TUGAS AKHIR – IF184802

DETEKSI KELUHAN PENGGUNA SECARA REALTIME PADA TWEET BERBAHASA INDONESIA MENGGUNAKAN APACHE FLINK

AKBAR NOTO PONCO BIMANTORO
NRP 05111640000028

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

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

TUGAS AKHIR – IF184802

DETEKSI KELUHAN PENGGUNA SECARA REALTIME PADA TWEET BERBAHASA INDONESIA MENGGUNAKAN APACHE FLINK

AKBAR NOTO PONCO BIMANTORO
NRP 05111640000028

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

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

[Halaman ini sengaja dikosongkan]

FINAL PROJECT – IF184802

REALTIME USER COMPLAINT DETECTION ON INDONESIAN TWEETS USING APACHE FLINK

AKBAR NOTO PONCO BIMANTORO
NRP 05111640000028

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

INFORMATICS DEPARTMENT
Faculty of Intelligent Electrical and Informatics Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2020

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

DETEKSI KELUHAN PENGGUNA SECARA REALTIME PADA TWEET BERBAHASA INDONESIA MENGGUNAKAN APACHE FLINK

TUGAS AKHIR

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

Oleh:
AKBAR NOTO PONCO BIMANTORO
NRP: 05111640000028

Disetujui oleh Dosen Pembimbing Tugas Akhir:

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

Nurul Fajrin Ariyani, S.Kom., M.Sc. .....
NIP: 19860722 201504 2 003 (pembimbing 2)

SURABAYA
23 JULI 2020

[Halaman ini sengaja dikosongkan]

DETEKSI KELUHAN PENGGUNA SECARA REALTIME PADA TWEET BERBAHASA INDONESIA MENGGUNAKANAPACHE FLINK

Nama Mahasiswa	: Akbar Noto Ponco Bimantoro
NRP	: 05111640000028
Departemen	: Informatika FTEIC-ITS
Dosen Pembimbing 1	: Abdul Munif, S.Kom., M.Sc.
Dosen Pembimbing 2	: Nurul Fajrin Ariyani, S.Kom., M.Sc.

ABSTRAK

Perkembangan dan mudahnya akses internet membuat media sosial sangat populer di kalangan masyarakat. Masyarakat cenderung menggunakan media sosial seperti facebook, instagram, twitter, dan media lain sejenis sebagai tempat mengekspresikan opini. Hal ini mengundang perusahaan atau organisasi untuk ikut serta menggunakan sosial media sebagai bentuk memahami konsumen, sehingga membuka peluang akses informasi yang bisa didapatkan. Contohnya, opini publik terhadap organisasi, perusahaan, atau produk. Salah satu analisa yang menarik digunakan pada media sosial Twitter adalah mendeteksi keluhan pengguna seperti ganguan jaringan pada perusahaan penyedia layanan telekomunikasi.

Permasalahan jaringan pada penyedia jaringan seperti layanan telekomunikasi merupakan salah satu masalah yang cukup krusial dan perlu penanganan secepat mungkin sehingga memerlukan pemrosesan data secara realtime agar informasi terkait gangguan didapatkan sedini mungkin. Bila seseorang merasakan gangguan jaringan yang tidak disertai dengan penanganan yang cepat dan baik, maka besar kemungkinan orang tersebut akan beralih ke penyedia jasa lain serupa.

Oleh karena itu, pada tugas akhir ini akan dibuat sebuah aplikasi yang dapat mengklasifikasikan jenis keluhan pengguna untuk mendeteksi gangguan jaringan pada penyedia layanan telekomunikasi melalui media sosial Twitter. Pengembangan aplikasi menggunakan Apache Flink diperlukan model klasifikasi yang toleran terhadap kegagalan (serializable). Hal ini dapat diselesaikan dengan mendeklarasikan model SVM pada sebuah State Class. Selanjutnya, datastream yang telah melalui preprocessing diklasifikasikan dan diverifikasi berdasarkan kejadian serupa pada tempat dan waktu yang berdekatan. Lalu hasil pemrosesan tersebut ditampilkan bersama dengan hasil ekstraksi entitas-entitas seperti Named Entity Recoginition dan ekstraksi lokasi pada sebuah REST API.

