏVE BAYES, RANDOM FOREST, DAN SUPPORT VECTOR MACHINE DALAM STUDI KASUS KEMACETAN DI SURABAYA**"

Terselesaikannya Tugas Akhir ini tidak terlepas dari bantuan dan dukungan banyak pihak, Oleh karena itu melalui lembar ini penulis ingin mengucapkan terima kasih dan penghormatan kepada:

1. Allah SWT Yang Maha Kuasa, karena limpahan rahmat dan karunia-Nya penulis dapat menjalankan perkuliahan di Departemen Informatika Institut Teknologi Sepuluh Nopember dan dapat menyelesaikan Tugas Akhir guna memenuhi syarat kelulusan sebagai Sarjana.
2. Kedua orangtua penulis Papa, Mama, Kak Nana, Rizki dan Kak Alim yang telah memberikan dukungan doa, dan bentuk apapun kepada penulis sehingga penulis dapat menyelesaikan Tugas Akhir ini.
3. Bapak Abdul Munif, S.Kom., M.Sc. dan Ibu Nurul Fajrin Ariyani, S.Kom., M.Sc. selaku pembimbing I dan II yang telah membimbing, memberi nasihat, serta mengorbankan waktu dan tenaga untuk membimbing dalam menyelesaikan Tugas Akhir ini.
4. Dr. Eng. Darlis Herumurti, S.Kom., M.Kom. selaku Ketua Departemen Informatika ITS dan segenap dosen dan karyawan Departemen Informatika ITS yang telah memberikan ilmu dan pengalaman kepada penulis selama menjalani masa kuliah di Informatika ITS.

5. Purina Qurota Ayunin, teman seperjuangan tugas akhir yang selalu mendukung dalam menyelesaikan tugas akhir.
6. Anisa PD, teman Kerja Praktik yang berjuang dalam dunia kerja bersama.
7. A, Nahda, Hania, Rizka, Zakiya dan teman-teman yang tidak bisa disebutkan satu persatu.
8. Teman-teman Informatika angkatan 2015,teman-teman Lab Manajemen Informasi, BPH Schematics 2017, dan teman-teman Saman TC.
9. Serta semua pihak yang yang telah turut membantu penulis dalam menyelesaikan Tugas Akhir ini.

Penulis memohon maaf jika terdapat kesalahan maupun kekurangan dalam Tugas Akhir. Oleh karena itu dengan segala kerendahan hati penulis mengharapkan kritik dan saran yang membangun dapat disampaikan sebagai bahan perbaikan untuk kedepannya. Semoga laporan Tugas Akhir ini dapat berguna bagi pembaca.

Surabaya, Januari 2019

Rakhma Rufaida Hanum

DAFTAR ISI

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

DAFTAR KODE SUMBER

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

1.1 Latar Belakang

Kemacetan adalah kondisi dimana arus lalu lintas yang lewat pada ruas jalan yang ditinjau melebihi kapasitas rencana jalan tersebut mengakibatkan kecepatan bebas ruas jalan tersebut mendekati 0 km/jam atau bahkan menjadi 0 km/jam sehingga mengakibatkan terjadinya antrian (MKJI,1997). Kota Surabaya merupakan kota terbesar kedua di Indonesia setelah Jakarta, Ibu Kota Jawa Timur ini berada di posisi 8 dalam daftar kota termacet di Tanah Air [1].

Twitter merupakan media sosial berbasis *microblogging* yang memungkinkan pengguna untuk memposting tulisan pendek yang dikenal dengan istilah *tweet*. Pada *Twitter*, untuk membuat sebuah kabar menjadi viral atau *trending topic*, pengguna dapat memberi tanda *hashtag* di depan kata kunci yang diinginkan [2]. Pada *Twitter* terdapat fitur *follow*, yaitu pengguna dapat mengikuti halaman akun pengguna lain. Dengan fitur *follow* tersebut, pengguna dapat melihat *tweet* dari pengguna lain yang di *follow*. Pada *tweet* hanya dibatasi 280 karakter saja. Pengguna dapat menuliskan kejadian di sekitar lingkungan mereka, contohnya adalah informasi kemacetan. Dengan mudahnya pengarsipan ucapan dan ekspresi manusia, maka volume data teks akan bertambah seiring waktu [3]. Dengan banyaknya data *tweet*, maka diperlukan sebuah model klasifikasi untuk mengklasifikasikan informasi dari *tweet.* Pada referensi-referensi Tugas Akhir sebelumnya mengenai klasifikasi maupun *dataset* yang menggunakan data *tweet,* dapat disimpulkan kelebihan dari Tugas Akhir yang sudah ada yaitu hasil uji yang dilakukan menghasilkan tingkat akurasi yang tinggi, contohnya pada judul "Aplikasi Deteksi Kejadian di Jalan Raya berdasarkan Data Twitter Menggunakan Metode Support Vector Machine" oleh Vessa Rizky Oktavia. Pada Tugas Akhir tersebut evaluasi berupa akurasi paling baik yang didapatkan sebesar 96.25%, tetapi metode klasifikasi

yang digunakan hanya menggunakan *Support Vector Machine* (SVM). *Dataset* yang digunakan menggunakan *tweet* dari *Twitter* [4]. Proses pembobotan yang digunakan menggunakan algoritma TF-IDF yang sebelumnya dilakukan ekstraksi fitur. Kedua, tugas akhir dengan judul "Deteksi Gempa Berdasarkan Data Twitter Menggunakan *Decision Tree*, *Random Forest*, dan SVM" oleh Rendra Dwi Lingga. Pada Tugas Akhir tersebut melakukan klasifikasi deteksi gempa dengan dataset *tweet* dengan menggunakan beberapa *classifier* yaitu *Decision Tree*, *Random Forest* dan SVM. Sebelum dilakukan klasifikasi, dataset dilakukan pembobotan menggunakan *Term-Frequency* (TF) yang merupakan *frequency* kemunculan *term* atau kata dalam dokumen. Dengan menggunakan beberapa metode maka evaluasi hasil akhirnya adalah membandingkan evaluasi setiap metode dan menyimpulkan metode mana yang memiliki *recall* yang tertinggi untuk studi kasus deteksi gempa. Dari kedua tugas akhir tersebut, pembobotan dilakukan menggunakan TF-IDF dan TF. Kedua tugas akhir tersebut belum melakukan pemaknaan suatu kata (merepresentasikan makna suatu kata) karena hanya melakukan pembobotan *frequency* kemunculan suatu kata. Dalam merepresentasikan suatu makna kata, dapat dilakukan menggunakan *word vector representation*.

Oleh karena itu dalam tugas akhir ini yang akan dilakukan yaitu bagaimana caranya mempresentasikan suatu makna kata menjadi vektor dan mengimplementasikan *word vector representation* dan memberikan hasil uji pada metode klasifikasi *Naïve Bayes*, *Random Forest* dan *Support Vector Machine* (SVM) berupa akurasi. *Dataset* untuk klasifikasi berupa *tweet* dari media sosial *Twitter*. Model klasifikasi merupakan metode *data mining* yang mengelompokkan data sesuai label kelas yang sudah ditentukan. Klasifikasi yang akan dilakukan adalah klasifikasi *tweet* dengan dataset yang diambil dari akun yang memiliki *username* @LMSurabaya dan @sits_dishubsby yang diharapkan akan mengelompokkan *tweet* yang memberikan informasi tentang keadaan lalu lintas di Surabaya. Representasi kata-kata

terdistribusi dalam ruang vektor diharapkan dapat membantu *learning algortihms* untuk mendapatkan hasil yang lebih baik dalam *natural language processing* dengan mengelompokkan kata-kata yang serupa. Salah satu *word vector representation* adalah *Word2vec*. *Word2vec* salah satu *word embedding* yang dipublikasikan pertama kali oleh Mikolov et al [5]. Vektor yang dihasilkan diharapkan bisa mewakili makna dari sebuah kata. Tugas akhir ini akan memanfaatkan *word vector representation* yaitu *Word2vec*. Selain itu untuk klasifikasi menggunakan beberapa metode *classifier*. Setelah melakukan klasifikasi data tersebut, maka akan didapatkan informasi mengenai klasifikasi *tweet* kemacetan.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam tugas akhir ini adalah sebagai berikut:

1. Bagaimana cara mendapatkan representasi makna kata yang baik?
2. Apakah *Word2vec* dapat memberikan hasil yang lebih baik untuk metode klasifikasi *Naïve Bayes*, *Random Forest*, dan *Support Vector Machine* pada studi kasus kemacetan?

1.3 Batasan Permasalahan

Batasan masalah pada tugas akhir ini antara lain:

1. Bahasa yang digunakan pada dataset adalah Bahasa Indonesia.
2. Tidak bisa menangani kata-kata yang tidak ada dalam *corpus.*
3. *Dataset* yang digunakan berasal dari data *tweet* dari media sosial *Twitter* dengan mengambil tweet akun pengguna @LMSurabaya, dan @sits_dishubsby.
4. Proses klasifikasi dilakukan menggunakan metode klasifikasi *Naïve Bayes*, *Random Forest* dan *Linear Support Vector Machine* menggunakan *PySpark*.

1.4 Tujuan

Tujuan dari pembuatan tugas akhir ini adalah untuk merepresentasikan makna kata untuk metode klasifikasi dalam studi kasus kemacetan lalu lintas di Surabaya. Dimana dengan merepresentasikan makna kata dimungkinkan akan memberikan hasil uji yang lebih baik dalam kasus klasifikasi.

1.5 Manfaat

Manfaat dari pembuatan tugas akhir ini antara lain:

1. Mengetahui salah satu metode untuk merepresentasikan makna dari suatu kata.
2. Mengetahui hasil dari metode klasifikasi dengan menggunakan *Word2vec*.

1.6 Metodologi

Langkah-langkah yang harus ditempuh dalam pengerjaan tugas akhir ini adalah sebagai berikut:

1. Penyusunan proposal tugas akhir

Proposal tugas akhir ini berisi tentang pendahuluan tugas akhir yang akan dibuat. Pendahuluan tugas akhir berisi deskripsi tugas akhir yang meliputi latar belakang dalam pembuatan tugas akhir, rumusan masalah yang akan diangkat dalam tugas akhir, batasan masalah dalam pembuatan tugas akhir, tujuan dari dibuatnya tugas akhir, dan manfaat dari hasil pembuatan tugas akhir. Selain yang telah disebutkan, pendahuluan tugas akhir ini juga menjabarkan tinjauan pustaka yang dijadikan referensi atau pendukung dalam pembuatan tugas akhir.

2. Studi Literatur

Pada studi literatur tugas akhir ini telah dipelajari beberapa referensi yang diperlukan antara lain mengenai *Text Mining*, *Word2vec, Text Preprocessing, classifier Naïve Bayes, classifier Random Forest* dan *classifier Support Vector Machine (SVM)*.

3. Analisis dan desain perangkat lunak

Pada implementasi dalam mempresentasikan makna dari suatu kata, maka akan dihasilkan suatu makna kata dalam bentuk vektor. Selain itu juga akan dijabarkan hasil uji dari penggunaan *Word2vec* untuk metode klasifikasi *Naïve Bayes, Random Forest* dan *Support Vector Machine* (SVM).

4. Implementasi perangkat lunak

Perangkat keras yang digunakan adalah perangkat keras yang mendukung *graphical processing unit* (GPU) NVIDIA GeForce. Sementara, sistem operasi yang digunakan adalah Windows 10. Pada tugas akhir kali ini, perangkat keras dan sistem operasi yang digunakan mempunyai spesifikasi sebagai berikut:

- Intel Core i5
- NVIDIA GeForce
- 8GB of RAM
- Windows 10

Sedangkan untuk *environment* perangkat lunak, digunakan bahasa pemrograman Python 3.6.4 dengan berbagai macam *library* pendukungnya. Semua *library* tersebut digunakan untuk melakukan manipulasi dan rekayasa data-data mentah sehingga memenuhi kualifikasi untuk menjadi masukan pada *library* utama.

Pada tugas akhir ini, kode *Word2vec* tidak akan ditulis dari awal, melainkan menggunakan *source code Word2vec* yang dipublikasikan secara *open source* di alamat: http://textminingonline.com/training-word2vec-model-on-english-wikipedia-by-gensim dan https://yudiwbs.wordpress.com/2018/03/31/word2vec-wikipedia-bahasa-indonesia-dengan-python-gensim.

Masukan untuk metode *Word2vec* adalah hasil *dump* dari artikel-artikel dari Wikipedia Bahasa Indonesia yang dapat diakses melalui alamat: https://dumps.wikimedia.org/idwiki/latest dan data yang diambil adalah yang terformat XML.

Sebelum melakukan *modelling* vektor kata menggunakan *Word2vec*, terlebih dahulu dilakukan langkah *preprocessing* yang

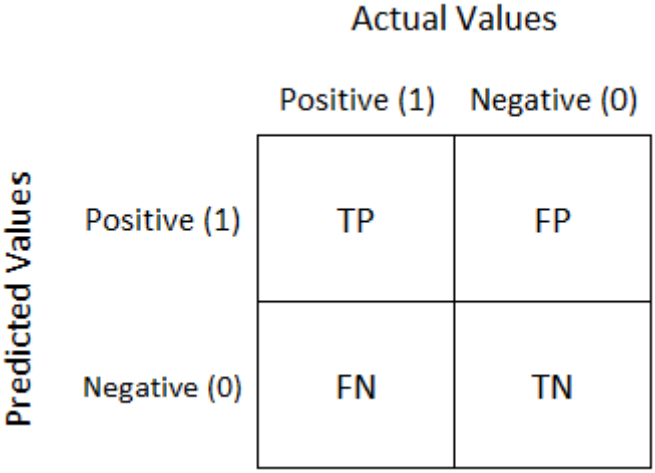
melibatkan *library* Gensim. Instalasi cukup dengan mengetikkan `pip install gensim`.

Lalu, dalam mengaplikasikan metode *klasifikasi Naïve Bayes*, *Random Forest*, dan *Support Vector Machine* menggunakan *Spark*.

5. Pengujian dan Evaluasi

Pengujian dan evaluasi pada tugas akhir ini dilakukan dengan metode klasifikasi *Naïve Bayes*, *Random Forest*, dan *Support Vector Machine* (SVM) dengan menggunakan *Spark*. Evaluasi hasil uji berupa *accuracy*, *precision*, *recall* dan *F-Measure*.

Terdapat istilah-istilah yang perlu diketahui dalam menentukan rumus-rumus evaluasi dengan menggunakan *confusion matrix* yang dijelaskan pada Gambar 1-1 sebagai berikut:



Gambar 1-1 *Confusion Matrix*

Dimana TP adalah *True Positive*, yaitu jumlah kebenaran antara hasil klasifikasi dengan jumlah seluruh data. TN adalah *True Negative*, yaitu jumlah hasil klasifikasi yang diduga benar, tetapi sebenarnya salah. FP adalah *False Positive*, yaitu jumlah hasil

klasifikasi yang diduga salah, tetapi sebenarnya benar. Dan FN adalah *False Negative*, yaitu jumlah dari kesamaan hasil klasifikasi dan sebenarnya adalah salah.

Penjelasan evaluasi *accuracy*, *precision*, *recall*, dan *F-Measure* yang lebih rinci dijabarkan sebagai berikut:

a. *Accuracy*

Akurasi adalah nilai rasio data yang diklasifikasikan benar dari jumlah total data. Perumusan akurasi dapat dituliskan dengan rumus:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}. \quad (1.1)$$

b. *Precision*

Precision atau presisi adalah nilai total data positif yang diklasifikasikan dengan benar dibagi dengan hasil prediksi data positif. Perumusan presisi dapat dituliskan dengan rumus:

$$precision = \frac{TP}{TP + FP}. \quad (1.2)$$

c. *Recall*

Recall adalah nilai rasio dari jumlah total data positif yang diklasifikasikan dengan benar diagi dengan jumlah data positif. Perumusan *recall* dapat dituliskan dengan rumus:

$$recall = \frac{TP}{TP + FN}. \quad (1.3)$$

d. F-*Measure*

F-Measure adalah nilai yang diperoleh dari *recall* dan *precision* yang menggunakan *harmonic mean*. Perumusan *F-Measure* dapat dituliskan dengan rumus:

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}. \quad (1.4)$$

Evaluasi dilakukan dengan membagi data menjadi data *training* dan data *testing*.

6. Penyusunan buku tugas akhir

Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam tugas akhir ini serta hasil dari implementasi metode yang telah dibuat. Sistematika penulisan buku tugas akhir ini secara garis besar antara lain:

1. Pendahuluan
    a. Latar Belakang
    b. Rumusan Masalah
    c. Batasan Tugas Akhir
    d. Tujuan
    e. Manfaat
    f. Metodologi
    g. Sistematika Penulisan
2. Tinjauan Pustaka
3. Desain dan Implementasi
4. Pengujian dan Evaluasi
5. Kesimpulan dan Saran
6. Daftar Pustaka

1.7 *Sistematika Penulisan Laporan*

Buku tugas akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan tugas akhir. Selain itu, diharapkan dapat berguna untuk pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku tugas akhir terdiri atas beberapa bagian seperti berikut ini:

BAB I. Pendahuluan

Bab ini berisi latar belakang masalah, tujuan dan manfaat pembuatan tugas akhir, permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penyusunan tugas akhir.

BAB II. Dasar Teori

Bab ini menjelaskan beberapa teori yang dijadikan penunjang dan berhubungan dengan pokok

pembahasan yang mendasari pembuatan tugas akhir.

BAB III. Analisis dan Perancangan Sistem

Bab ini membahas mengenai perancangan sistem yang akan dibangun. Perancangan sistem meliputi perancangan data dan alur proses dari sistem itu sendiri.

BAB IV. Implementasi

Bab ini berisi implementasi dari perancangan sistem yang telah ditentukan sebelumnya.

BAB V. Pengujian dan Evaluasi

Bab ini membahas pengujian dari metode yang ditawarkan dalam tugas akhir untuk mengetahui kesesuaian metode dengan data yang ada.

BAB VI. Kesimpulan

Bab ini berisi kesimpulan dari hasil pengujian yang telah dilakukan. Bab ini juga membahas saran-saran untuk pengembangan sistem lebih lanjut.

Daftar Pustaka

Merupakan daftar referensi yang digunakan untuk mengembangkan tugas akhir.

Lampiran

Merupakan bab tambahan yang berisi data atau daftar istilah yang penting pada tugas akhir ini.