Sistem yang dibuat menggunakan Apache Flink versi Standalone pada komputer i7-4720HQ dapat mengklasifikasikan 3 datastream dalam waktu 1 hingga 2,5 detik dengan akurasi 43%. Dari pengujian yang dilakukan, terdapat 70 gangguan yang terdeteksi dari 5580 tweet.

Kata kunci: Deteksi Keluhan, Realtime, Tweet, Apache Flink, Streaming.

REALTIME USER COMPLAINT DETECTION ON INDONESIAN TWEETS USING APACHE FLINK

Name	: Akbar Noto Ponco Bimantoro
NRP	: 05111640000028
Department	: Informatics FTEIC-ITS
Supervisor I	: Abdul Munif, S.Kom., M.Sc.
Supervisor II	: Nurul Fajrin Ariyani, S.Kom., M.Sc.

ABSTRACT

The development of easy internet access has made social media very popular. Society tends to use social media like facebook, instagram, twitter, and other similar media as a plac to express opinions. This invites companies or organizations to participate in using social media to understand their consumers. Thus, opening opportunities to information access that can be obtained like public opinions on organizations, companies, or products. One interesting anaylsis used on Twitter is detecting user complaints such as network disturbances in telecommunications service providers.

Network problems is one of crucial problems that needs to be addressed as soon as possible so that it requires realtime data processing. With realtime data processing, information related to those interference can be obtained as early as possible. If someone feels that the provider doesn't reponse their network outage properly, then it is likely that the person will move to another provider with similar services.

Therefore, this final project will create an application that can classify types of user complaints to detect network disruptions in telecommunication service provider through social media Twitter.

In developing the system using Apache Flink, a classification model must be failure tolerant (serializable). This

can be achieved by declaring the SVM model in a State Class. Then, a preprocessed datastreams are classified and verified based on similar events with adjacent places and time. The results of the classification are displayed altogether with the results of important entities such as Named Entitiy Recognition and place extraction in a REST API.

The application is created by using Apache Flink Standalone version on a i7-4720HQ computer. The system can classify 3 datastream in 1 to 2.5 seconds with 43% of accuracy. From the experiment conducted, from 5580 tweets there are 70 trouble detected.

Keywords: Complaint Detection, Realtime, Tweet, Apache Flink, Streaming.

KATA PENGANTAR

Bismillaahirrahmaanirrahim.

Alhamdulillahirabbil'alamin, segala puji bagi Allah subhaanahu wa ta'ala atas segala karunia dan rahmat-Nya penulis dapat menyelesaikan tugas akhir yang berjudul :

"Deteksi Keluhan secara Realtime pada Tweet Berbahasa Indonesia Menggunakan Apache Flink"

Harapan dari penulis semoga apa yang tertulis di dalam buku tugas akhir ini dapat bermanfaat bagi pengembangan ilmu pengetahuan saat ini, serta dapat memberikan kontribusi yang nyata.

Dalam pelaksanaan dan pembuatan Tugas Akhir ini tentunya sangat banyak bantuan yang penulis terima dari berbagai pihak, tanpa mengurangi rasa hormat penulis ingin mengucapkan terima kasih sebesar-besarnya kepada:

1. Ayah dan Ibu penulis, M. Hatta dan Sriana yang tiada hentinya memberikan dukungan doa, moral, dan material kepada penulis sehingga penulis dapat menyelesaikan Tugas Akhir ini.
2. Bapak Abdul Munif, S.Kom., M.Sc. dan Ibu Nurul Fajrin Ariyai S.Kom., M.Sc. selaku pembimbing I dan II yang telah membimbing dan memberikan motivasi, nasihat dan bimbingan dalam menyelesaikan Tugas Akhir ini.
3. Segenap dosen dan karyawan Departemen Informatika ITS yang telah memberikan ilmu dan pengalaman kepada penulis selama menjalanai masa kuliah di Informatika ITS.
4. Desy Nilasari, Denise Sonia Rahmadina, dan Fadilla Sukma Alfiani sebagai teman seperjuangan tugas akhir yang selalu memberi semangat dan bantuan dalam pengerjaan tugas akhir ini.
5. Muhammad Isa Senoaji sebagai rekan kerja yang selalu memberikan bantuan mental dan materiil ketika penulis mengalami masalah finansial selama pengerjaan Tugas Akhir.

6. Serta semua pihak yang telah turut membantu penulis dalam menyelesaikan Tugas Akhir ini.

Bagaimanapun juga penulis telah berusaha sebaik-baiknya dalam menyelesaikan Tugas Akhir ini. Namun, penulis mohon maaf apabila terdapat kekurangan ataupun kesalahkan yang penulis lakukan. Kritik dan saran yang membangun dapat disampaikan sebagai bahan perbaikan untuk ke depannya.

Surabaya, 23 Juli 2020

Akbar Noto Ponco Bimantoro

DAFTAR ISI

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR KODE SUMBER

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

1.1. Latar Belakang

Menurut data perusahaan Wearesocial, pengguna aktif media sosial di Indonesia mencapai 150 juta orang. Jumlah tersebut sama dengan jumlah pengguna internet di Indonesia [1]. Hal ini menandakan 1 dari 1 masyarakat Indonesia yang menggunakan internet merupakan pengguna media sosial. Masyarakat cenderung menggunakan media sosial seperti Facebook, Instagram, Twitter, dan media lain yang sejenis sebagai tempat mengekspresikan opini. Hal ini mengundang perusahaan atau organisasi untuk ikut serta menggunakan sosial media sebagai sarana pendekatan kepada pelanggan, sehingga membuka peluang akses informasi yang didapatkan, terutama opini publik terhadap organisasi, perusahaan, atau produk. Salah satu analisa yang dapat digunakan pada media sosial seperti Twitter adalah mendeteksi keluhan pengguna seperti gangguan jaringan pada perusahaan penyedia layanan telekomunikasi.

Permasalahan jaringan pada penyedia jaringan seperti layanan telekomunikasi merupakan salah satu masalah yang cukup krusial. Bila seseorang merasakan gangguan jaringan yang tidak disertai dengan penanganan yang cepat dan baik, maka mungkin orang tersebut akan lebih memilih untuk pindah ke penyedia jasa serupa. Sehingga memerlukan pemrosesan data secara *realtime* agar informasi terkait gangguan didapatkan sedini mungkin.

Data tweet yang didapatkan dari Twitter, dapat dijadikan dasar untuk mendeteksi gangguan layanan seperti gangguan pada jaringan di beberapa daerah dan permasalahan pada produk yang disediakan. Namun, data tweet yang tidak terstruktur mengakibatkan data tweet banyak mengandung kata tidak baku, ditambah kemungkinan makna tweet yang tidak relevan dengan kejadian saat ini seperti kejadian lampau atau tidak sedang terjadi. Sehingga perlu dibuat model klasifikasi untuk memastikan apakah tweet memberikan informasi tentang adanya gangguan jaringan.

Oleh karena itu dibuatlah sistem dengan memanfaatkan teknologi komputasi berbasis *streaming* seperti Apache Flink agar analisis dapat dilakukan secara *realtime* sehingga suatu organisasi bisa mendapatkan informasi dini yang akurat mengenai keluhan pengguna dan ada atau tidaknya gangguan jaringan.