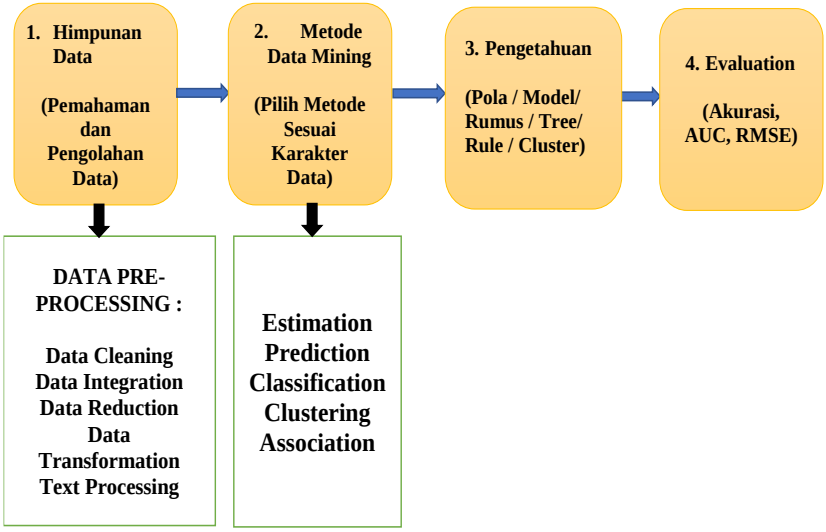
[Halaman ini sengaja dikosongkan]

BAB II
DASAR TEORI

Pada bab ini akan dibahas mengenai teori-teori yang menjadi dasar tugas akhir ini.

2.1 *Text Mining*

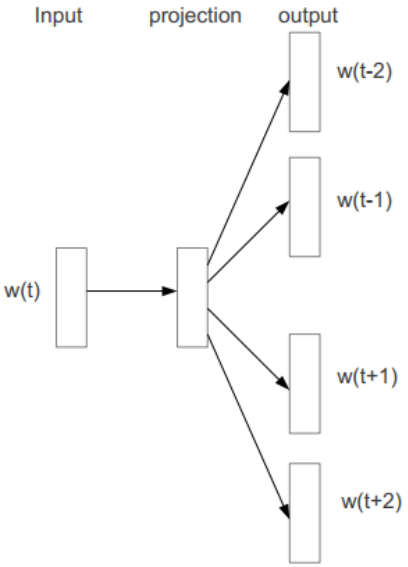
Text Mining memiliki definisi menambang data yang berupa teks dimana sumber data biasanya didapatkan dari dokumen, dan tujuannya adalah mencari kata-kata yang dapat mewakili isi dari dokumen sehingga dapat dilakukan analisa keterhubungan antar dokumen. *Text Mining* merupakan penerapan konsep dan teknik *data mining* untuk mencari pola dalam teks, yaitu proses penganalisisan teks guna menemukan informasi yang bermanfaat untuk tujuan tertentu [6]. Gambar 2-1 dibawah ini menggambarkan proses data mining:



Gambar 2-1 Proses *Data Mining* [6]

2.2 *Word2vec*

Word2vec merupakan salah satu teknik *word embedding* yang dipublikasikan pertama kali oleh Mikolov et al [7]. Sebagai gambaran bahwa vektor dari *Word2vec* bisa mewakili makna dari sebuah kata dan dapat mengukur beberapa vektor sebagai perbandingan. Gagasan utama *Word2vec* adalah memprediksi antara setiap kata dan kata konteksnya. *Word2vec* menggunakan 2 algoritma yaitu algoritma *Skip-gram* dan *Continuous Bag of Word* (CBOW) [8] .



Gambar 2-2 Arsitektur Model *skip-gram* [7]

Algoritma *Skip-gram* untuk memprediksi kata konteks yang diberikan target (*position independent*) dan memaksimalkan rata-rata *log probability*, persamaannya yaitu:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0} log\ p(w_{t+j}|w_t) \quad (2.1)$$

dimana ‘w’ adalah *word*, ‘c’ adalah *size of the training context* (yang dapat menjadi fungsi dari *center word wt*). Lebih besar c maka menghasilkan lebih banyak *training example* dan dengan demikian dapat menghasilkan akurasi yang lebih tinggi dengan mengorbankan *training time* [7]. Formulasi dasar *skip-gram* mendefinisikan

$$p(w_{t+j}|w_t) \quad (2.2)$$

menggunakan *softmax function* untuk membuat probabilitas yang dihitung akan berada dikisaran 0 sampai 1 dan jumlah probabilitas sama dengan 1:

$$p(w_o|w_t) = \frac{exp(v'_{wo^\top} v'_{wI})}{\sum_{w=1}^{W} exp(v'_{wo^\top} v'_{wI})} \quad (2.3)$$

dimana ‘v`w’ dan ‘v`w ‘ adalah *input* dan *output* representasi vektor dari ‘w’, ‘W’ adalah jumlah kata dalam kosakata.
Sebagai gambaran, vektor dari *Word2vec* bisa mewakili makna dari sebuah kata, kita bisa mengukur beberapa vektor sebagai perbandingan. Apabila kita mengukur jarak antara kata “France” dengan “Paris” maka akan ditemukan bahwa jarak yang akan muncul pada angka yang berdekatan. Pertama yang dilakukan adalah menentukan *corpus* untuk ditraining kedalam *Word2vec*, setelah itu *Word2vec* dimodelkan dengan hasil yaitu setiap kata dalam *corpus* dijadikan vektor. Berikut adalah contoh jika *Word2vec* sudah dimodelkan dan kita ingin melihat kesamaan pada kata “Soekarno”, hasil yang muncul ditunjukkan pada Gambar 2-3 dibawah ini :

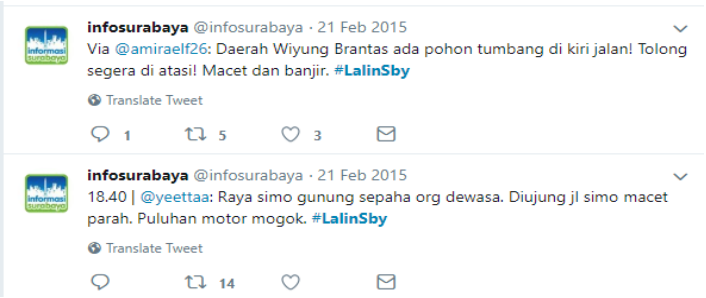
```
Soekarno:[('Sukarno', 0.813288152217865), ('Soeharto',
0.7391608953475952), ('Megawati', 0.6650642156600952),
('Suharto', 0.6611993908882141), ('Hatta',
0.6327983736991882), ('SBY', 0.6301325559616089),
('Bung', 0.6262293457984924), ('Jokowi',
0.6140671968460083), ('Yudhoyono', 0.5906702876091003),
('Karno', 0.5696855187416077)]
```

Gambar 2-3 Contoh hasil kedekatan kata Word2vec

Pada hasil yang digambarkan yaitu menampilkan vektor-vektor yang berdekatan dengan vektor kata "Soekarno". Vektor-vektor yang dihasilkan merupakan *output* dari proses *natural language processing*.

2.3 *Dataset*

Dataset merupakan kumpulan *item-item* yang berhubungan dan terpisah dari data yang dapat diakses secara individual atau dalam kombinasi atau dikelola sebagai satu kesatuan utuh [9]. *Dataset* yang digunakan pada tugas akhir ini diambil dari *tweet* pada Twitter. *Tweet* merupakan catatan singkat dimana pengguna dapat tuliskan tentang apa yang terjadi di lingkungan sekitarnya. *Dataset twitter* yang akan digunakan pada tugas akhir akan dilakukan pelabelan secara manual dan akan diuji akurasinya menggunakan metode klasifikasi *Naïve Bayes*, *Random Forest* dan *Support Vector Machine*. Berikut adalah contoh *tweet* tentang kemacetan:



Gambar 2-4 Contoh *tweet* macet

2.4 *Text Preprocessing*

Text Preprocessing merupakan sebuah tahap dimana sistem melakukan seleksi data yang akan diproses pada dokumen. Pada proses *preprocessing* ini meliputi *case folding*, *tokenizing*, *filtering*, *stemming* [10]. *Preprocessing* merupakan tahap awal yang bertujuan untuk mempersiapkan agar data teks dapat diubah menjadi lebih terstruktur dan memastikan data yang akan diolah di *data mining* adalah data yang terstruktur dan baik. Berikut merupakan penjelasan tahap-tahap *preprocessing*:

a. *Case Folding*

Case Folding merupakan proses penyamaan *case* (semua huruf) dalam dokumen menjadi huruf kecil.

b. Tokenizing

Tokenizing merupakan proses pemotongan *string input* berdasarkan tiap kata yang menyusunnya.

c. Stemming

Stemming merupakan proses mengubah suatu kata menjadi kata dasar.

2.5 *POS Tagging*

POS Tagging merupakan cara pengkategorikan kelas kata. Contoh kelas kata yang dikategorikan yaitu, kata benda, kata sifat, kata kerja, dan lain-lain. Berikut merupakan macam-macam kelas kata [11]:

1. Kata Benda (Nomina)

   Kata benda (nomina) adalah kata-kata yang merujuk pada bentuk suatu benda. Bentuk benda dapat bersifat abstrak maupun konkret.

2. Kata Kerja (Verba)

   Kata kerja atau verba adalah jenis kata yang menyatakan suatu perbuatan.

3. Kata Sifat (Adjektiva)

   Kata sifat adalah kelompok kata yang mampu menjelaskan atau mengubah kata benda atau kata ganti menjadi lebih spesifik. Selain itu, kata sifat mampu menerangkan kuantitas dan kualitas dari kelompok kelas kata benda atau kata ganti.

4. Kata Ganti (Pronomia)

   Kelompok kata ini dipakai untuk menggantikan benda atau sesuatu yang dibendakan.

5. Kata Keterangan (Adverbia)

   Kata keterangan adalah jenis kata yang memberikan keterangan pada kata kerja, kata sifat, dan kata bilangan, bahkan mampu memberikan keterangan pada seluruh kalimat.

6. Kata Bilangan (Numeralia)

   Kata bilangan adalah jenis kelompok kata yang menyatakan jumlah, kumpulan, dan urutan sesuatu yang dibendakan.

7. Kata Tugas

Kata tugas merupakan kata yang memiliki arti gramatikal dan tidak memiliki arti leksikal. Dari segi bentuk umumnya, kata-kata tugas sukar mengalami perubahan bentuk, seperti kata dengan, telah, dan, tetapi. Namun, ada sebagian yang dapat mengalami perubahan golongan kata, tetapi jumlahnya sangat terbatas, seperti kata tidak dan kata sudah. Meskipun demikian, kedua kata tersebut dapat mengalami perubahan menjadi menidakkan dan menyudahkan.

2.6 *Naïve Bayes*

Naïve Bayes merupakan sebuah metode klasifikasi menggunakan metode probabilitas dan statistik. Algoritma *Naïve Bayes* memprediksi peluang di masa depan berdasarkan pengalaman di masa sebelumnya sehingga dikenal sebagai *Teorema Bayes*. Ciri utama dari *Naïve Bayes* adalah asumsi yang sangat kuat akan indepedensi dari masing-masing kondisi atau kejadian. *Teorema Bayes* memiliki tingkat akurasi yang tinggi dan kecepatan yang baik ketika diterapkan pada *database* yang besar [12]. Persamaan 2.4 merupakan model dari *Naïve Bayes* yang selanjutnya akan digunakan dalam proses klasifikasi.

$$P(A\,|B) = (P(B|A) * P(A)/P(B) \tag{2.4}$$

Persamaan 2.5 merupakan penjabaran dari persamaan 2.4

$$P(C_I\,|D) = (P(D|C_I) * P(C_I))/P(D) \tag{2.5}$$

Multinomial Naïve Bayes adalah model penyederhanaan dari metode *Bayes* yang cocok dalam klasifikasi teks atau dokumen. Persamaannya sebagai berikut:

$$V_{MAP} = \arg\max P\big(V_j \,\big|a_1, a_2, \ldots\ldots, a_n\big) \tag{2.6}$$

Menurut persamaan (2.6), maka persamaan (2.4) dapat ditulis:

$$V_{MAP} = \underset{v_j \in V}{\arg max}\, \frac{P(a_1,a_2,\ldots\ldots,a_n \mid v_j)P(v_j)}{P(a_1,a_2,\ldots\ldots,a_n)} \tag{2.7}$$

P(a_1, a_2, …., a_n) konstan, sehingga dapat dihilangkan menjadi

$$V_{MAP} = \underset{v_j \in V}{\arg max}\, P(a_1, a_2, \ldots\ldots, a_n \mid v_j)P(v_j) \tag{2.8}$$

Karena $P(a_1, a_2, \ldots\ldots, a_n \mid v_j)P(v_j)$ sulit untuk dihitung, maka akan diasumsikan bahwa setiap kata pada dokumen tidak mempunyai keterkaitan.

$$V_{MAP} = \underset{v_j \in V}{\arg max}\, P(v_j) \prod i\, P(a_i|v_j) \quad (2.9)$$

Keterangan:

$$P(a_i|v_i) = \frac{|\, docs_j \,|}{contoh} \quad (2.10)$$

$$P(w_k|v_j) = \frac{n_k+1}{n+|kosakata|} \quad (2.11)$$

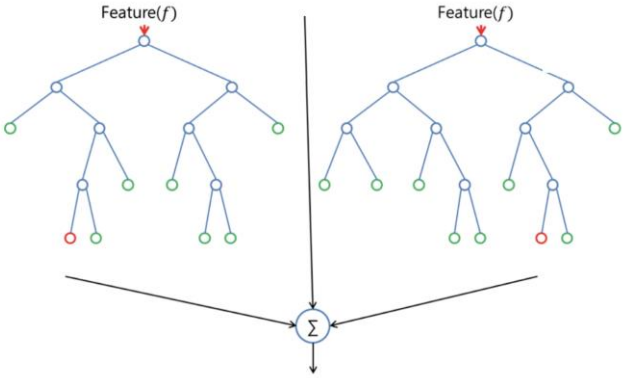
Dimana '$P(v_j)$' adalah probabilitas setiap dokumen terhadap sekumpulan dokumen, '$P(w_k|v_j)$' adalah probabilitas kemunculan kata w_k pada suatu dokumen dengan kategori class v_j , '| doc |' adalah frekuensi dokumen pada setiap kategori, '| contoh |' adalah jumlah dokumen yang ada, 'N_k' adalah frekuensi kata ke-k pada setiap kategori, 'kosakata' adalah jumlah kata pada dokumen *test* [13].

Naïve Bayes memiliki tingkat akurasi yang tinggi dan kecepatan yang baik ketika diterapkan pada *database* yang besar [12]. Selain itu *Naïve Bayes* merupakan salah satu algoritma yang sederhana tetapi memiliki kemampuan dan akurasi tinggi. Jadi, dapat dimungkinkan bahwa dengan mengklasifikasikan data kemacetan yang besar dapat menghasilkan akurasi yang baik.

2.7 *Random Forest*

Random Forest merupakan *supervised learning algorithm*, 'forest' yang dibangun adalah *ensemble decision tree*. *Random Forest* membangun banyak *decision tree* dan menggabungkannya untuk mendapatkan prediksi yang lebih akurat dan stabil [12]. *Random Forest* bekerja dengan cara membangun lebih dari satu *Decision Tree* secara *random* saat *training*. Hasil yang diberikan oleh *Random Forest* untuk klasifikasi adalah modus dari *decision tree* nya. Sementara nilai yang diberikan untuk regresi adalah

mean. Di bawah ini merupakan bagaimana tampilan *Random Forest* dengan dua *tree*:



Gambar 2-5 *Random Forest* dengan dua *tree* [14]

Formula *Random Forest* adalah sebagai berikut:

$$P(c|f) \quad = \sum_{t=1}^{T} \mathrm{Pt(c|f)} \tag{2.12}$$

Dimana 'P(c|f)' adalah klasifikasi *final test*, 'Pt(c|f)' adalah klasifikasi pada setiap *tree*, dan 'T' adalah jumlah *tree*.

2.8 *Support Vector Machine*

Support Vector Machine (SVM) adalah salah satu algoritma *supervised machine learning* yang dapat digunakan baik itu dalam masalah *klasifikasi* dan regresi. Dalam SVM terdapat istilah *support vector* yang merupakan titik terdekat dengan *hyperplane*. *Hyperplane* dapat dimisalkan sebuah garis yang memisahkan dan

mengklasifikasikan secara linier satu set data [15]. Tujuan SVM adalah untuk mendapatkan *optimal hyperplane* (*boundary*) yang memisahkan dua kelas yang berbeda.

Pada *Linear Support Vector Machine* dimisalkan data *training* direpresentasikan dengan

$$x_1, y_1, \dots, \left(x_k, y_k\right), x \in R^n, y \in \{+1, -1\} \quad (2.13)$$

Pemisah antar kelas dipisahkan oleh *hyperplane*

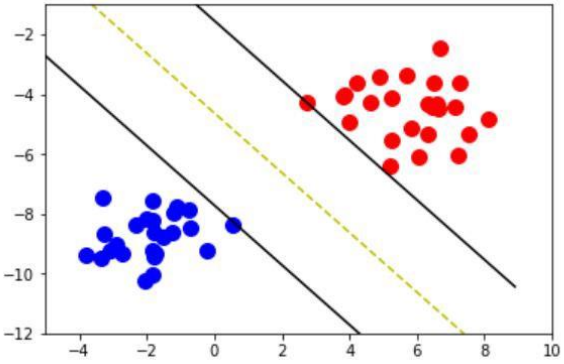
$$x_1, y_1 - b = 0 \quad (2.14)$$

SVM digunakan untuk menemukan fungsi pemisah yang optimal yang bisa memisahkan dua set data dari dua kelas yang berbeda dan menghasilkan *hyperplane* dengan *margin* terbesar, yaitu dirumuskan sebagai berikut.

$$y_i[(w \cdot x_i) - b] \geq 1. i = 1,2, \dots, l \quad (2.15)$$

Dimana ‘x’ adalah titik data berada, ‘y’ adalah kelas data (-1 atau +1), ‘w’ adalah vektor bobot, ‘b’ adalah skalar yang digunakan sebagai bias, ‘l’ adalah banyaknya data, ‘n’ adalah dimensi data, dan ‘R’ adalah ruang dimensi data.

Pada studi kasus yang akan diuji memiliki dua kelas yang berbeda yaitu macet dan tidak. Contoh *hyperplane* pada SVM ditunjukkan pada Gambar 2-6.



Gambar 2-6 *Hyperplane* pada SVM

2.9 *Tweepy*

Untuk berinteraksi dengan API Twitter, diperlukan *python client* yang mengimplementasikan panggilan berbeda ke API itu sendiri. Bagian pertama untuk dapat berinteraksi dengan API Twitter yaitu melibatkan autentikasi yaitu berupa *consumer key*, *consumen secret*, *access token* dan *access token secret* [16].

Fungsi `get_twitter_auth()` berfungsi untuk autentikasi. Fungsi `get_twitter_client()` berfungsi untuk membuat *instance* dari tweepy.API, digunakan untuk berbagai jenis interaksi dengan Twitter. Pada tugas akhir ini, *tweepy* digunakan untuk mengambil data *tweet* dari suatu akun. Untuk mengambil data tersebut digunakan fungsi `tweepy.Cursor` untuk *loop home timeline* untuk 10 *tweet* pertama. Dengan `Cursor(client.home_timeline)` dapat mengambil data dari *home timeline* seseorang. Fungsi tersebut juga dapat mengatur jumlah *tweet* yang dapat diambil. Data yang diambil berbentuk *file jsonlines* [16].