1.2. Rumusan Permasalahan

Rumusan masalah yang diangkat dalam Tugas Akhir ini adalah sebagai berikut:

1. Bagaimana cara membedakan kategori keluhan gangguan jaringan dengan keluhan terkait produk layanan?
2. Bagaimana cara merancang sistem deteksi gangguan secara *real-time* menggunakan Apache Flink?
3. Bagaimana performa Apache Flink dalam mengklasifikasikan *datastream?*
4. Bagaimana cara mendeteksi ada atau tidaknya gangguan?
5. Bagaimana cara mendeteksi terjadinya lokasi gangguan jaringan?

1.3. Batasan Permasalahan

Permasalahan yang dibahas dalam Tugas Akhir ini memiliki beberapa batasan, diantaranya sebagai berikut:

1. Data yang digunakan sebagai bahan analisis adalah tweet berbahasa Indonesia yang ditujukan kepada akun *customer service* penyedia layanan telekomunikasi seperti Telkomsel (@Telkomsel), Indosat (@IndosatCare), Smartfren (@smartfrencare), dan akun penyedia lainnya apabila diperlukan.
2. Aplikasi yang digunakan untuk menganalisa data secara *real-time* adalah Apache Flink.
3. Kategori keluhan yang akan dideteksi ada 3, yaitu:
    a. Gangguan jaringan seperti koneksi yang lambat dan tidak ada sinyal

b. Permasalahan pada layanan seperti pertanyaan seputar produk. Contohnya tidak bisa melakukan transaksi pembelian paket internet.
c. Permasalahan selain produk dan jaringan dikategorikan sebagai keluhan umum atau lain-lain.

1.4. Tujuan

Tujuan dari pembuatan Tugas Akhir ini adalah merancang sistem menggunakan Apache Flink yang dapat memberikan informasi kepada penyedia layanan telekomunikasi terkait keluhan sesuai dengan kategori permasalahan dan mendeteksi ada atau tidaknya gangguan jaringan beserta lokasi terjadinya gangguan.

1.5. Manfaat

Tugas akhir ini diharapkan dapat membantu suatu organisasi untuk mendapatkan informasi *realtime* terkait keluhan pengguna dan ada atau tidaknya gangguan jarigan beserta lokasi terjadinya gangguan.

1.6. Metodologi

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

1. **Penyusunan proposal tugas akhir**

   Tahap awal untuk mengerjakan Tugas Akhir adalah penyusunan proposal Tugas Akhir. Pada proposal ini, penulis mengusulkan gagasan untuk mendeteksi gangguan jaringan pada *real-time* tweet berbahasa Indonesia menggunakan Apache Flink.

2. **Studi literatur**

   Studi literatur yang dilakukan dengan pengumpulan informasi mengenai apa saja yang bisa dijadikan referensi dalam pengerjaan Tugas Akhir. Mengumpulkan informasi dan studi literatur mengenai *text mining*, dan aplikasi terkait pengerjaan Tugas Akhir seperti Apache Flink dan Apahce Kafka. Informasi didapatkan dari buku, paper, jurnal, dan materi perkuliahan yang berkaitan dengan topik Tugas Akhir.

3. **Implementasi**

   Implementasi yang akan dilakukan yaitu perancangan sistem berdasarkan studi literatur dan pengumpulan informasi yang diperlukan. Impementasi ini dilakukan dengan menggunakan *library* Sastrawi, spaCy, dan Gensim untuk melakukan *pre-processing* data dan ekstraksi fitur. Serta Tweepy untuk memudahkan dalam mendapatkan data Tweet.

4. Pengujian dan evaluasi

   Tahap pengujian dan evaluasi dilakukan menggunakan metode *cross-validation* terhadap data uji yang didapatkan untuk mengetahui hasil dan performa metode yang telah dibangun. Evaluasi dan perbaikan akan dilakukan hingga perangkat lunak yang diuji menghasilkan hasil performa yang sesuai dengan data uji permasalahan tersebut.