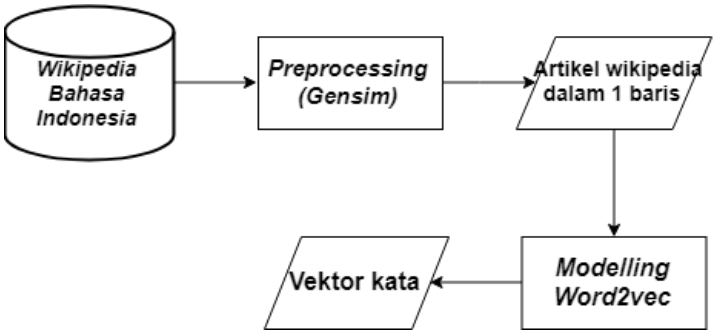
[Halaman ini sengaja dikosongkan]

BAB III
ANALISIS DAN PERANCANGAN SISTEM

Pada bab ini akan dijelaskan mengenai analisis dan perancangan sistem tugas akhir yang meliputi tahap perancangan data dan perancangan proses. Bab ini juga menjelaskan tentang analisis implementasi metode secara umum pada sistem.

3.1 Analisis Metode Secara Umum

Pada tugas akhir ini akan dibangun suatu sistem yang dapat mengklasifikasikan kemacetan di Surabaya dengan menggunakan *tweet* dari media sosial *Twitter*. Proses-proses yang dilakukan dalam pengimplementasian sistem ini meliputi membangun *Word2vec* dengan *corpus* yang berasal dari Wikipedia Bahasa Indonesia. Diagram alir tahapan membangun *Word2vec* ditunjukkan oleh Gambar 3-1 sebagai berikut:



Gambar 3-1 Diagram alir membangun *Word2vec*

Word2vec dibangun dengan menggunakan korpus yang berasal dari data Wikipedia Bahasa Indonesia. Tahap untuk membangun *Word2vec* yaitu dilakukan tahap *preprocessing* terhadap data

Wikipedia Bahasa Indonesia dan setelah itu dilakukan pemodelan vektor.

Pada Gambar 3-2, *dataset* untuk tugas akhir ini berasal dari *tweet* yang didapat dari media sosial *Twitter* menggunakan library *Tweepy*. Tahap pertama setelah mendapatkan *tweet* tersebut adalah melakukan pelabelan *tweet* dan selanjutnya *preprocessing tweet* tersebut. Setelah itu melakukan *sentence mapping* yaitu mencari bobot *tweet* tersebut. Selanjutnya, yaitu tahap melakukan klasifikasi menggunakan metode *Naïve Bayes*, *Random Forest*, dan *Support Vector Machine* menggunakan *Spark*. Dan yang terakhir adalah tahap *visualisasi*. Visualisasi yang dilakukan adalah menampilkan lokasi mana saja yang dapat dideteksi menggunakan API Google Maps. Pengguna dapat melakukan pemfilteran waktu berupa rentang tanggal dan jam. Dengan begitu, pengguna mendapat informasi lokasi berdasarkan waktu yang diinputkan pengguna.

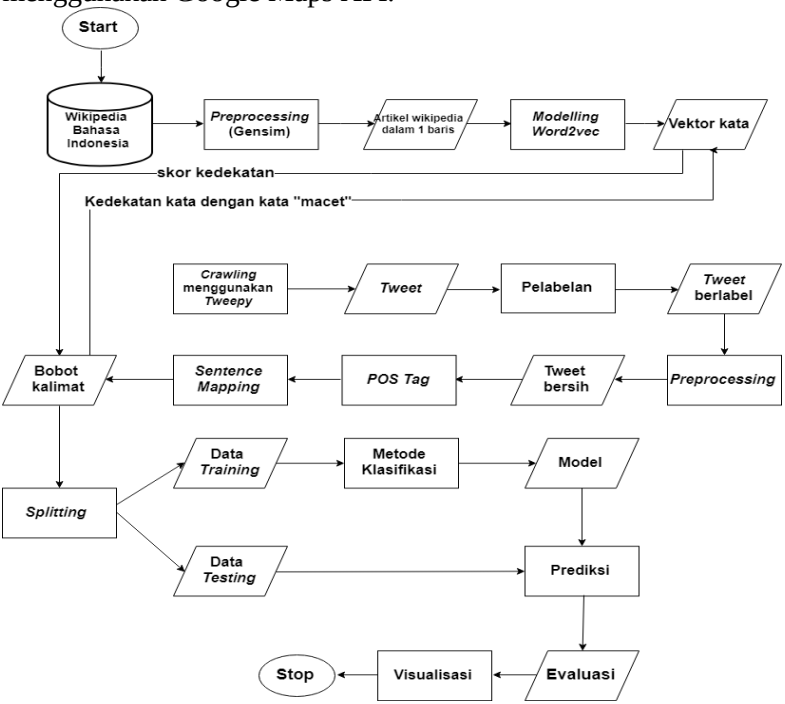
Tahap *preprocessing* yaitu tahap dimana data *tweet* hasil *crawling* dari media sosial *Twitter* dilakukan proses pembersihan data. Hasil dari tahap *preprocessing* tersebut berupa kata-kata yang diperlukan dalam tahap *sentence mapping*.

Tahap *POS Tagging* dan *sentence mapping* dilakukan pada hasil *preprocessing tweet* yang sebelumnya sudah dilakukan. Tahapan ini hanya akan diambil kata kerja (*verb*), kata sifat (*adjective*), dan kata keterangan (*adverb*) saja untuk dilakukan *sentence mapping*. *Sentence mapping* yang dilakukan adalah untuk mencari bobot kalimat. Bobot kalimat didapatkan dari hasil rata-rata kedekatan kata (*most similar*) dengan kata "macet". *POS Tagging* merupakan suatu cara untuk mengkategorikan kelas kata, seperti kata benda, kata kerja, dan lain-lain.

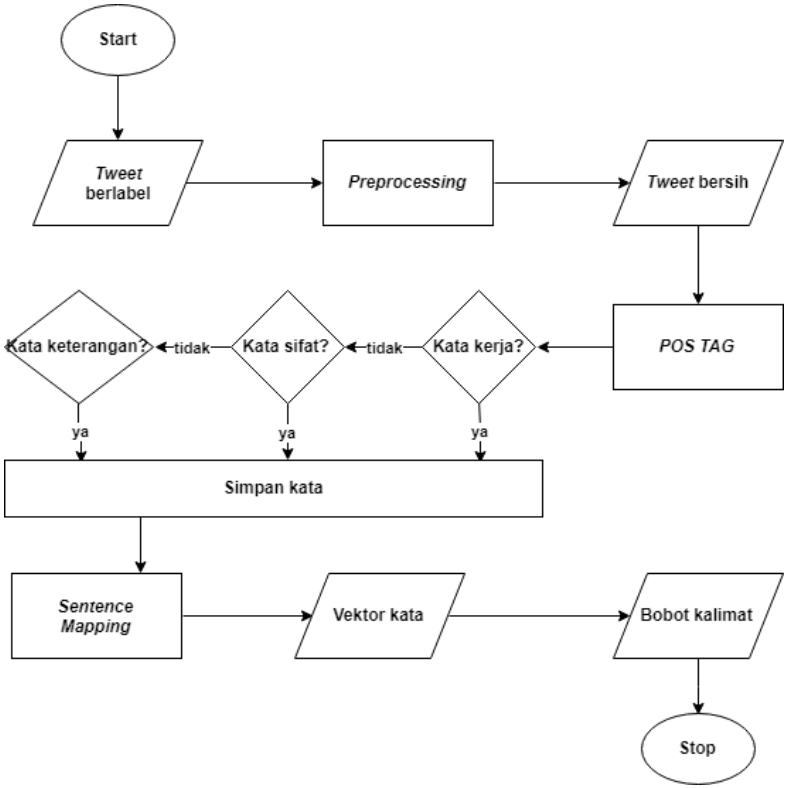
Pada Gambar 3-3, yang dimaksud *sentence mapping* adalah mencari bobot kata dari hasil kata kerja, kata sifat, dan kata keterangan pada *tweet* yang telah dikategorikan kelas katanya dan kemudian akan dicari bobot kedekatan dengan kata "macet" dengan inputan dari hasil *preprocessing tweet* dan hasilnya dirata-ratakan.

Tahap klasifikasi menggunakan metode *Naïve Bayes, Random Forest*, dan *Support Vector Machine*. Fitur-fitur dari model klasifikasi yang digunakan yaitu bobot *tweet* dan label. Evaluasi dari hasil uji klasifikasi berupa *accuracy, precision, recall*, dan *F-Measure*.

Tahap visualiasi dilakukan dengan menentukan lokasi dari suatu *tweet*. Setiap *tweet* tersebut akan dilakukan deteksi lokasi. Setelah lokasi dideteksi dan disimpan, maka akan dilakukan pencarian *latitude* dan *longitude* setiap lokasi tersebut menggunakan Geocoding API. Setelah *latitude* dan *longitude* didapatkan, maka akan disimpan untuk tahap *marker* lokasi dengan menggunakan Google Maps API.



Gambar 3-2 Diagram alir tugas akhir



Gambar 3-3 Diagram alir *POS Tag & Sentence Mapping*

3.2 Perancangan Data

Pada subbab ini akan menjelaskan proses perancangan data. Data yang digunakan adalah data *tweet* dari media sosial *Twitter* yaitu *tweet* yang berasal dari akun @LMSurabaya dan akun @sits_dishubsby. Alasan pemilihan dari 2 akun tersebut dikarenakan akun tersebut membagikan *tweet* mengenai kondisi lalu lintas di Kota Surabaya. Akun-akun tersebut tidak hanya memberikan informasi mengenai kondisi lalu lintas Surabaya,

tetapi juga melakukan *retweet* terhadap *tweet* dari akun manapun yang membagikan informasi mengenai kondisi lalu lintas di Surabaya. Setelah *tweet* didapatkan, *tweet* tersebut diberi label yaitu macet atau tidak. Terdapat total sebanyak 1636 *tweet* yang terdiri dari 1168 *tweet* berlabel macet dan 468 *tweet* berlabel tidak. Data tersebut didapatkan dari gabungan 1082 *tweet* dari akun @LMSurabaya dan 554 *tweet* dari akun @sits_dishubsby. *Tweet* tersebut diambil berdasarkan rentang waktu yaitu dari tanggal 31 Oktober 2017 sampai tanggal 28 September 2018 yang diambil melalui media sosial Twitter.com. Berikut contoh *dataset* dari *tweet* ditunjukkan oleh Tabel 3-1.

Tabel 3-1 Contoh *dataset tweet*

Tweet	Label
exit tol sidoarjo gunung sari macet	macet
arus lalu lintas simpang adityawarman indragiri terpantau ramai lancar	tidak

3.3 Perancangan Proses

Pada subbab ini akan menjelaskan mengenai perancangan proses yang dilakukan untuk setiap tahap pembuatan tugas akhir ini berdasarkan Gambar 3-2.

3.3.1 Desain *Word2vec*

Output dari *Word2vec* adalah vektor yang dapat merepresentasikan suatu kata (*word embeddings*). *Word2vec* membutuhkan korpus yang besar atau membutuhkan jumlah kata yang besar sehingga dapat menghasilkan *word embeddings*. Jumlah kata yang digunakan bisa puluhan juta sampai milyaran kata.

Pertama, dilakukan *parsing* dari format XML untuk medapatkan konten dari setiap artikel di Wikipedia Bahasa Indonesia. Data Wikipedia Bahasa Indonesia didapatkan dari

https://dumps.wikimedia.org/idwiki/latest/. Setelah itu dilakukan praproses menggunakan *library Gensim*. *Gensim* sudah menyediakan fasilitas untuk mentrain data Wikipedia. Praproses yang dilakukan adalah *tokenization* yaitu proses memisahkan setiap kata pada korpus dan menjadikan token-token. Lalu, token-token tersebut dijadikan huruf kecil atau *lower case*. Hasil dari praproses tersebut adalah 1 *file* yang berisi gabungan dari seluruh artikel di Wikipedia Bahasa Indonesia menjadi satu baris.

Kedua, setelah dilakukan praproses maka dilakukan pemodelan *Word2vec* yaitu menggunakan file hasil praproses untuk dijadikan *input* dari pemodelan *Word2vec*.

3.3.2 *Text Preprocessing*

Pada tahap ini, *tweet* yang sudah didapatkan dan dilabeli akan dilakukan proses *text preprocessing*. Proses itu meliputi *case folding* yaitu membuat huruf menjadi huruf kecil atau *lower case*, *tokenization* yaitu memisahkan kata menjadi token, *stopword removal* yaitu membuang kata yang sering muncul, dan *replace slang* yaitu mengubah kata singkatan menjadi kata yang sebenarnya. Tahap *stopword removal* dilakukan menggunakan *library* Sastrawi pada *python* yang sangat membantu dalam proses tersebut. Sehingga didapatkan *tweet* yang sudah bersih dan dapat dilakukan proses selanjutnya. *Input* dan *output* dari proses ini ditunjukkan pada Gambar 3-4 dan Gambar 3-5.

```
"#SBY RT @Hara_ci: Exit tol dari sidoarjo ke gunung sari macet. https://t.co/ocZXK6LQiK (via @maman_tea)"
"#SBY RT @byset: exit tol romokalisari macet.. https://t.co/acrCFHDHke (via @maman_tea)"
"#SBY RT @masdonicom: injoko arah gayungsari macet. https://t.co/roiZGKgs57 (via @maman_tea)"
"#SBY RT @dov_Now: Lalin ruas jln raya kenjeran arah jl kapasan padat tidak bergerak. https://t.co/0h0szdqnXB (via @TRIS2320)"
```

Gambar 3-4 *Input text preprocessing*

```
exit tol sidoarjo gunung sari macet
exit tol romokalisari macet
injoko arah gayungsari macet
lalu lintas ruas jalan raya kenjeran arah jalan kapasan padat bergerak
```

Gambar 3-5 *Output text preprocessing*

3.3.3 *POS Tagging*

Pada proses ini setiap kata dari *output preprocessing* akan diberi label kelas setiap katanya. Pelabelan dilakukan untuk dapat memberikan keterangan setiap kata pada kalimat sehingga dapat digunakan untuk pembobotan kata dengan vektor yang dihasilkan dari model *Word2vec*. Terdapat beberapa kelas kata yang digunakan yaitu kata kerja (*verb*), kata sifat (*adjective*), dan kata keterangan (*adverb*). *POS Tag* yang digunakan adalah *Polyglot*. *Polyglot* mengenali 17 *parts of speech*, ditunjukkan pada Tabel 3-2 [17]:

Tabel 3-2 Parts of speech pada Polyglot

Part of speech	Keterangan
ADJ	adjective
ADP	adposition
ADV	adverb
AUX	auxiliary verb
CONJ	coordinating conjunction
DET	determiner
INTJ	interjection
NOUN	noun
NUM	numeral
PART	particle
PRON	pronoun
PROPN	proper noun
PUNCT	punctuation
SCONJ	subordinating conjunction
SYM	symbol
VERB	verb
X	lainnya

Pada *POS Tag Polyglot* menyediakan berbagai Bahasa untuk dilakukan *POS Tag*. Pada tugas akhir ini akan digunakan Bahasa Indonesia dalam pendefinisian Bahasa yang digunakan.

Berikut merupakan penjelasan berbagai Bahasa yang disediakan oleh *Polyglot* pada Tabel 3-3:

Tabel 3-3 Bahasa yang disediakan *Polyglot* [17]

No	Bahasa
1	German
2	Italian
3	Danish
4	Czech
5	Slovene
6	French
7	English
8	Swedish
9	Bulgarian
10	Spanish; Castilian
11	Indonesian
12	Portuguese
13	Finnish
14	Irish
15	Hungarian
16	Dutch

3.3.4 *Sentence Mapping*

Setelah *tweet* melalui tahap *text preprocessing*, kemudian *tweet* tersebut akan dicari bentuk representasi vektornya. Proses mencari representasi vektor pada kata disebut dengan *sentence mapping*. Representasi vektor dari *tweet* dilakukan dengan cara merata-rata kedekatan kata setiap *Word Embeddings* kata pembentuk *tweet*. Jika ada kata yang tidak terdapat dalam *korpus* (tidak terdapat dalam *Word Embeddings*), maka kata tersebut tidak dianggap ada dalam *tweet* tersebut. Vektor dari suatu kata didapatkan dari *training word2vec* Wikipedia bahasia Indonesia yang sebelumnya telah dilakukan. Representasi vektor yang

digunakan pada setiap *tweet* adalah kata kerja (*verb*), kata sifat (*adjective*), dan kata keterangan (*adverb*) saja yang akan dicari rata-rata kedekatannya dengan kata "macet" yang sebelumnya sudah melalui tahap *Pos Tagging*.

3.3.5 Pemisahan *Dataset (Splitting*)

Pemisahan dataset menjadi *data training* dan *data testing* dilakukan agar dapat mengukur performa dari *Sentence Embeddings*. *Data training* adalah data yang menjadi bahan metode klasifikasi untuk melakukan *learning*. Sedangkan *data testing* adalah untuk memprediksi data *tweet*. Perbandingan data *training* adalah 70% dan data *testing* sebesar 30%.

3.3.6 Klasifikasi Menggunakan Metode *Naïve Bayes, Random Forest,* dan *Support Vector Machine*

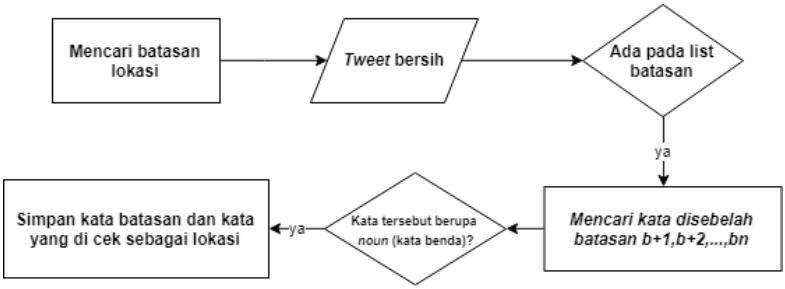
Pada tugas akhir ini, pengujian dilakukan dengan beberapa *classifier*, yaitu *Naïve Bayes, Random Forest,* dan *Linear Support Vector Machine*. *Naïve Bayes* memiliki tingkat akurasi yang tinggi dan kecepatan yang baik ketika diterapkan pada *database* yang besar. Untuk *Random Forest,* mampu menjadi sebuah metode klasifikasi yang baik karena beberapa *Decision Tree* yang ikut dibuat saat konstruksi memiliki kemampuan prediksi yang baik. Sedangkan untuk SVM, digunakan untuk menemukan fungsi pemisah yang optimal yang bisa memisahkan dua set data dari dua kelas yang berbeda. Pengujian klasifikasi menggunakan *library* pada PySpark yaitu *RandomForestClassifier*, *NaiveBayes,* dan *LinearSVC*.

3.3.7 Evaluasi

Evaluasi dari pengujian metode klasifikasi adalah untuk mengukur performa model. Evaluasi berupa nilai *accuracy*, *precision, recall,* dan *F-Measure* dari setiap *classifier* yang didapatkan dari *confusion matrix*.

3.3.8 Visualisasi

Visualiasai pada tugas akhir ini dilakukan dengan menampilkan lokasi kemacetan pada *map* (peta). Penentuan lokasi kemacetan dilakukan dengan mencari kata batasan yang terdapat pada *tweet*. Daftar kata batasan yang sudah ditentukan merupakan penanda bahwa kata tersebut menerangkan suatu lokasi. Setelah lokasi pada setiap *tweet* sudah didapatkan maka akan dilakukan penentuan *latitude* dan *longitude* dari lokasi tersebut dikarenakan *input* untuk menggunakan Google Maps API adalah *latitude* dan *longitude* lokasi. Perubahan lokasi menjadi *latitude* dan *longitude* dilakukan dengan menggunakan Geocoding API. Setelah *latitude* dan *longitude* dari lokasi kemacetan tersebut didapatkan, maka Google Maps API dapat melakukan *marker* lokasi kemacetan pada *map* (peta) sesuai dengan tanggal dan waktu yang diinginkan pengguna.Pada Gambar 3-6 menunjukkan diagram alir dalam mendapatkan lokasi kemacetan.



Gambar 3-6 Diagram alir deteksi lokasi

Algoritma untuk mendeteksi lokasi adalah sebagai berikut:

1. Mencari kata penanda lokasi pada *tweet* (daftar kata penanda lokasi terdapat di Lampiran bagian L2) .
2. Jika kata penanda ditemukan pada *tweet*, dilakukan pengecekan apakah kata penanda yang ditemukan berada pada posisi diujung kalimat atau tidak.

3. Jika kata penanda tidak berada pada ujung kalimat, akan dilakukan pengecekan kata disebelahnya (posisi setelah kata penanda) sampai kata yang berada diujung *tweet*. Kata-kata tersebut dilakukan pengecekan apakah kelas kata tersebut merupakan kata benda (*noun*) atau tidak.
4. Jika kelas kata yang dilakukan pengecekan adalah *noun*, maka kata penanda dan kata tersebut dideteksi sebagai lokasi dan disimpan sebagai lokasi *tweet* tersebut.
5. Jika kata penanda tidak ditemukan atau kata setelah penanda bukan berupa *noun*, maka lokasi dari *tweet* tersebut dianggap tidak ada dan tidak dapat dideteksi lokasinya.

[Halaman ini sengaja dikosongkan]

BAB IV
IMPLEMENTASI

Pada bab ini akan dijelaskan implementasi pada tugas akhir ini yaitu menjelaskan tahap-tahap dalam pengerjaan tugas akhir. Bab ini juga akan merinci *tools* yang digunakan pada tugas akhir.

4.1 *Lingkungan Implementasi*

Lingkungan implementasi sistem tugas akhir ini memiliki spresifikasi perangkat keras dan perangkat lunak yang dijelaskan oleh Tabel 4-1.

Tabel 4-1 Lingkungan implementasi

Perangkat	Spesifikasi
Perangkat Keras	• Prosesor: Intel® Core™ i5 • Memori: 8GB
Perangkat Lunak	• Sistem Operasi Windows 10 • Perangkat Pembantu Sublime Text 3, Microsoft Word 2016, Microsoft Power Point 2016, Microsoft Excel 2016

Pada Tabel 4-2 diuraikan pula *tools* yang digunakan untuk pengerjaan tugas akhir ini.

Tabel 4-2 *Tools* yang digunakan pada tugas akhir

No	Tools	Deskripsi
1	Python	Bahasa *Python* digunakan untuk menangani *task Natural Language Processing* (NLP).
2	Gensim	*Library* pada *Python* yang digunakan untuk melakukan *text preprocessing*.
3	Tweepy	*Library Python* yang digunakan untuk *crawling dataset tweet*.
4	Polyglot	*Library* yang digunakan untuk melakukan *POS Tag*.
5	PySpark (ml)	*Library* yang digunakan untuk melakukan klasifikasi dan evaluasi klasifikasi.
6	API Geocoding	API yang digunakan untuk mendapatkan *latitude* dan *longitude* lokasi kemacetan.
7	API Google Maps	API yang digunakan untuk melakukan *marker* pada lokasi kemacetan.

4.2 *Implementasi Proses*

Implementasi proses merupakan tahap implementasi pada perancangan proses yang sebelumnya sudah dijelaskan pada bab analisis dan perancangan sistem.

4.2.1 Implementasi *Word2vec*

Pada sub-bab ini akan dijelaskan tahap-tahap implementasi *Word2vec* untuk mendapatkan *Word Embeddings*. Korpus yang digunakan pada tugas akhir ini adalah Wikipedia Bahasa Indonesia yang dipublikasi secara terbuka di alamat https://dumps.wikimedia.org/idwiki/latest.Total token yang digunakan sejumlah 412424 token. Kata yang digunakan pada proses training adalah kata yang muncul paling sedikit 3 kali pada korpus.

4.2.1.1 *Preprocessing* pada Korpus

Setelah korpus Wikipedia Bahasa Indonesia diunduh, kemudian dilakukan proses *parsing* dari format XML. Data yang diambil hanya bagian isi artikel saja yang akan dijadikan korpus. Proses *parsing* dilakukan dengan menggunakan *library Gensim* yaitu menggunakan fungsi `gensim.corpora.WikiCorpus()`. Proses tersebut ditunjukkan pada Kode Sumber 4-1.

```
1  file_wiki_xml = nama/letak file korpus wikipedia
2  korpus_wiki = nama/letak file output
3
4  wiki = WikiCorpus(file_wiki_xml, lemmatize=False,
                     dictionary={}, lower=True)
5  for text in wiki.get_texts():
6      output.write(korpus_wiki)
```

Kode Sumber 4-1 Proses parsing korpus Wikipedia Bahasa Indonesia

Kelas `WikiCorpus` melakukan *parsing* seluruh artikel yang terdapat pada Wikipedia Bahasa Indonesia. `WikiCorpus` akan mengekstrak dan memproses korpus dengan *preprocessing* sederhana. Sedangkan `lemmatize=False` diperlukan agar ekstraksi tidak diperlambat oleh lematisasi. Lematisasi adalah proses yang bertujuan untuk melakukan normalisasi pada teks atau kata dengan berdasarkan pada bentuk dasar yang merupakan bentuk lemma-nya. Untuk `dictionary={}` agar korpus hanya dibaca satu kali dan karena hanya memerlukan token dari setiap kata saja [18]. Fungsi `get_text` pada *Gensim* berfungsi untuk mengambil bagian isi dari setiap artikel di Wikipedia Bahasa Indonesia. Sebelum hasilnya ditulis pada sebuah *file* (*output file*), setiap artikel yang didapatkan akan dilakukan proses *text preprocessing*. Setiap kata diubah menjadi *lower case*. Terakhir, kata-kata ditambahkan ke dalam *array* dan disimpan pada *file output*. *File output* yang dihasilkan

berupa 1 *file* besar yang berisi artikel-artikel pada Wikipedia Bahasa Indonesia yang dijadikan 1 baris.

4.2.1.2 *Training Word2vec*

Setelah korpus yang berasal dari Wikipedia Bahasa Indonesia dilakukan tahap *text preprocessing*, maka tahap selanjutnya adalah *training Word2vec*. Implementasi proses *training Word2vec* ditunjukkan pada Kode Sumber 4-2.

```
1  gensim.models import Word2Vec
2  gensim.models.word2vec import LineSentence
3
4  korpus_wiki = nama/letak file korpus wikipedia
5  model_word2vec = nama/letak file model word2vec
6  model = Word2Vec(LineSentence(namaFileInput) size
                    =400, window=5, min_count=3, sg=1)
```

Kode Sumber 4-2 *Training Word2vec*

Pada proses *training Word2vec* terhadap korpus Wikipedia terdapat beberapa parameter yang dapat diinisialisasi. Parameter tersebut berupa *size*, *window*, *min count*, dan juga algortima yang digunakan. *Size* merupakan dimensi dari vektor kata, *window* merupakan jarak maksimum antara posisi kata dengan kata prediksi dalam kalimat, dan *min count* merupakan jumlah minimum kata yang digunakan pada proses *training* (jumlah berapa kali kata digunakan dalam korpus) [19]. Pada tugas akhir ini digunakan parameter *size* 400 dan *min count* 3. *Output* dari *training Word2vec* tersebut adalah *file model word2vec*. Sebelumnya telah dilakukan percobaan dengan menggunakan *size* 100 dan *min count* 3, *size* 100 dan *min count* 5, serta *size* 400 dan *min count* 5. Dari hasil percobaan tersebut dihasilkan hasil yang lebih baik menggunakan *size* 400 dan *min count* 3.

4.2.2 Implementasi Tahap *Preprocessing Dataset*

Pada subab ini akan membahas implementasi tahapan *preprocessing*. Tahap *preprocessing* yang dilakukan adalah *case folding* yaitu merubah huruf pada *dataset* menjadi huruf kecil atau *lower case*, *tokenization* yaitu memisahkan kata menjadi token, *stopword removal* yaitu membuang kata yang sering muncul, dan *replace slang* yaitu membuang dan menjadikan kata yang disingkat menjadi kata yang sebenarnya. Tahap *stopword removal* dilakukan menggunakan *library* Sastrawi tetapi dengan menambahkan dan menghapus kata-kata yang sebelumnya tidak terdapat dalam *library* Sastrawi. Kata-kata yang ditambahkan yaitu kata "pkl" dan "wib", sedangkan kata yang dihapus yaitu kata "tidak". Bagian pertama mengenai proses ini adalah menentukan *regex* yang ditunjukkan pada Kode Sumber 4-3. Bagian kedua adalah proses implementasi *regex* dan *stopword removal* ditunjukkan pada Kode Sumber 4-4.

```
1  #regresion untuk menghapus URL dan menjadikan tweet menjadi huruf kecil
2  Regex1 = re.compiler (r'https?:\/\/.*[\ ]*', ' ', text.lower ())
3  #regresion untuk menghapus simbol retweet dan mention pada tweet
4  Regex2 = re.compiler (r'rt\ @\w+' , ' ')
5  #regresion untuk menghapus mention pada tweet
6  Regex3 = re.compiler (r'@\w+' , ' ' )
7  #regresion untuk menghapus hashtag
8  Regex4 = re.compiler (r'#\w+' , ' ')
9  #regresion untuk menghapus selain huruf
10 Regex5 = re.compiler ('[^ a-zA-Z]', ' ')
```

Kode Sumber 4-3 Implementasi *preprocessing* bagian 1

```
1  from Sastrawi.StopWordRemover.StopWordRemoverFactory import StopWordRemoverFactory
2
3  tweet = re.sub (r'https?:\/\/.*[\ ]*', ' ', text.lower ())
4  tweet = re.sub (r'rt\ @\w+' , ' ' , tweet)
5  tweet = re.sub (r'@\w+' , ' ' , tweet)
6  tweet = re.sub (r'#\w+' , ' ' , tweet)
7  tweet = re.sub ('[^ a-zA-Z]', ' ', tweet)
8  #stopword Sastrawi
9  factory = StopWordRemoverFactory()
10 stop = factory.create_stop_word_remover()
11 tweet = stop.remove(tweet)
12 #mengganti semua kata yang disingkat
13 def findAndReplace(sentence):
14     splitted=sentence.split()
15     with open('nama file kamus kata singkat') as f:
16         findAndReplace = json.load(f)
17         for index,words in enumerate(splitted):
18             splitted[index]=findAndReplace[words]
19     return sentence
```

Kode Sumber 4-4 Implementasi *preprocessing* bagian 2

Sedangkan pada tahap *replace slang* dilakukan dengan membuat kamus kata tersendiri yang disimpan dalam file berformat json. Daftar kamus kata ditunjukkan pada Lampiran bagian L.1.

4.2.3 Implementasi Tahap *POS Tagging*

Sebelum dilakukan tahap *sentence mapping* untuk pembobotan kata, perlu adanya proses *POS Tagging* untuk mendapatkan kelas dari suatu kata pada *tweet*. Kelas kata yang digunakan hanya kata kerja (*verb*), kata sifat (*adjective*), dan kata keterangan (*adverb*) akan disimpan dan dicari bobot kedekatan vektornya dengan kata "macet". Dalam mengimplementasi POS Tagging, digunakan *library Polyglot*. Implementasi *POS Tagging* pada *dataset* ditunjukkan pada Kode Sumber 4-5.

```
1  from polyglot.downloader import downloader
2  from polyglot.text import Text
3  #inisialisasi Bahasa yang digunakan
4  def getPosTag(tweet):
5      posTag=Text(tweet, hint_language_code='id')
6
7      listTag=['ADJ','ADV','VERB']
8      for kata in posTag.pos_tags:
9          if kata[0] in listKata:
10             hasil.append(kata[0])
11         elif kata[1] in listTag:
12             hasil.append(kata[0])
```

Kode Sumber 4-5 Implementasi *POS Tag* menggunakan *Polyglot*

Dikarenakan masih ada beberapa kekurangan pada *library Polyglot* dalam menentukan kelas kata pada suatu kata, maka dilakukan penambahan kamus *list* kata sehingga hasil dari *POS Tag* lebih *valid*. Maksud dari lebih *valid* adalah penambahan kata yang kemungkinan akan berpengaruh dalam pembobotan kata. *List*

kata tersebut antara lain “macet”, “tersendat”, “padat”, “merayap”, “merambat”.

Fungsi `posTag()` adalah untuk menginisialisasi jika Bahasa yang digunakan adalah Bahasa Indonesia dengan kode “id”. Variabel `listTag` adalah untuk menerangkan jika kata yang ingin didapatkan adalah kata kerja (VERB), kata sifat (ADJ) dan kata keterangan (ADV).

4.2.4 Implementasi Tahap *Sentence Mapping*

Sebelum *dataset tweet* masuk dalam proses *learning* dengan metode klasifikasi, *dataset tweet* perlu dilakukan pembobotan nilai *tweet*. *Sentence Embeddings* merupakan vektor yang merepresentasikan sebuah *tweet*. Nilai bobot yang didapatkan dengan cara merata-rata kedekatan *Word Embeddings* kata kerja (*verb*), kata sifat (*adjective*), dan kata keterangan (*adverb*) dengan kata “macet” pada kalimat pembentuk dari setiap *tweet*.

Implementasi dari proses *sentence mapping* ditunjukkan pada Kode Sumber 4-6.

```
1  #load model word2vec
2  namaFileModel = "w2vec_wiki_id_case"
3  model = gensim.models.Word2Vec.load(namaFileModel)
4
5  predictSentence=getPosTag(tweet)
6  jumlah_kata = 0
7  kata_yang_ada = 0
8  #mencari vektor kedekatan dengan kata “macet”
9  for word in predictSentence:
10         jumlah_kata += model.similarity (word, 'macet')
11         kata_yang_ada += 1
```

```
12  #mendapatkan vektor kalimat
13      if kata_yang_ada == 0:
14          tweet.bobot = 0
15      else:
16          bobot = jumlah_kata / kata_yang_ada
```

Kode Sumber 4-6 *Sentence Mapping*

Pada Kode Sumber 4-6, pada baris 9-11 diterangkan bahwa jika kata pembentuk *tweet* ditemukan dalam korpus, maka kata tersebut akan dicari bobot kedekatan kata dengan kata "macet" menggunakan fungsi `model.similar`. Setiap kata yang ditemukan dalam korpus dan sudah didapatkan bobot kedekatan dengan kata "macet", maka bobot setiap kata yang ditemukan akan dijumlahkan dan jumlah kata akan disimpan.

Selanjutnya pada Kode Sumber 4-6 pada baris 13-16 diterangkan bahwa jika tidak ada kata pembentuk *tweet* yang ditemukan pada korpus, maka bobot *tweet* tersebut adalah 0. Sedangkan jika ditemukan kata pada korpus, maka bobot setiap kata akan dijumlahkan dan dibagi dengan jumlah kata kata yang ditemukan pada korpus, sehingga didapatkan nilai bobot *tweet* tersebut.

4.2.5 Implementasi Pemisahan *Dataset* (*Splitting*)

Pemisahan dataset dilakukan agar performa metode klasifikasi dapat dievaluasi. Dataset dibagi menjadi data *training* dan data *testing*. Pada tugas akhir ini perbandingan data *training* dan data *testing* adalah 70 dibanding 30. Pemisahan *dataset* dapat ditinjau dari Kode Sumber 4-7.

```
1  (trainingData, testData) = data.randomSplit([0.7, 0.3])
```

Kode Sumber 4-7 Pemisahan *dataset*

4.2.6 Implementasi Klasifikasi

Proses implementasi metode klasifikasi dilakukan dengan *library* Mlib. *Apache Spark* menyediakan *library* berbagai metode klasifikasi. Metode klasifikasi yang digunakan pada tugas akhir ini yaitu *Naïve Bayes*, *Random Forest*, dan *Linear Support Vector Machine*. Sebelum dilakukan klasifikasi, perlu dilakukan *data preparation* yang dapat ditinjau pada Kode Sumber 4-8.

```
1  from pyspark.ml.feature import VectorAssembler
2  from pyspark.ml.feature import IndexToString,
   StringIndexer, VectorIndexer
3  #mengubah dataframe menjadi vektor
4  assembler =
   VectorAssembler(inputCols=["weight"],outputCol="feature
   s")
5  output = assembler.transform(parsedData)
6  #index label, menambahkan metadata ke kolom label
7  labelIndexer = StringIndexer(inputCol="kolom_label",
   outputCol="label").fit(output)
8  #identifikasi fitur kategorikal dan indeksnya
9  featureIndexer =\
       VectorIndexer(inputCol="kolom_feature",
   outputCol="features").fit(output)
```

Kode Sumber 4-8 *Data preparation*

Fungsi `VectorAssembler()` yaitu menggabungkan beberapa kolom menjadi kolom vektor. Fungsi `StringIndexer()` yaitu mengubah nilai kategorikal menjadi indeks kategori. Sedangkan `VectorIndexer()` untuk *indexing* kolom fitur dalam *dataset* dari vektor [20]. Masing-masing fungsi memiliki parameter *input* dan *output* kolom.