5. Penyusunan buku Tugas Akhir

   Pada tahap ini dilakukan proses dokumentasi dan pembuatan laporan dari seluruh konsep, tinjauan pustaka, metode, implementasi, proses yang telah dilakukan, pengujian, evaluasi dan hasil-hasil yang telah didapatkan selama pengerjaan tugas akhir.

1.7. Sistematika Penulisan

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, rumusan masalah, tujuan dan manfaat pembuatan tugas akhir, batasan masalah, metodologi yang digunakan, dan sistematika penyusunan tugas akhir.

Bab II **Tinjauan Pustaka**

Bab ini menjelaskan beberapa pustaka-pustaka yang dijadikan penunjang dan berhubungan dengan pokok pembahasan yang mendasari pembuatan tugas akhir.

Bab III **Desain dan Perancangan Sistem**

Bab ini membahas mengenai desain dan perancangan sistem yang akan dibangun.

Bab IV **Implementasi Sistem**

Bab ini membahas mengenai bagaimana implementasi sistem dari desain yang sudah dirancang.

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 dan Saran**

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
TINJAUAN PUSTAKA

Bab ini menjelaskan tentang tinjauan pustaka yang menjadi dasar pembuatan tugas akhir. Beberapa teori, pustaka, dan teknologi yang mendasari pengerjaan tugas akhir ini diantaranya meliputi definisi Twitter, *text processing*, pembobotan model Doc2Vec, dan lain sebagainya. Penjelasan secara khusus masing-masing tinjauan pustaka dapat dilihat pada masing-masing subbab berikut ini.

2.1. Penelitian Terkait

Terdapat beberapa penelitian yang telah dilakukan sebelumnya untuk mendeteksi kejadian pada media sosial. Beberapa contoh penelitian tersebut adalah mendeteksi gempa [2] dan mendeteksi kecelakaan [3]. Kaitan kedua penelitian tersebut yaitu mendeteksi kejadian pada media sosial twitter. Akan tetapi, keduanya memiliki implementasi dan pendekatan yang berbeda. Perbedaan tersebut dijabarkan pada Tabel 2.1.

Tabel 2.1Benchmark penelitian sebelumnya

	Deteksi Gempa	**Deteksi Kecelakaan**	**Tugas Akhir**
Tujuan	Mendeteksi Gempa	Mendeteksi Kecelakaan	Mendeteksi Keluhan dan Gangguan Jaringan
Data Latih	Data diambil dari Twitter dan diberi label secara manual	Data diambil dari Twitter dan diberi label secara manual	Data diambil dari Twitter dan diberi label secara manual
Pre-processing	Tokenisasi; menghapus entitas kurang penting seperti	*Case folding*, menghapus simbol; normalisasi	*Case folding*; menghapus simbol, angka, URL, mention,

	simbol dan angka; *stopword removal*; normalisasi	*slang;* stopword removal; tokenisasi	dan *hashtag*; normalisasi *slang*; *stemming*
Ekstraksi Fitur	Mengabaikan relasi kata dengan *Term Frequency*	Menggunakan Word Embedding Word2Vec	Menggunakan pengembangan Word2Vec, yaitu Doc2Vec
Metode Klasifikasi	Membandingkan performa berbagai macam algoritma klasifikasi seperti: Decision Tree, Random Forest, dan SVM	Menggunakan *Recurrent Neural Network*	Menggunakan SVM karena terbatasnya pustaka pada versi Apache Flink yang digunakan
Deteksi Lokasi	-	Geotag dan NER yang diverifikasi menggunakan korpus lokasi	NER, POSTAG, dan Rule Based Matching yang diverifikasi menggunakan korpus lokasi
Verifikasi Deteksi Kejadian	-	Menggunakan *hash-map* sehingga ketika terjadi *hash collision* akibat tweet dengan kemiripan yang tinggi akan dibuang / dianggap duplikat	Menghitung jumlah keluhan jaringan pada lokasi yang sama dengan kurun waktu tertentu