4.2.6.1 *Naïve Bayes*

Klasifikasi *Naïve Bayes* menggunakan *library ml* yang disediakan *Apache Spark*. Dalam mengimplementasi klasifikasi menggunakan *Naive Bayes*, maka digunakan fungsi NaiveBayes() dengan menggunakan beberapa parameter yaitu memberikan parameter kolom label serta kolom *feature*. Parameter yang digunakan pada klasifikasi *Naïve Bayes* adalah smoothing =1.0, yaitu *Laplace smoothing* yang nilai *default*nya adalah 1. Implementasi *Naïve Bayes* menggunakan *PySpark* dapat ditinjau pada Kode Sumber 4-9.

```
1  from pyspark.ml import Pipeline
2  from pyspark.ml.classification import NaiveBayes
3
4  #model naïve bayes
5  nb = NaiveBayes(smoothing=1.0, labelCol="label",
           featuresCol="features")
6  pipeline = Pipeline(stages=[labelIndexer, featureIndexer,
                              nb, labelConverter])
7  #train model
8  model = pipeline.fit(trainingData)
9  #save model
10 model.write().overwrite().save("nama_model")
```

Kode Sumber 4-9 Implementasi *Naive Bayes*

4.2.6.2 *Random Forest*

Klasifikasi *Random Forest* menggunakan *library ml* yang disediakan *Apache Spark*. Dalam mengimplementasi klasifikasi menggunakan *Random Forest*, maka digunakan fungsi RandomForestClassifier() dengan menggunakan beberapa parameter yaitu parameter kolom label, kolom *feature*, serta jumlah *tree*. Dalam implementasi klasifikasi ini, parameter jumlah *tree* yang digunakan yaitu 10 *tree*. Implementasi *Random Forest* menggunakan *PySpark* dapat ditinjau pada Kode Sumber 4-10.

```
1  from pyspark.ml import Pipeline
2  from pyspark.ml.classification import
   RandomForestClassifier
3  #model random forest
4  rf = RandomForestClassifier(labelCol="label",
                               featuresCol="features", numTrees=10)
5  pipeline = Pipeline(stages=[labelIndexer, featureIndexer, rf,
   labelConverter])
6  #train model
7  model = pipeline.fit(trainingData)
8  #save model
9  model.write().overwrite().save("nama_model")
```

Kode Sumber 4-10 Implementasi *Random Forest*

4.2.6.3 *Linear Support Vector Machine*

Klasifikasi *Linear Support Vector Machine* menggunakan *library ml* yang disediakan *Apache Spark*. Dalam mengimplementasi klasifikasi menggunakan *Linear Support Vector Machine*, maka digunakan fungsi LinearSVC() dengan

menggunakan beberapa parameter yaitu memberikan parameter kolom label serta kolom *feature*. Implementasi *Linear Support Vector Machine* menggunakan *PySpark* dapat ditinjau pada Kode Sumber 4-11.

```
1  from pyspark.ml import Pipeline
2  from pyspark.ml.classification import LinearSVC
3  #model linear support vector machine
4  lsvc = LinearSVC(labelCol="label",
                    featuresCol="features")
5  pipeline = Pipeline(stages=[labelIndexer, featureIndexer,
   lsvc])
6  #train model
7  model = pipeline.fit(trainingData)
8  #save model
9  model.write().overwrite().save("nama_model")
```

Kode Sumber 4-11 Implementasi *Linear Support Vector Machine*

4.2.6.4 Evaluasi hasil klasifikasi

Evaluasi pada setiap metode klasifikasi akan menggunakan akurasi, *precision*, *recall*, dan *F-Measure (f1)*. Sebelum dapat menghitung metrik-metrik tersebut, perlu didapatkan hasil prediksi model klasifikasi terhadap data *testing*. Evaluasi menggunakan *PySprak* terdapat *library* `MulticlassClassificationEvaluator()`.Evaluasi menggunakan *PySpark* dapat ditinjau dari Kode Sumber 4-12.

```
1  from pyspark.ml.evaluation import
   MulticlassClassificationEvaluator
2  #prediksi model klasifikasi terhadap data testing
3  predictions = model.transform(testData)
4  #evaluasi terhadap metric
5  evaluator = MulticlassClassificationEvaluator(
                    labelCol="label",
                     predictionCol="predictions",
                     metricName="metric")
6  #hasil evaluasi metrik
7  evaluasi = evaluator.evaluate(predictions)
```

Kode Sumber 4-12 Evaluasi pengujian klasifikasi

Fungsi `MulticlassClassificationEvaluator()` dapat melakukan evaluasi dari metrik-metrik. Metrik-metrik tersebut antara lain `accuracy`, `weightedPrecision`, `weightedRecall`, dan `f1`.

4.2.7 Implementasi Visualiasi

Tahap visualisasi yang dilakukan adalah menandai titik-titik lokasi (*marker*) kemacetan pada map. Untuk melakukan visualisasi pada *map*, maka dibutuhkan informasi mengenai lokasi kemacetan yang berasal dari *tweet*. Lokasi macet didapatkan dengan cara pertama adalah dengan membatasi kata penanda lokasi pada *tweet*. Daftar kata penanda lokasi pada *tweet* untuk implementasi tugas akhir ditunjukkan pada Lampiran bagian L.2

Kata penanda tersebut merupakan batasan untuk mendapatkan informasi lokasi terjadinya macet pada setiap *tweet*. Proses mendapatkan informasi lokasi kemacetan dapat ditinjau pada Kode Sumber 4-13.

```
1   batasan = daftar penanda lokasi
2   daftar_daerah = list kamus nama daerah
3
4   for word in tweet
5       index = tweet.find(batasan)
6       if index != -1
7           cek = tweet[(batasan)+1]
8               for word in cek
9                   index = kata.index(word)
10                      if postag[index] = noun
11                          return lokasi = (batasan+kata[index])
12                      else
13                          lokasi = null
```

Kode Sumber 4-13 Implementasi deteksi lokasi

Untuk mendapatkan lokasi macet dan melakukan pengecekan kata apakah kata tersebut *noun* (kata benda) yang diartikan sebagai nama daerah lokasi macet, maka dibuatlah kamus yang berisi daftar nama jalan atau daerah di Kota Surabaya dan daftar tersebut juga terdapat kelas kata. Kelas kata didapatkan dari pengecekan di KBBI Daring yang merupakan laman resmi pencarian kata dalam Kamus Besar Bahasa Indonesia (KBBI). KBBI Daring tersebut dikembangkan dan dikelola oleh Badan Pengembangan dan Pembinaan Bahasa. Situs KBBI Daring dapat diakses di situs https://kbbi.kemdikbud.go.id/. Daftar kamus lokasi tersebut ditunjukkan pada Lampiran bagian L.3. Alasan mengapa menggunakan kata penanda lokasi dan kamus tersebut karena dengan menggunakan penanda lokasi, maka diharapkan hasilnya

lebih baik dikarenakan pencarian lokasi pada *tweet* lebih spesifik jika menggunakan kata penanda tersebut.

Setelah mendapatkan informasi lokasi terjadinya macet pada setiap *tweet*, maka informasi lokasi tersebut akan disimpan. Langkah selanjutnya adalah mendapatkan *latitude* dan *longitude* dari lokasi yang telah didapatkan dan disimpan sebelumnya. Proses mencari *latitude* dan *longitude* lokasi macet tersebut dilakukan karena pada implementasi menandai lokasi macet pada *map* yang dibutuhkan adalah lokasi *latitude* dan *longitude*-nya. Untuk mendapatkan informasi *latitude* dan *longitude*, maka digunakan API Geocoding. Untuk menggunakan API Geocoding maka diperlukan Google API Key yang didapatkan dengan mendaftarkan sesuai prosedur penggunaan API Google. Proses mendapatkan *latitude* dan *longitude* dapat ditinjau dari Kode Sumber 4-14.

```
1  import geocoder
2  import request
3  key = '(api_key)'
4
5  res = requests.get(lokasi.tweet)
6  if res['status'] == 'OK':
7      latlng = res['results']
8      output_final.append(tweet)
9  elif res['status'] == 'OVER_QUERY_LIMIT':
10       time.sleep(2)
```

Kode Sumber 4-14 Implementasi mendapatkan Latitude & Longitude

Fungsi `request.get` berfungsi untuk mendapatkan *latitude* dan *longitude* dari lokasi. Pada baris 6-10 ditunjukkan bahwa jika lokasi ditemukan *latitude* dan *longitudenya*, maka *latitude* dan *longitude* lokasi akan disimpan pada setiap *tweet*. Jika keterangan status ‘OVER_QUERY_LIMIT’, maka dilakukan fungsi *delay*

setiap *request*. API Google hanya bisa menerima 1 *request* perdetik maka dibutuhkan fungsi `time.sleep()`.

Setelah *latitude* dan *longitude* lokasi didapatkan, langkah selanjutnya adalah menandai (*marker*) lokasi pada *map*. Menandai lokasi menggunakan Google Maps API. Dari informasi lokasi dan waktu setiap *tweet*, maka dapat dilakukan pemfilteran data lokasi kemacetan sesuai durasi waktu yang diinginkan. Penandaan lokasi macet dapat ditinjau dari Kode Sumber 4-15.

```
1   #input range waktu
2   start_time = waktu mulai
3   end_time = waktu selesai
    #filter waktu
4   for each (tweet)
5       if (tweet.created_at >= start_time && tweet.created_at
        <= end_time) then
6           new google.maps.Marker where lat = tweet.lat
                                        && lng  = tweet.lng
        end if
```

Kode Sumber 4-15 Implementasi visualisasi pada *map*

Fungsi `google.maps.Marker` adalah menandai lokasi pada map sesuai *latitude* dan *longitude* pada *tweet* yang sudah di filter sebelumnya berdasarkan rentang waktu yang diinginkan.

[Halaman ini sengaja dikosongkan

BAB V
UJI COBA DAN EVALUASI

Dalam bab ini dibahas mengenai hasil uji coba sistem yang telah dirancang dan dibuat. Pengujian dilakukan untuk mengetahui kinerja sistem dengan lingkungan uji coba yang telah ditentukan.

5.1 Lingkungan Uji Coba

Lingkungan pengujian sistem pada pengerjaan tugas akhir ini dilakukan pada lingkungan dan alat kakas sebagai berikut:

Prosesor	: Intel(R) Core(TM) i5-4200M CPU @ 2.50GHz (4 CPUs), ~2.5GHz
RAM	: 8 GB
Jenis Device	: Laptop
Sistem Operasi	: Windows 10 64-bit

5.2 *Word2vec*

Pada sub-bab ini dijelaskan uji coba dan evaluasi pada *Word Embeddings* yang dihasilkan oleh algortima *Word2vec*.

a. Uji dan Evaluasi 10 Kata Terdekat

Alasan digunakannya *Word2vec* pada tugas akhir ini adalah untuk mengatasi beragam kata yang memiliki kata yang mirip. Pada tugas akhir ini, kata "macet" dijadikan sebagai patokan yang dicari kemiripannya. Setiap kata kerja, kata sifat, dan kata keterangan dari pembentuk *tweet* akan dicari kedekatannya dengan kata "macet" dan akan dijumlah dan dicari rata-rata yang dijadikan bobot kalimat. Pada uji coba kali ini, kata "macet" akan dijadikan kata kunci untuk dicari 10 kata terdekatnya. Berikut merupakan hasil uji dari 10 kata terdekat dari kata "macet" yang merupakan hasil *training* dari Wikipedia Bahasa Indonesia yang dapat ditinjau

dari Tabel 5-1. Tabel 5-2 menunjukkan hasil uji 10 kata terdekat dari hasil *training* dari data uji (*tweet*):

Tabel 5-1 Hasil 10 kata terdekat korpus Wikipedia Bahasa Indonesia

No	Kata	Kedekatan kata
1	kemacetan	0.7408
2	anjlok	0.5895
3	ambruk	0.5849
4	kelelahan	0.5824
5	melambat	0.5788
6	keterlambatan	0.5678
7	terhambat	0.5590
8	tersendat	0.5581
9	panik	0.5475
10	padatnya	0.5470

Tabel 5-2 Hasil 10 kata terdekat korpus data uji (tweet)

No	Kata	Kedekatan kata
1	padat	0.99997419
2	surabaya	0.99997407
3	arah	0.99997317
4	jalan	0.99996876
5	hujan	0.99996829
6	truk	0.99996584
7	daerah	0.99996471
8	mogok	0.99996316
9	mulai	0.99996304
10	gresik	0.99996280

Dari hasil yang dapat ditinjau dari Tabel 5-1 dan Tabel 5-2, dapat disimpulkan bahwa *Word2vec* dapat merepresentasikan makna kata tetapi hasil tersebut bergantung pada *training* korpus

yang digunakan yaitu berasal dari artikel-artikel di Wikipedia Bahasa Indonesia dan data uji (*tweet*). Pada Tabel 5-1, dihasilkan 10 kata yang paling dekat vektor katanya dengan vektor kata "macet", artinya memiliki makna yang dekat dengan kata "macet". Kata-kata tersebut didapatkan dari hasil korpus yang berasal dari artikel-artikel pada Wikipedia Bahasa Indonesia.

Pada Tabel 5-2, dihasilkan 10 kata yang paling dekat vektor katanya dengan vektor kata "macet. Kata-kata tersebut didapatkan dari hasil korpus yang berasal dari data uji (*tweet)*. Data uji (*tweet)* yang dilakukan *training Word2vec* diambil dari *tweet* yang berisi keterangan kondisi jalan. Hasil skor kedekatan kata yang dihasilkan sangat tinggi yaitu 0.9999. Hasil yang sangat tinggi tersebut dikarenakan korpus data uji (*tweet*) memiliki jumlah kata yang sedikit yaitu sekitar 5-10 kata setiap *tweet*-nya.

Pada Tabel 5-2, skor kedekatan memiliki nilai yang tinggi, tetapi hasil kata terdekat kurang sesuai dengan makna kata "macet". Hasil skor pada Tabel 5-1 tidak memiliki skor yang lebih tinggi jika dibandingkan dengan hasil skor Tabel 5-2, tetapi mayoritas hasil kata terdekat pada Tabel 5-1 berupa kata sifat atau kata yang menerangkan kondisi jalan. Sedangkan, mayoritas hasil kata terdekat pada Tabel 5-2 berupa kata benda yang tidak memiliki kemiripan makna kata dengan kata "macet".

Metode *Word2vec* berfungsi untuk merepresentasikan makna kata dari korpus yang besar dengan cepat, hingga jutaan bahkan milyaran kata unik. Skor kedekatan kata yang dapat ditinjau pada Tabel 5-2 sangat tinggi dikarenakan jumlah korpus untuk *training* sangat sedikit dibandingkan dengan Wikipedia Bahasa Indonesia. Dari hasil kata terdekat dengan kata "macet" menggunakan korpus data uji (*tweet*) pada Tabel 5-2, *Word2vec* tidak berhasil merepresentasikan makna kata, dikarenakan kata yang dihasilkan tidak mempunyai makna yang mirip dengan kata "macet". Dapat disimpulkan *Word2vec* kurang cocok untuk korpus yang memiliki data yang sedikit seperti pada Twitter, tetapi *Word2vec* berhasil merepresentasikan makna kata dengan korpus yang besar, yaitu Wikipedia Bahasa Indonesia.

5.3 Data Uji Coba Klasifikasi

Data yang digunakan pada tugas akhir ini adalah data *tweet* yang berasal dari media sosial twitter. Pembagian data *training* dan *testing* mempunyai rasio perbandingan 70 : 30. Data uji coba dibagi menjadi 3 bagian yang dapat ditinjau dari Tabel 5-3.

Tabel 5-3 Data uji coba klasifikasi

Nama	Jumlah data berlabel macet	Jumlah data berlabel tidak	Jumlah data uji	Keterangan
Data Uji 1	1034	461	1495	Berisi tweet yang menginformasikan mengenai kondisi jalan atau lalu lintas di Surabaya.
Data Uji 2	214	727	941	Berisi tweet yang bervariasi, yaitu mengenai kondisi jalan dan berbagai informasi yang lainnya.
Data Uji 3	1248	1188	2436	Berisi tweet gabungan dari Data Uji 1 dan Data Uji 2.

5.4 Skenario Uji Coba Klasifikasi

Sub-bab ini akan menjelaskan skenario uji yang telah dilakukan. Terdapat beberapa skenario uji coba yang telah dilakukan. Pada masing-masing skenario dilakukan uji coba dengan metode klasifikasi dengan melakukan pengujian terhadap

jenis data uji coba yang sebelumnya sudah dijabarkan. Berikut merupakan penjelasan setiap skenario pengujian:

1. Skenario Pengujian 1: dalam skenario ini akan dilakukan pengujian dengan metode klasifikasi *Naïve Bayes* pada setiap data uji, yaitu Data Uji 1, Data Uji 2, dan Data Uji 3. Pengujian dilakukan dengan menggunakan vektor kedekatan dengan kata "macet" hasil *training Word2vec* dengan korpus yang berasal dari Wikipedia Bahasa Indonesia, vektor kedekatan kata "macet" hasil *training Word2vec* dengan korpus yang berasal dari data uji (*tweet*), dan dengan menggunakan TF-IDF dengan mengambil 10 *term* dengan jumlah *term frequency* terbanyak. Hasil dari pengujian terhadap setiap uji coba berupa nilai *accuracy*, *precision*, *recall*, dan *f-measure*.
2. Skenario Pengujian 2: dalam skenario ini akan dilakukan pengujian dengan metode klasifikasi *Random Forest* pada setiap data uji, yaitu Data Uji 1, Data Uji 2, dan Data Uji 3. Pengujian dilakukan dengan menggunakan vektor kedekatan dengan kata "macet" hasil *training Word2vec* dengan korpus yang berasal dari Wikipedia Bahasa Indonesia, vektor kedekatan kata "macet" hasil *training Word2vec* dengan korpus yang berasal dari data uji (*tweet*), dan dengan menggunakan TF-IDF dengan mengambil 10 *term* dengan jumlah *term frequency* terbanyak. Hasil dari pengujian terhadap setiap uji coba berupa nilai *accuracy*, *precision*, *recall*, dan *f-measure*.
3. Skenario Pengujian 3: dalam skenario ini akan dilakukan pengujian dengan metode klasifikasi *Linear Support Vector Machine (LSVC)* pada setiap data uji, yaitu Data Uji 1, Data Uji 2, dan Data Uji 3. Pengujian dilakukan dengan menggunakan vektor kedekatan dengan kata "macet" hasil *training Word2vec* dengan

korpus yang berasal dari Wikipedia Bahasa Indonesia, vektor kedekatan kata "macet" hasil *training Word2vec* dengan korpus yang berasal dari data uji (*tweet*), dan dengan menggunakan TF-IDF dengan mengambil 10 *term* dengan jumlah *term frequency* terbanyak. Hasil dari pengujian terhadap setiap uji coba berupa nilai *accuracy*, *precision*, *recall*, dan *f-measure*.

4. Skenario Pengujian 4: dalam skenario ini akan dilakukan pengujian dengan metode klasifikasi *Naïve Bayes*, *Random Forest*, dan *Linear Support Vector Machine* dengan menggunakan model yang dihasilkan *training* Data Uji 1. Dari model tersebut akan dilakukan uji coba menggunakan Data Uji 2 dan Data Uji 3 sebagai data *testing*. Pengujian dilakukan dengan menggunakan fitur vektor kedekatan dengan kata "macet" dari *Word2vec* dari korpus yang berasal dari Wikipedia Bahasa Indonesia, *Word2vec* dengan korpus yang berasal dari data uji (*tweet*), dan TF-IDF.
5. Skenario Pengujian 5: dalam skenario ini akan dilakukan pengujian dengan metode klasifikasi *Naïve Bayes*, *Random Forest*, dan *Linear Support Vector Machine* dengan menggunakan model yang dihasilkan *training* Data Uji 2. Dari model tersebut akan dilakukan uji coba menggunakan Data Uji 1 dan Data Uji 3 sebagai data *testing*. Pengujian dilakukan dengan menggunakan fitur vektor kedekatan dengan kata "macet" dari *Word2vec* dari korpus yang berasal dari Wikipedia Bahasa Indonesia, *Word2vec* dengan korpus yang berasal dari data uji (*tweet*), dan TF-IDF.
6. Skenario Pengujian 6: dalam skenario ini akan dilakukan pengujian dengan metode klasifikasi *Naïve Bayes*, *Random Forest*, dan *Linear Support Vector Machine* dengan menggunakan model yang dihasilkan *training* Data Uji 3. Dari model tersebut akan dilakukan

uji coba menggunakan Data Uji 1 dan Data Uji 2 sebagai data *testing*. Pengujian dilakukan dengan menggunakan fitur vektor kedekatan dengan kata "macet" dari *Word2vec* dari korpus yang berasal dari Wikipedia Bahasa Indonesia, *Word2vec* dengan korpus yang berasal dari data uji (*tweet*), dan TF-IDF.

7. Skenario Pengujian 7: dalam skenario ini akan dilakukan pengujian dengan metode klasifikasi *Random Forest* pada Data Uji 3 yang dibagi menjadi beberapa tahap jumlah data. Jumlah data yang diuji sebesar 100 *tweet*, 500 *tweet*, 1000 *tweet*, 1500 *tweet*, 2000 *tweet* dan 2500 *tweet*. Pengujian dilakukan dengan *training Word2vec* dengan korpus data uji (*tweet*), dan dengan TF-IDF.

5.4.1 Skenario Pengujian 1

Pada skenario ini dilakukan uji coba menggunakan metode klasifikasi *Naïve Bayes*. Pengujian dilakukan terhadap 3 jenis Data Uji. Tabel 5-4 merupakan hasil klasifikasi dengan menggunakan *training Word2vec* menggunakan korpus Wikipedia Bahasa Indonesia. Tabel 5-5 merupakan hasil klasifikasi dengan menggunakan *training Word2vec* menggunakan korpus data uji (*tweet*). Sedangkan, Tabel 5-6 merupakan hasil dari klasifikasi dengan metode *Naïve Bayes* menggunakan TF-IDF.

Gambar 5-1 merupakan rangkuman hasil dari skenario pengujian 1 yang terdiri dari hasil akurasi klasifikasi *Naïve Bayes* dengan hasil *training Word2vec* dari korpus Wikipedia Bahasa Indonesia, *Word2vec* dari korpus data uji (*tweet*), dan TF-IDF.

Tabel 5-4 Hasil skenario pengujian 1 dengan *Word2vec* Wikipedia Bahasa Indonesia

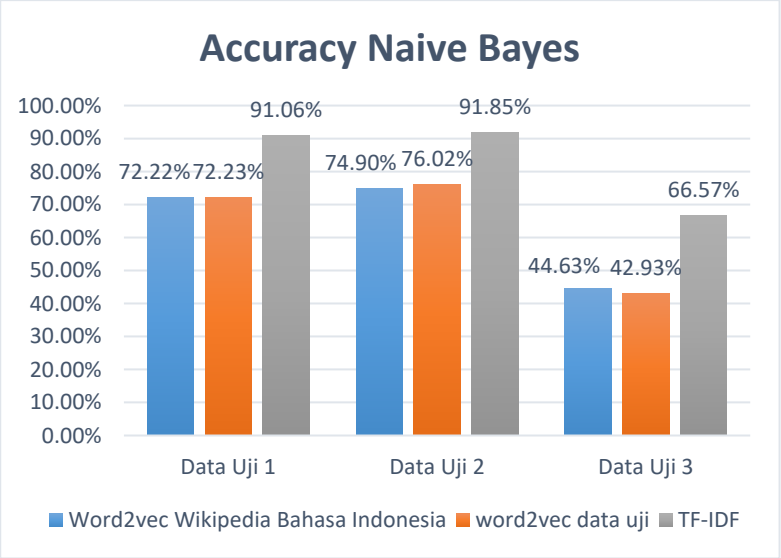
Data Uji	**Naïve Bayes**			
	Accuracy	**Precision**	**Recall**	**F-Measure**
Data Uji 1	72.22 %	52.17 %	72.23 %	60.59 %
Data Uji 2	74.90 %	56.10 %	74.90 %	64.15 %
Data Uji 3	44.63 %	19.92 %	44.63 %	27.54 %

Tabel 5-5 Hasil skenario pengujian 1 dengan *Word2vec* data uji (*tweet*)

Data Uji	**Naïve Bayes**			
	Accuracy	**Precision**	**Recall**	**F-Measure**
Data Uji 1	72.23 %	52.17 %	72.23 %	60.59 %
Data Uji 2	76.02 %	57.80 %	76.02 %	65.67 %
Data Uji 3	42.93 %	18.43 %	42.93 %	25.79 %

Tabel 5-6 Hasil skenario pengujian 1 dengan TF-IDF

Data Uji	**Naïve Bayes**			
	Accuracy	**Precision**	**Recall**	**F-Measure**
Data Uji 1	91.06 %	91.80 %	91.06 %	91.22%
Data Uji 2	91.85 %	92.66 %	91.85 %	92.07 %
Data Uji 3	66.57 %	71.48 %	66.57 %	64.13 %



Gambar 5-1 Grafik hasil skenario pengujian 1

5.4.2 Skenario Pengujian 2

Pada skenario ini dilakukan uji coba menggunakan metode klasifikasi *Random Forest*. Pengujian dilakukan terhadap 3 jenis Data Uji. Tabel 5-7 merupakan hasil klasifikasi dengan menggunakan *training Word2vec* menggunakan korpus Wikipedia Bahasa Indonesia. Tabel 5-8 merupakan hasil klasifikasi dengan menggunakan *training Word2vec* menggunakan korpus data uji (*tweet*). Sedangkan, Tabel 5-9 merupakan hasil dari klasifikasi dengan metode *Random Forest* menggunakan TF-IDF.

Gambar 5-2 merupakan rangkuman hasil dari skenario pengujian 2 yang terdiri dari hasil akurasi klasifikasi *Random Forest* dengan hasil *training Word2vec* dari korpus Wikipedia Bahasa Indonesia, *Word2vec* dari korpus data uji (*tweet*), dan TF-IDF.

Tabel 5-7 Hasil skenario pengujian 2 dengan *Word2vec* Wikipedia Bahasa Indonesia

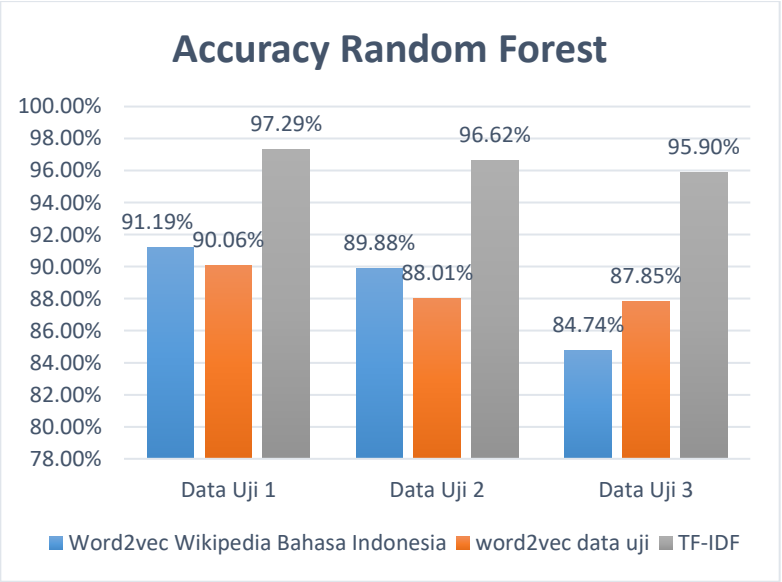
Data Uji	**Random Forest**			
	Accuracy	**Precision**	**Recall**	**F-Measure**
Data Uji 1	91.19 %	91.07 %	91.19 %	91.09 %
Data Uji 2	89.88 %	89.89 %	89.88 %	89.35 %
Data Uji 3	84.74 %	84.72 %	84.74 %	84.72 %

Tabel 5-8 Hasil skenario pengujian 2 dengan *Word2vec* data uji (*tweet*)

Data Uji	**Random Forest**			
	Accuracy	**Precision**	**Recall**	**F-Measure**
Data Uji 1	90.06 %	90.32 %	90.06 %	90.16 %
Data Uji 2	88.01 %	88.79 %	88.01 %	86.72 %
Data Uji 3	87.85 %	88.58 %	87.85 %	87.91 %

Tabel 5-9 Hasil skenario pengujian 2 dengan TF-IDF

Data Uji	**Random Forest**			
	Accuracy	**Precision**	**Recall**	**F-Measure**
Data Uji 1	97.29 %	97.34 %	97.29 %	97.25 %
Data Uji 2	96.62 %	96.60 %	96.62 %	96.60 %
Data Uji 3	95.90 %	96.03 %	95.90 %	95.91 %



Gambar 5-2 Grafik hasil skenario pengujian 2

5.4.3 Skenario Pengujian 3

Pada skenario ini dilakukan uji coba menggunakan metode klasifikasi *Linear Support Vector Machine*. Pengujian dilakukan terhadap 3 jenis Data Uji. Tabel 5-10 merupakan hasil klasifikasi dengan menggunakan *training Word2vec* menggunakan korpus Wikipedia Bahasa Indonesia. Tabel 5-11 merupakan hasil klasifikasi dengan menggunakan *training Word2vec* menggunakan korpus data uji (*tweet*). Sedangkan, Tabel 5-12 merupakan hasil dari klasifikasi dengan metode *Random Forest* menggunakan TF-IDF.

Gambar 5-3 merupakan rangkuman hasil dari skenario pengujian 3 yang terdiri dari hasil akurasi klasifikasi *Linear Support Vector Machine* dengan hasil *training Word2vec* dari

korpus Wikipedia Bahasa Indonesia, *Word2vec* dari korpus data uji (*tweet*), dan TF-IDF:

Tabel 5-10 Hasil skenario pengujian 3 dengan *Word2vec* Wikipedia Bahasa Indonesia

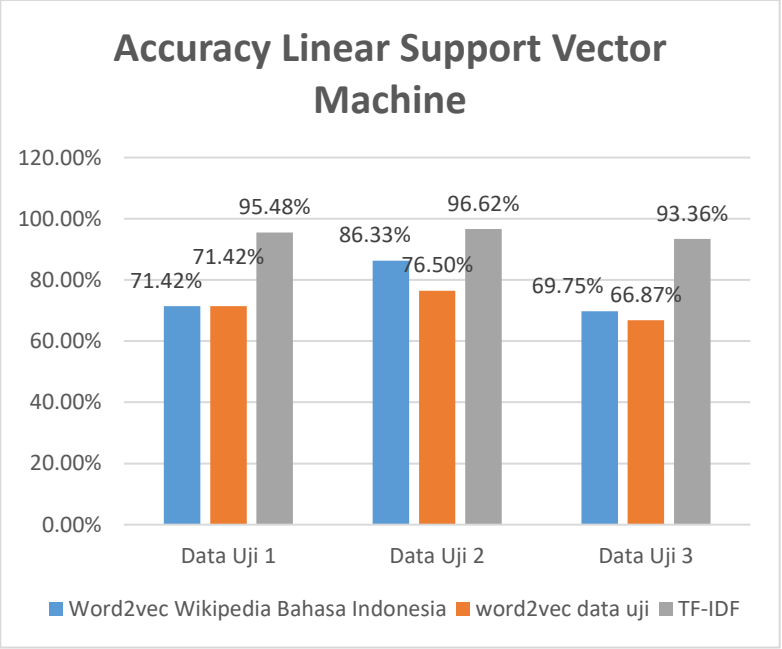
Data Uji	**Linear Support Vector Machine**			
	Accuracy	**Precision**	**Recall**	**F-Measure**
Data Uji 1	71.42 %	51.02 %	71.42 %	59.52 %
Data Uji 2	86.33 %	88.43 %	86.33 %	84.27 %
Data Uji 3	69.75 %	78.29 %	69.75 %	68.15 %

Tabel 5-11 Hasil skenario pengujian 3 dengan *Word2vec* data uji (*tweet*)

Data Uji	**Linear Support Vector Machine**			
	Accuracy	**Precision**	**Recall**	**F-Measure**
Data Uji 1	71.42 %	51.02 %	71.42 %	59.52 %
Data Uji 2	76.50 %	58.52 %	76.50 %	66.31 %
Data Uji 3	66.87 %	76.64 %	66.87 %	60.56 %

Tabel 5-12 Hasil skenario pengujian 3 dengan TF-IDF

Data Uji	Linear Support Vector Machine			
	Accuracy	**Precision**	**Recall**	**F-Measure**
Data Uji 1	95.48 %	95.48 %	95.48 %	95.48 %
Data Uji 2	96.62 %	96.60 %	96.62 %	96.60 %
Data Uji 3	93.36 %	93.90 %	93.36 %	93.38 %



Gambar 5-3 Grafik hasil skenario pengujian 3

5.4.4 Skenario Pengujian 4

Pada skenario ini dilakukan uji coba menggunakan metode klasifikasi *Naïve Bayes*, *Random Forest*, dan *Linear Support Vector Machine*. Pengujian dilakukan menggunakan model hasil *training* Data Uji 1. Data *testing* yang digunakan adalah Data Uji 2 dan Data Uji 3. Uji coba dilakukan dengan hasil *training Word2vec* dengan korpus Wikipedia Bahasa Indonesia, *Word2vec* dengan korpus data uji (*tweet*), dan TF-IDF. Hasil uji coba *Word2vec* dengan korpus Wikipedia Bahasa Indonesia dapat ditinjau pada Tabel 5-13, hasil uji coba *Word2vec* dengan korpus data uji (*tweet*) dapat ditinjau pada Tabel 5-14, sedangkan hasil uji coba TF-IDF dapat ditinjau pada Tabel 5.15.

Tabel 5-13 Hasil skenario pengujian 4 *Word2vec* Wikipedia Bahasa Indonesia

Data Uji	Evaluasi	Naïve Bayes	Random Forest	Linear SVM
Data Uji 2	accuracy	25.09 %	75.65 %	25.09 %
	precision	6.29 %	84.14 %	6.29 %
	recall	25.09 %	75.65 %	25.09 %
	f-measure	10.67 %	77.23 %	10.06 %
Data Uji 3	accuracy	55.36 %	84.88 %	55.36 %
	precision	30.65 %	85.26 %	30.65 %
	recall	55.36 %	84.88 %	55.36 %
	f-measure	39.46 %	84.71 %	39.46 %

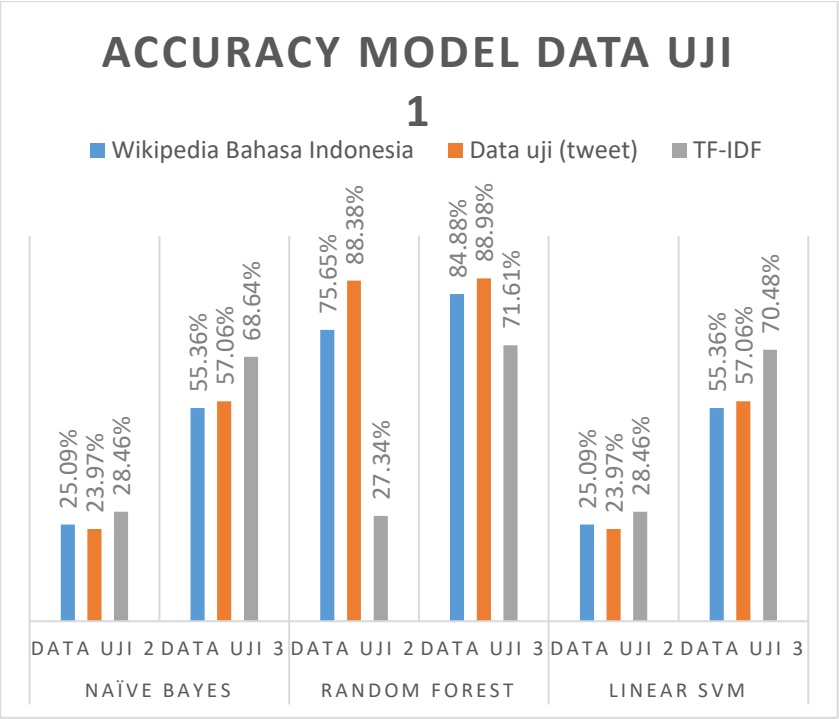
Gambar 5-4 merupakan rangkuman hasil skenario pengujian 4, yaitu pengujian terhadap model yang dihasilkan Data Uji 1. Pengujian dilakukan dengan metode klasifikasi *Naïve Bayes*, *Random Forest*, dan *Linear Support Vector Machine*.

Tabel 5-14 Hasil skenario pengujian 4 *Word2vec* data uji (*tweet*)

Data Uji	**Evaluasi**	**Naïve Bayes**	**Random Forest**	**Linear SVM**
Data Uji 2	accuracy	23.97 %	88.38 %	23.97 %
	precision	5.74 %	89.32 %	5.74 %
	recall	23.97 %	88.38 %	23.97 %
	f-measure	9.26 %	88.68%	9.26 %
Data Uji 3	accuracy	57.06 %	88.98 %	57.06 %
	precision	32.56 %	89.08 %	32.56 %
	recall	57.06 %	88.98 %	57.06 %
	f-measure	41.46 %	89 %	41.46 %

Tabel 5-15 Hasil skenario pengujian 4 TF-IDF

Data Uji	**Evaluasi**	**Naïve Bayes**	**Random Forest**	**Linear SVM**
Data Uji 2	accuracy	28.46 %	27.34 %	28.46 %
	precision	61.46 %	81.34 %	69.80 %
	recall	28.46%	27.34 %	28.46 %
	f-measure	19.45 %	14.61 %	17.89 %
Data Uji 3	accuracy	68.64 %	71.61 %	70.48 %
	precision	71.44 %	80.54 %	75.87 %
	recall	68.64 %	71.61 %	70.48 %
	f-measure	66.22 %	68.03 %	67.52 %



Gambar 5-4 Grafik hasil skenario pengujian 4

5.4.5 Skenario Pengujian 5

Pada skenario ini dilakukan uji coba menggunakan metode klasifikasi *Naïve Bayes*, *Random Forest*, dan *Linear Support Vector Machine*. Pengujian dilakukan menggunakan model hasil *training* Data Uji 2. Data *testing* yang digunakan adalah Data Uji 1 dan Data Uji 3. Uji coba dilakukan dengan hasil *training* *Word2vec* dengan korpus Wikipedia Bahasa Indonesia, *Word2vec* dengan korpus data uji (*tweet*), dan TF-IDF. Hasil uji coba *Word2vec* dengan korpus Wikipedia Bahasa Indonesia dapat ditinjau pada Tabel 5-16, hasil uji coba *Word2vec* dengan korpus data uji (*tweet*) dapat ditinjau pada Tabel 5-17, sedangkan hasil uji coba TF-IDF dapat ditinjau pada Tabel 5.18.

Tabel 5-16 Hasil skenario pengujian 5 *Word2vec* Wikipedia Bahasa Indonesia

Data Uji	Evaluasi	Naïve Bayes	Random Forest	Linear SVM
Data Uji 1	accuracy	27.76 %	61.85 %	58.01 %
	precision	7.70 %	71.06 %	83.28 %
	recall	27.76 %	61.85 %	58.01 %
	f-measure	12.06 %	63.84 %	58.45 %
Data Uji 3	accuracy	44.63 %	71.18 %	67.79 %
	precision	19.92 %	74.12 %	80.35 %
	recall	44.63 %	71.18 %	67.79 %
	f-measure	27.54 %	71 %	65.62 %

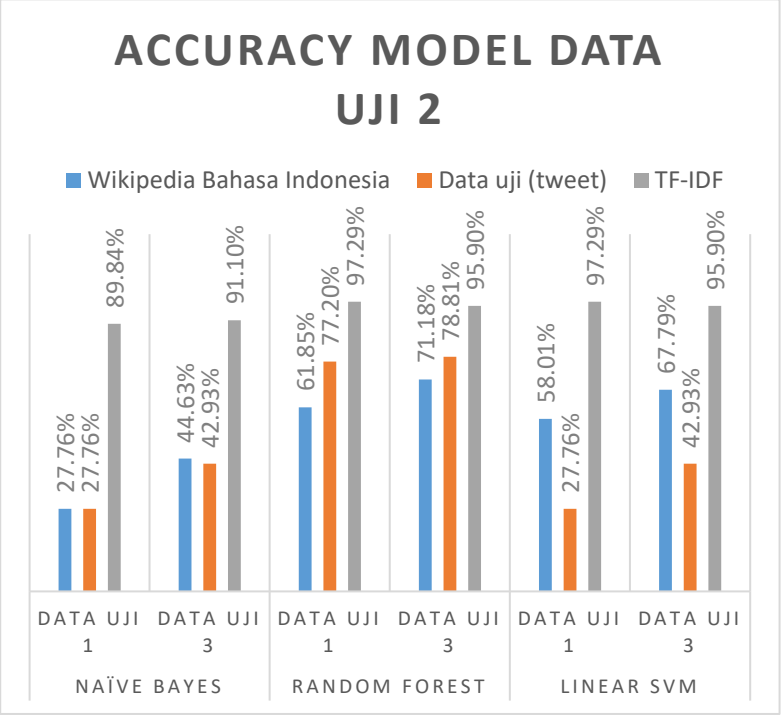
Gambar 5-5 merupakan rangkuman hasil skenario pengujian 5, yaitu pengujian terhadap model yang dihasilkan Data Uji 2. Pengujian dilakukan dengan metode klasifikasi *Naïve Bayes*, *Random Forest*, dan *Linear Support Vector Machine*.

Tabel 5-17 Hasil skenario pengujian 5 *Word2vec* data uji (*tweet*)

Data Uji	Evaluasi	Naïve Bayes	Random Forest	Linear SVM
Data Uji 1	accuracy	27.76 %	77.20 %	27.76 %
	precision	7.70 %	87.16 %	7.70 %
	recall	27.76 %	77.20 %	27.76 %
	f-measure	12.06 %	78.38 %	12.06 %
Data Uji 3	accuracy	42.93 %	78.81 %	42.93 %
	precision	18.43 %	85.07 %	18.43 %
	recall	42.93 %	78.81 %	42.93 %
	f-measure	25.79 %	78.56 %	25.79 %

Tabel 5-18 Hasil skenario pengujian 5 TF-IDF

Data Uji	Evaluasi	Naïve Bayes	Random Forest	Linear SVM
Data Uji 1	accuracy	89.84 %	97.29 %	97.29 %
	precision	89.90 %	97.53 %	97.53 %
	recall	89.84 %	97.29 %	97.29 %
	f-measure	89.42 %	97.32 %	97.32 %
Data Uji 3	accuracy	91.10 %	95.90 %	95.90 %
	precision	91.21 %	96.09 %	96.09 %
	recall	91.10 %	95.90 %	95.90 %
	f-measure	91.06 %	95.91 %	95.91 %



Gambar 5-5 Grafik hasil skenario pengujian 5

5.4.6 Skenario Pengujian 6

Pada skenario ini dilakukan uji coba menggunakan metode klasifikasi *Naïve Bayes*, *Random Forest*, dan *Linear Support Vector Machine.* Pengujian dilakukan menggunakan model hasil *training* Data Uji 3. Data *testing* yang digunakan adalah Data Uji 1 dan Data Uji 2. Uji coba dilakukan dengan hasil *training Word2vec* dengan korpus Wikipedia Bahasa Indonesia, *Word2vec* dengan korpus data uji (*tweet*), dan TF-IDF. Hasil uji coba *Word2vec* dengan korpus Wikipedia Bahasa Indonesia dapat ditinjau pada Tabel 5-19, hasil uji coba *Word2vec* dengan korpus data uji (*tweet*) dapat ditinjau pada Tabel 5-20, sedangkan hasil uji coba TF-IDF dapat ditinjau pada Tabel 5.21.

Tabel 5-19 Hasil skenario pengujian 6 *Word2vec* Wikipedia Bahasa Indonesia

Data Uji	Evaluasi	Naïve Bayes	Random Forest	Linear SVM
Data Uji 1	accuracy	27.76 %	87.81 %	59.59 %
	precision	7.70 %	88.02 %	83.54 %
	recall	27.76 %	87.81 %	59.59 %
	f-measure	12.06 %	87.89 %	60.25 %
Data Uji 2	accuracy	74.90 %	83.89 %	87.64 %
	precision	56.10 %	86.87 %	87.35 %
	recall	74.90 %	83.89 %	87.64 %
	f-measure	64.15 %	84.62 %	86.99 %

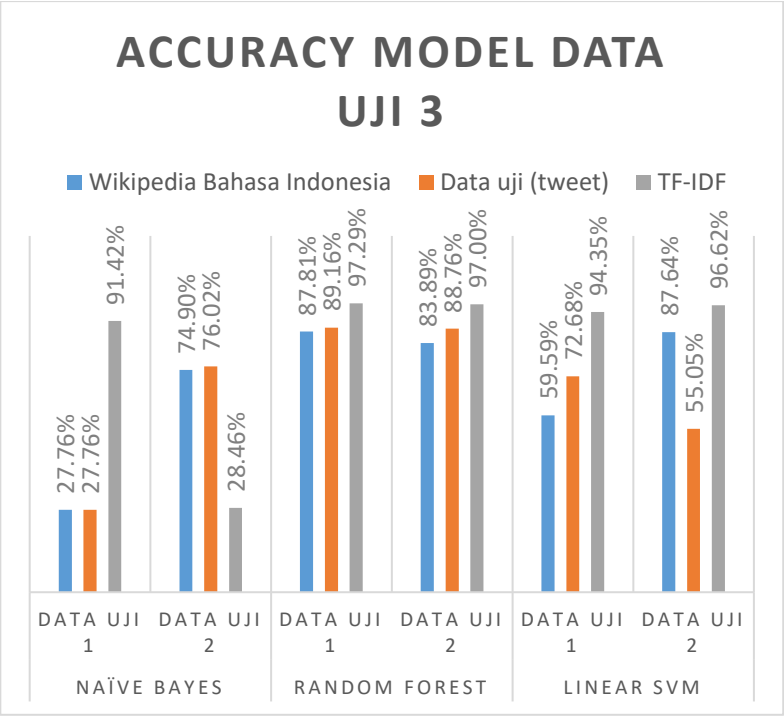
Gambar 5-6 merupakan rangkuman hasil skenario pengujian 6, yaitu pengujian terhadap model yang dihasilkan Data Uji 3. Pengujian dilakukan dengan metode klasifikasi *Naïve Bayes*, *Random Forest*, dan *Linear Support Vector Machine*.

Tabel 5-20 Hasil skenario pengujian 6 *Word2vec* data uji (*tweet*)

Data Uji	Evaluasi	Naïve Bayes	Random Forest	Linear SVM
Data Uji 1	accuracy	27.76 %	89.16 %	72.68 %
	precision	7.70 %	90.98 %	69.99 %
	recall	27.76 %	89.16 %	72.68 %
	f-measure	12.06 %	89.51 %	62.77 %
Data Uji 2	accuracy	76.02 %	88.76 %	55.05 %
	precision	57.80 %	89.04 %	83.43 %
	recall	76.02 %	88.76 %	55.05 %
	f-measure	65.67 %	88.87 %	56.62 %

Tabel 5-21 Hasil skenario pengujian 6 TF-IDF

Data Uji	Evaluasi	Naïve Bayes	Random Forest	Linear SVM
Data Uji 1	accuracy	91.42 %	97.29 %	94.35 %
	precision	92.20 %	97.53 %	95.30 %
	recall	91.42 %	97.29 %	94.35 %
	f-measure	91.60 %	97.32 %	94.50 %
Data Uji 2	accuracy	28.46 %	97 %	96.62 %
	precision	61.46 %	97 %	96.61 %
	recall	28.46 %	97 %	96.62 %
	f-measure	19.45 %	97 %	96.62 %



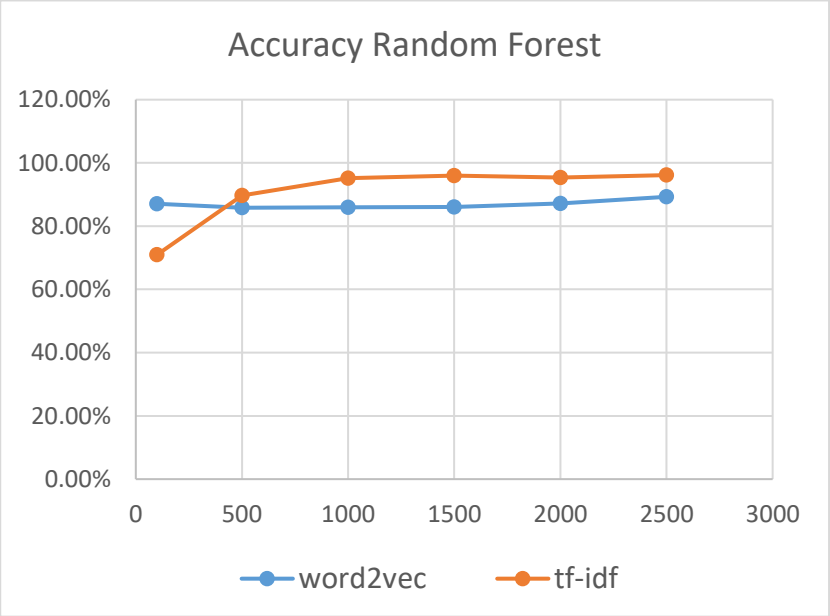
Gambar 5-6 Grafik hasil skenario pengujian 6

5.4.7 Skenario Pengujian 7

Pada skenario ini dilakukan uji coba menggunakan metode klasifikasi *Random Forest*. Pengujian dilakukan menggunakan Data Uji 3 yang dibagi menjadi beberapa tahap jumlah data. Jumlah data yang diuji sebesar 100 *tweet*, 480 *tweet*, 960 *tweet*, 1440 *tweet*, 1920 *tweet* dan 2400 *tweet*. Pengujian dilakukan dengan *training Word2vec* dengan korpus data uji (*tweet*) dan dengan TF-IDF. Evaluasi dari skenario pengujian ini berupa nilai akurasi. Hasil uji coba dapat ditinjau pada Tabel 5-22. Rangkuman dari hasil skenario pengujian 7 juga dapat ditinjau pada Gambar 5-7.

Tabel 5-22 Hasil akurasi skenario pengujian 7

Jumlah tweet	Word2vec	TF-IDF
100	87.09 %	70.96 %
480	85.82 %	89.76 %
960	85.97 %	95.20 %
1440	86.11 %	96 %
1920	87.21 %	95.38 %
2400	89.28 %	96.14 %



Gambar 5-7 Grafik hasil skenario pengujian 7

5.5 Visualisasi

Pada sub-bab ini akan menjelaskan uji coba dalam mengimplementasikan visualisasi. Uji coba berupa deteksi lokasi, deteksi *latitude longitude* lokasi, dan visualiasi pada *map*.

5.5.1 Uji Coba Deteksi Lokasi

Uji coba deteksi lokasi dilakukan pada setiap *tweet* dengan mencari kata penanda lokasi dan akan dicek kata setelah kata penanda apakah kata benda (*noun*) atau tidak pada *tweet*, jika kata tersebut berupa kata benda (*noun*), maka kata tersebut dianggap sebagai lokasi. Daftar kata penanda dapat ditinjau dari Lampiran pada bagian L.2. Uji coba dilakukan pada Data Uji dan berikut merupakan hasil deteksi lokasi dari hasil uji untuk 10 data yang dapat ditinjau dari Tabel 5-23.

Tabel 5-23 Hasil uji coba deteksi lokasi

No	Data Uji	Deteksi Lokasi
1	exit tol sidoarjo gunung sari macet	exit tol sidoarjo gunung sari
2	exit tol romokalisari macet	exit tol romokalisari
3	injoko arah gayungsari macet	arah gayungsari
4	grahadi macet total	-
5	jalan mastrip macet, cuaca hujan	jalan mastrip
6	margorejo giant macet	-
7	setelah waru utama padat merambat cenderung berhenti	-
8	lalu lintas margomulyo padat merayap	lalu lintas margomulyo
9	masuk pintu tol waru arah sidoarjo macet	masuk pintu tol waru arah sidoarjo
10	keluar tol waru arah cito padat merambat	keluar tol waru arah cito

Dari hasil yang dapat ditinjau pada Tabel 5-23 dapat disimpulkan bahwa lokasi yang tidak dapat dideteksi merupakan lokasi merupakan *tweet* yang tidak memiliki kata penanda. Sedangkan *tweet* yang terdeteksi lokasinya merupakan *tweet* yang mempunyai kata penanda batasan pada *tweet* tersebut.

5.5.2 Uji Coba Deteksi *Latitude* dan *Longitude* Lokasi

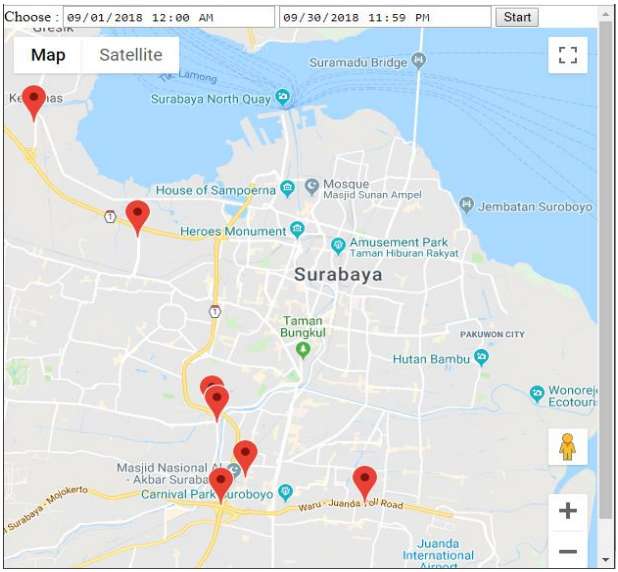
Uji coba deteksi *latitude* dan *longitude* lokasi dilakukan pada hasil deteksi lokasi pada setiap *tweet*. Lokasi yang sudah dideteksi akan dideteksi *latitude* dan *longitude* menggunakan API Geocoding. Lokasi yang sudah dideteksi *latitude* dan *longitude*-nya, selanjutnya akan divisualisasikan pada *map*. Visualisasi pada *map* tidak dapat dilakukan dengan nama lokasi, tetapi dengan *latitude* dan *longitude*-nya sehingga lokasi dapat dilakukan *marker* pada *map*. Uji coba deteksi latitude dan longitude dilakukan pada Data Uji dan berikut merupakan hasil deteksi *latitude* dan *longitude* lokasi dari hasil uji untuk 7 data yang dapat ditinjau dari Tabel 5-24.

Tabel 5-24 Hasil uji coba deteksi *Latitude* & *Longitude*

No	Lokasi	Latitude	Longitude
1	exit tol sidoarjo gunung sari	-7.3098097	112.7080185
2	exit tol romokalisari	-7.201547700000001	112.6456553
3	arah gayungsari	-7.333833599999999	112.7196709
4	jalan mastrip	-7.340491900000001	112.6969401
5	lalu lintas margomulyo	-7.244166	112.6821195
6	masuk pintu tol waru arah sidoarjo	-7.3441674	112.7110325
7	keluar tol waru arah cito	-7.343400600000001	112.7616055

Dari hasil uji deteksi *latitude* dan *longitude* lokasi, selanjutnya adalah pengujian pada map menggunakan API Google Maps. Pada Gambar 5-8 dapat ditinjau hasil dari visualiasi 7 lokasi yang terdeketsi.

Untuk dapat dilihat lebih rinci dan spesifik terhadap visualisasi pada map dan membuktikan apakah hasil marker sesuai dengan Data Uji. Berikut merupakan hasil visualiasi pada setiap Data Uji yang dapat ditinjau dari Gambar 5-9, Gambar 5-10, Gambar 5-11, Gambar 5-12, Gambar 5-13, Gambar 5-14, dan Gambar 5-15.



Gambar 5-8 Hasil visualiasi data uji pada map

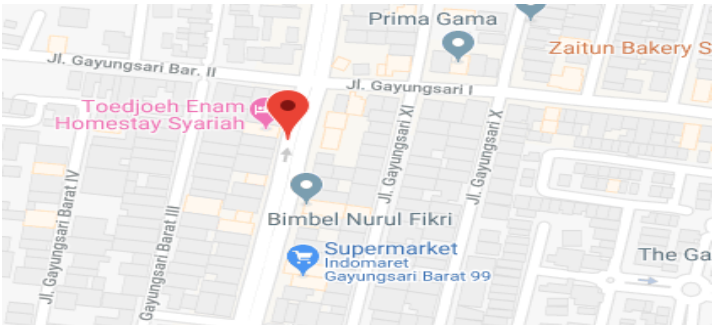
Gambar 5-9 Hasil visualisasi data uji no. 1

Pada Gambar 5-9 menggambarkan bahwa lokasi yang tertanda (marker) yaitu di sekitar Gerbang Tol Gunung Sari 2. Data Uji No. 1 yaitu berlokasi di “exit tol sidoarjo gunung sari”. Berdasarkan Data Uji dan hasil visualisasi, maka disimpulkan bahwa data uji dan hasil visualisasi adalah benar.

Gambar 5-10 Hasil visualisasi data uji no. 2

Pada Gambar 5-10 menggambarkan bahwa lokasi yang tertanda (marker) yaitu di sekitar Gerbang Tol Romokalisari. Data

Uji No. 2 yaitu berlokasi di “exit tol romokalisari”. Berdasarkan Data Uji dan hasil visualisasi, maka disimpulkan bahwa data uji dan hasil visualisasi adalah benar.



Gambar 5-11 Hasil visualisasi data uji no. 3

Pada Gambar 5-11 menggambarkan bahwa lokasi yang tertanda (marker) yaitu di sekitar Jalan Gayungsari. Data Uji No. 3 yaitu berlokasi di “arah gayungsari”. Berdasarkan Data Uji dan hasil visualisasi, maka disimpulkan bahwa data uji dan hasil visualisasi adalah benar.

Gambar 5-12 Hasil visualisasi data uji no. 4

Pada Gambar 5-12 menggambarkan bahwa lokasi yang tertanda (marker) yaitu di sekitar Jalan Mastrip. Data Uji No. 4 yaitu berlokasi di “jalan mastrip”. Berdasarkan Data Uji dan hasil visualisasi, maka disimpulkan bahwa data uji dan hasil visualisasi adalah benar.

Gambar 5-13 Hasil visualisasi data uji no. 5

Pada Gambar 5-13 menggambarkan bahwa lokasi yang tertanda (marker) yaitu di sekitar Jalan Masrgomulyo. Data Uji No. 5 yaitu berlokasi di “lalu lintas margomulyo”. Berdasarkan Data Uji dan hasil visualisasi, maka disimpulkan bahwa data uji dan hasil visualisasi adalah benar.

Gambar 5-14 Hasil visualisasi data uji no. 6

Pada Gambar 5-14 menggambarkan bahwa lokasi yang tertanda (marker) yaitu di sekitar Tol Waru. Data Uji No. 6 yaitu berlokasi di “masuk pintu tol waru arah sidoarjo”. Berdasarkan Data Uji dan hasil visualisasi, maka disimpulkan bahwa data uji dan hasil visualisasi adalah benar.

Gambar 5-15 Hasil visualisasi data uji no. 7

Pada Gambar 5-15 menggambarkan bahwa lokasi yang tertanda (marker) yaitu di sekitar Tol Waru. Data Uji No. 7 yaitu berlokasi di “keluar tol waru arah cito”. Berdasarkan Data Uji dan hasil visualisasi, maka disimpulkan bahwa data uji dan hasil visualisasi adalah benar.

5.6 Evaluasi

Pengujian *Word2vec* menggunakan korpus Wikipedia Bahasa Indonesia berhasil merepresentasikan maka kata, dapat ditinjau pada Tabel 5-1. Pada Tabel 5-1, dibuktikan bahwa kata terdekat yang dihasilkan merupakan kata yang mirip maknanya dengan kata “macet”. Hasil pengujian kata terdekat dengan kata “macet” menggunakan korpus data uji (*tweet*) menghasilkan skor kedekatan sangat tinggi (0.9999), hasil dapat ditinjau pada Tabel 5-2. Walaupun hasil skor kedekatan kata pada Tabel 5-1 tidak lebih tinggi dibandingkan dengan skor kedekatan pada Tabel 5-2, hasil

kata terdekat Tabel 5.2 tidak merepresentasikan makna kata yang mirip dengan kata “macet”. Dapat disimpulkan bahwa *Word2vec* tidak cocok menggunakan korpus yang berasal dari *tweet*. Hasil *Word2vec* dipengaruhi oleh besar kecilnya korpus yang digunakan.

Pengujian 1 Klasifikasi diperoleh bahwa klasifikasi *Naive Bayes* dengan TF-IDF lebih baik dibandingan hasil *Word2vec* yang menggunakan korpus Wikipedia Bahasa Indonesia maupun korpus data uji.

Pengujian 2 Klasifikasi diperoleh bahwa hasil klasifikasi *Random Forest* dengan TF-IDF lebih baik dibandingan hasil *Word2vec* yang menggunakan korpus Wikipedia Bahasa Indonesia maupun korpus data uji.

Pengujian 3 Klasifikasi diperoleh bahwa klasifikasi *Linear Support Vector Machine* dengan TF-IDF lebih baik dibandingan hasil *Word2vec* yang menggunakan korpus Wikipedia Bahasa Indonesia maupun korpus data uji.

Pengujian 4 Klasifikasi pada model *training* Data Uji 1 dan data *test* menggunakan Data Uji 2 diperoleh bahwa nilai akurasi dengan model *Word2vec* dengan korpus Wikipedia Bahasa Indonesia, model *Word2vec* dengan korpus data uji (*tweet*) dan dengan menggunakan TF-IDF, memiliki nilai terbaik keseluruhan model diperoleh dengan metode klasifikasi *Random Forest*, yaitu nilai akurasi dengan korpus Wikipedia Bahasa Indonesia sebesar 75.65%, nilai akurasi dengan korpus data uji (*tweet*) sebesar 88.38%, dan nilai akurasi dengan TF-IDF sebesar 27.34%. Sedangkan dengan data *test* menggunakan Data Uji 3 diperoleh nilai akurasi dengan korpus Wikipedia Bahasa Indonesia sebesar 84.88%, nilai akurasi dengan korpus data uji (*tweet*) sebesar 88.98%, dan nilai akurasi dengan TF-IDF sebesar 71.61%.

Pengujian 5 Klasifikasi pada model *training* Data Uji 2 dan data *test* menggunakan Data Uji 1 diperoleh bahwa nilai akurasi dengan model *Word2vec* dengan korpus Wikipedia Bahasa Indonesia, model *Word2vec* dengan korpus data uji (*tweet*) dan

dengan menggunakan TF-IDF memiliki nilai terbaik keseluruhan model diperoleh dengan metode klasifikasi *Random Forest*, yaitu nilai akurasi dengan korpus Wikipedia Bahasa Indonesia sebesar 61.85%, nilai akurasi dengan korpus data uji (*tweet*) sebesar 77.20%, dan nilai akurasi dengan TF-IDF sebesar 97.29%. Sedangkan dengan data *test* menggunakan Data Uji 3 diperoleh nilai akurasi dengan korpus Wikipedia Bahasa Indonesia sebesar 71.18%, nilai akurasi dengan korpus data uji (*tweet*) sebesar 78.81%, dan nilai akurasi dengan TF-IDF sebesar 95.90%.

Pengujian 6 Klasifikasi pada model *training* Data Uji 3 dan data *test* menggunakan Data Uji 1 diperoleh bahwa nilai akurasi dengan model *Word2vec* dengan korpus Wikipedia Bahasa Indonesia, model *Word2vec* dengan korpus data uji (*tweet*) dan dengan menggunakan TF-IDF memiliki nilai terbaik keseluruhan model diperoleh dengan metode klasifikasi *Random Forest*, yaitu nilai akurasi dengan korpus Wikipedia Bahasa Indonesia sebesar 87.81%, nilai akurasi dengan korpus data uji (*tweet*) sebesar 89.16%, dan nilai akurasi dengan TF-IDF sebesar 97.29%. Sedangkan dengan data *test* menggunakan Data Uji 2 diperoleh nilai akurasi dengan korpus Wikipedia Bahasa Indonesia sebesar 83.89%, nilai akurasi dengan korpus data uji (*tweet*) sebesar 88.76%, dan nilai akurasi dengan TF-IDF sebesar 97%.

Pengujian 7 Klasifikasi *Random Forest* terhadap Data Uji 3 yang dibagi menjadi beberapa bagian yang diujikan dengan *Word2vec* menggunakan korpus data uji (*tweet*) dan TF-IDF. Hasil akurasi untuk 100 *tweet* pada *Word2vec* memiliki hasil yang lebih baik dibandingkan TF-IDF yaitu sebesar 87.09%. Untuk jumlah data yang sedikit, *Word2vec* memiliki hasil yang lebih baik. Seiring dengan bertambahnya jumlah data (*tweet*), nilai akurasi metode TF-IDF semakin baik karena semakin banyak data, maka nilai TF-IDF perkata semakin mendefinisikan suatu kelas tertentu.

Pengujian Visualisasi diperoleh bahwa dalam mendeteksi lokasi berhasil dilakukan jika *tweet* tersebut memiliki kata penanda lokasi sesuai daftar kata yang dibuat. Daftar kata penanda tersebut

menjadi batasan dalam deteksi lokasi. Jika *tweet* tidak memiliki unsur kata penanda, maka *tweet* tidak dapat dideteksi lokasinya dan tidak dapat divisualisasikan pada *map*.

[Halaman ini sengaja dikosongkan]

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini akan diberikan kesimpulan yang diperoleh selama pengerjaan tugas akhir. Kesimpulan nantinya sebagai jawaban dari rumusan masalah yang dikemukakan. Selain kesimpulan, juga terdapat saran yang ditujukan untuk pengembangan penelitian lebih lanjut di masa depan.

6.1 Kesimpulan

Dari hasil pengamatan selama proses perancangan, implementasi, dan pengujian yang dilakukan, dapat diambil kesimpulan sebagai berikut:

1. Cara mendapatkan representasi makna kata adalah dengan metode *Word2vec*. *Word2vec* merupakan algoritma *word embedding*, yang melakukan pemetaan dari kata menjadi vektor dan berguna dalam berbagai macam tugas *Natural Language Processing* (NLP). *Word2vec* melakukan prediksi antara setiap kata dengan kata konteksnya menjadi vektor. Vektor yang dihasilkan pada tiap katanya mengandung makna. Semakin tinggi skor kedekatan vektor kedua kata, maka kedua kata tersebut semakin mirip maknanya.

   Preprocessing yang dilakukan pada Wikipedia Bahasa Indonesia menggunakan *library Gensim*, yaitu mengambil isi artikel yang terdapat pada Wikipedia Bahasa Indonesia dan menjadikan setiap artikel menjadi 1 baris. Sedangkan *preprocessing* yang dilakukan pada data uji (*tweet*) adalah melakukan pemfilteran pada *tweet* yang berisi kondisi jalan saja. Setelah itu dilakukan *case folding* (menjadi huruf kecil), menjadikan token dan mengganti kata singkatan. Pada *tweet* juga dilakukan penghapusan *retweet*, *mention*, URL, angka, karakter, dan *stopword*.

   Word2vec dapat merepresentasikan suatu makna kata yang dibuktikan dengan dapat menghasilkan kedekatan antar kata dengan cukup baik menggunakan korpus Wikipedia Bahasa

Indonesia, terbukti dengan hasil uji kedekatan kata dengan kata "macet" yang dapat ditinjau pada Tabel 5-1.

Hasil *Word2vec* tersebut dapat dipengaruhi pada beberapa hal, contohnya adalah keberagaman kata pada korpus, parameter-parameter yang digunakan pada training *Word2vec* seperti *size*, *window*, dan *min count*. *Word2vec* membutuhkan korpus yang besar sehingga dapat merepresentasikan makna kata dengan baik. Jika tidak memiliki korpus yang besar, maka kedekatan vektor menjadi sangat dekat seperti yang dapat ditinjau pada Tabel 5-2. Pada Tabel 5-2, dilakukan uji 10 kata terdekat dengan kata "macet" dengan korpus yang berasal dari data uji (*tweet*), yang memberikan hasil skor kedekatan yang sangat tinggi (0.9999…). Tetapi, hasil uji kata terdekat pada Tabel 5-2 tidak memberikan hasil yang mirip makna katanya dengan kata "macet". Sedangkan, hasil uji kata terdekat dengan korpus Wikipedia Bahasa Indonesia menghasilkan skor kedekatan kata yang tidak lebih baik dibandingkan dengan korpus data uji (*tweet*). Namun, kata terdekat yang dihasilkan berhasil merepresentasikan makna kata yang dekat dengan kata "macet". Dapat disimpulkan bahwa *Word2vec* tidak cocok jika menggunakan *tweet* sebagai korpus.

2. Hasil klasifikasi dengan *Word2vec* tidak lebih baik dibandingan dengan metode TF-IDF. Setelah dilakukan klasifikasi menggunakan *Naïve Bayes*, *Random Forest*, dan SVM, hasil yang didapat memberikan nilai akurasi terbaik metode *Random Forest* secara keseluruhan yaitu 84.74% dengan menggunakan korpus Wikipedia Bahasa Indonesia, dan 87.85% menggunakan korpus data uji (*tweet*). Hasil akurasi yang didapat tidak lebih baik dibandingkan dengan TF-IDF sebesar 95.90% dengan metode *Random Forest*. Hasil dengan TF-IDF menghasilkan nilai akurasi yang lebih baik dikarenakan dari algoritma TF-IDF menggunakan *term frequency* yang berasal dari dokumen yang diujikan sehingga semakin mendefinisikan suatu kelas tertentu pada dokumen

tersebut. Untuk dataset *tweet* dari *Twitter* lebih cocok menggunakan metode TF-IDF.

6.2 Saran

Berikut merupakan beberapa saran untuk pengembangan sistem di masa yang akan dating. Saran-saran ini didasarkan pada hasil perancangan, implementasi, dan pengujian yang telah dilakukan.

1. Melakukan uji coba terhadap algortima *word embeddings* lainnya, contohnya adalah Word Embeddings GloVe, sehingga dapat dilakukan perbandingan dan perbedaan hasil setiap penggunaan *word embeddings.*
2. Melakukan deteksi lokasi tanpa adanya batasan sehingga hasil deteksi lokasi lebih maksimal.
3. Sistem dibuat untuk data yang *realtime*.

[Halaman ini sengaja dikosongkan]

DAFTAR PUSTAKA

[1] A. Ramadhiani, "kompas," 25 02 2018. [Online]. Available: https://properti.kompas.com/read/2018/02/25/182046621/ini-10-kota-termacet-di-indonesia. [Diakses 23 Mei 2018].

[2] Seon, "Seon," 2018. [Online]. Available: https://seon.co.id/pengertian-media-sosial-facebook-twitter-google-youtube-instagram/. [Diakses 23 Mei 2018].

[3] C. C. Aggarwal, Data Mining The Textbook, vol. 181, London: Springer International Publishing Switzerland, 2015, pp. 1138-1152.

[4] V. R. Oktavia, Aplikasi Deteksi Kejadian di Jalan Raya berdasarkan Data Twitter Menggunakan Metode Support Vector Machine, Surabaya: Institut Teknologi Sepuluh Nopember, 2017.

[5] T. Mikolov, K. Chen, G. Corrado dan J. Dean, "Efficient Estimation of Word Representations in," 2013.

[6] R. S. Wahono, "romisatriawahono," 11 2017. [Online]. Available: http://romisatriawahono.net/dm/. [Diakses 22 05 2018].

[7] T. Mikolov, I. Sutskever, K. Chen, G. Corrado dan J. Dean, "Distributed Representations of Words and Phrases," dalam *NIPS Conference*, 2013.

[8] S. U. School of Engineering, "Lecture 2 | Word Vector Representations : word2vec," youtube.com, 2017.

[9] M. Rouse, "whatis.techtarget," Maret 2016. [Online]. Available: https://whatis.techtarget.com/definition/data-set. [Diakses 5 Mei 2018].

[10] KDnuggets, "kdnuggets.com," Desember 2017. [Online]. Available: https://www.kdnuggets.com/2017/12/general-approach-preprocessing-text-data.html. [Diakses 05 Mei 2018].

[11] B. Wahyono dan M. , "erlangga.co.id," Penerbit Erlangga, 26 Agustus 2016. [Online]. Available: https://erlangga.co.id/materi-

belajar/sma/8792-macam-macam-kelas-kata.html. [Diakses 10 November 2018].

[12] J. Han, M. Kamber dan J. Pei, Data Mining Concepts and Techniques 3rd edition, Elsevier Inc, 2012.

[13] S. L. B. Ginti dan R. P. Trinanda, "TEKNIK DATA MINING MENGGUNAKAN METODE BAYES CLASSIFIER UNTUK OPTIMALISASI PENCARIAN PADA APLIKASI PERPUSTAKAAN (STUDI KASUS : PERPUSTAKAAN UNIVERSITAS PASUNDAN – BANDUNG," *Jurnal Teknologi dan Informasi UNIKOM,* vol. Vol 1 No 6, 2014.

[14] N. Donges, "towardsdatascience.com," [Online]. Available: https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd. [Diakses 05 Mei 2018].

[15] B. Pang dan L. Lee, Opinion Mining and Sentiment Analysis, 2008.

[16] M. Bonzanini, Mastering Social Media Mining with Python, Birmingham: Packt Publishing Ltd., 2016.

[17] R. Al-Rfou, "polyglot," 2014. [Online]. Available: https://polyglot.readthedocs.io/en/latest/POS.html. [Diakses 10 10 2018].

[18] D. Nugraha, "medium.com," [Online]. Available: https://medium.com/@diekanugraha/membuat-model-word2vec-bahasa-indonesia-dari-wikipedia-menggunakan-gensim-e5745b98714d. [Diakses November 2018].

[19] R. Rehurek, "radimrehurek.com," 2009. [Online]. Available: https://radimrehurek.com/gensim/models/word2vec.html. [Diakses Oktober 2018].

[20] "spark.apache.org," The Apache Software Foundation, 2018. [Online]. Available: https://spark.apache.org/docs/2.3.1/ml-features.html. [Diakses 20 Oktober 2018].

LAMPIRAN

L.1 *Daftar Kamus Replace Slang*

Kata singkat	Kata sebenarnya
jl	jalan
jln	jalan
jlan	jalan
sby	surabaya
kecalakaan	kecelakaan
mrambat	merambat
sblm	sebelum
arh	arah
mnuju	menuju
yg	yang
lalin	lalu lintas
ry	raya
dikwsn	dikawasan
k	ke
mlg	malang
kluar	keluar
stlh	setelah
dlm	dalam
dmpk	dampak
lwt	lewat

Kata singkat	Kata sebenarnya
jmbatan	jembatan
gngsari	gunungsari
bund	bundaran
brhenti	berhenti
mnjelang	menjelang
prsimpangan	persimpangan
mrayap	merayap
rya	raya
srbaya	surabaya
mcet	macet
dwpan	depan
pnyebab	penyebab
kmacetan	kemacetan

L.2 *Daftar Kata Penanda Lokasi*

No	Kata penanda
1	pertigaan
2	perempatan
3	persimpangan
4	simpang
5	bundaran
6	masuk
7	lalu lintas
8	entry
9	gate
10	gerbang
11	lampu merah
12	laying
13	flyover
14	arteri
15	exit
16	keluar
17	tol
18	jalan
19	jalur
20	raya
21	arah
22	seputaran

L.3 Daftar Lokasi dan Kelas Kata

Kata	Kelas Kata
exit	n
tol	n
sidoarjo	n
waru	n
mastrip	n
arah	n
gunungsari	n
pabrik	n
saruta	n
driyorejo	n
larangan	n
spbu	n
legundi	n
jalan	n
kenjeran	n
arah	n
kapasan	n
jembatan	n
trosobo	n
krian	n
gerbang	n
menanggal	n
juanda	n
gate	n
taman	n
bibis	n
balongsari	n

Kata	Kelas Kata
kletek	n
dupak	n
malang	n
pandegiling	n
basra	n
digedangan	n
aloha	n
prapen	n
sma	n
tugu	n
pahlawan	n
pintu	n
japanan	n

[Halaman ini sengaja dikosongkan]

BIODATA PENULIS

Rakhma Rufaida Hanum, lahir di Jakarta pada tanggal 13 Mei 1996. Penulis menempuh Pendidikan mulai TK-IT Az-Zahra Tangerang (2001-2002), SD Negeri Gunung 05 Mexico Pagi Jakarta (2002-2008), SMP Negeri 11 Jakarta(2008-2011), SMA Negeri 70 Jakarta (2011-2014) dan sekarang sedang menempuh Pendidikan S1 Informatika di ITS. Penulis aktif dalam organisasi dan kepanitiaan Himpunan Mahasiswa Teknik Computer (HMTC) dan aktif dalam kepanitiaan Schematics ITS. Diantaranya adalah menjadi staff Departemen Minat dan Bakat HMTC 2016-2017, BPH III Revolutionary Entertainment and Expo with Various Art (REEVA) Schematics 2016, serta menjadi BPH Bendahara II Schematics 2017. Penulis berpengalaman sebagai asisten dosen pada matakuliah Manajemen Basis Data dan pernah melaksanakan kerja praktik di PT GMF AeroAsia di Bandara Soekarno Hatta, Cengkareng, Tangerang.