Metode Pengujian	*Cross Validation*	*Cross Validation;* Uji coba *streaming*	*Cross Validation*; Uji coba *streaming*
Alat	Weka	Tidak dijelaskan secara rinci	Apache Flink

2.2. Twitter

Twitter adalah media sosial yang memberikan layanan kepada penggunanya untuk saling berbagi kejadian dalam sebuah pesan pendek dengan maksimal 280 karakter yang disebut tweet [4]. Tweet memungkinkan penggunanya untuk mengekspresikan opini atau berkomunikasi dengan pengguna lainnya secara *real-time*. Contohnya ketika ada permasalahan pada jaringan telekomunikasi, banyak pengguna yang mengirimkan pertanyaan kepada akun penyedia layanan telekomunikasi terkait gangguan tersebut. Hal ini dapat dimanfaatkan sebagai sarana deteksi dini gangguan jaringan dengan mengklasifikasikan tweet yang masuk kedalam Twitter. Contoh gangguan jaringan yang umum terjadi adalah jaringan yang lambat, tidak ada sinyal atau tidak bisa mengakses beberapa layanan seperti tidak bisa membeli paket internet, dan lain sebagainya.

> Giphar, Om (Gifar_ALfaqih). "@IndosatCare min, sinyal di Sindang laut kecamatan Lemahabang kabupaten cirebon lemot banget ya? Pake 3g atau 4g sama aja, bukan saya aja min yang lain juga, tolong min respon ya" 6 Juni 2019, 22.20 WIB. Tweet

Sebagai contoh pada tweet diatas, mengandung makna bahwa sekarang sedang terjadi gangguan jaringan berupa koneksi yang lambat di kecamatan Lemahabang, sekitar Cirebon.

> Jayanti, Devi (devijayantie). "@Telkomsel kenapa setelah upgrade ke 4G, paket internet di *363# tidak bisa dibeli, misalkan paket yang 75rbu 15GB, atau paket lainnya dimenu Internet Lainnya, tidak bisa saya beli☹" 30 Agustus 2019, 09.50 WIB. Tweet.

Contoh lainnya adalah pada tweet devijayantie yang menanyakan kejelasan terkait permasalahan kesulitan atau tidak bisa membeli produk berupa paket internet.

> Masyud (ycpgg). "@IndosatCare iya mas Maxi. Ngasih tau temen-temen aja. Kebetulan sinyal Telkomsel di kosan saya uelek. Sabar :)" 12 Juni 2019, 17.51 WIB. Tweet

> (satrnst). "@Telkomsel min kemarin di daerah karanganyar sinyalnya kayak kurang stabil ya, tapi sekarang udah lancar" 13 Juni 2019, 06.50 WIB. Tweet

Sedangkan pada kedua tweet diatas tidak memiliki makna sedang ada gangguan sinyal, akan tetapi lebih bersifat informatif bahwa di daerah tersebut pernah mengalami gangguan jaringan atau gangguan telah teratasi meskipun memiliki beberapa kata kunci seperti "sinyal", "kurang stabil", dan "uelek" atau jelek. Sehingga, perlu sebuah metode untuk mengolah data tweet agar mendapatkan deteksi yang akurat.

2.3. Profil Provider

Secara umum Indosat, Telkomsel, dan Smartfren memiliki produk yang serupa yaitu layanan prabayar, pascabayar, dan konektivitas internasional. Layanan prabayar merupakan layanan yang mengharuskan penggunanya untuk membayar terlebih dahulu untuk dapat menggunakan layanan yang disediakan seperti paket Yellow pada Indosat dan paket SimpatiTAU pada Telkomsel.

Sedangkan layanan pascabayar memungkinkan penggunanya untuk menikmati layanan terlebih dulu lalu dibayarkan kemudian. Contohnya adalah IM3 Ooredoo Pascabayar [5], Kartu Halo [6], Smartfren Postpaid [7].

Masing-masing penyedia layanan tersebut juga memiliki sosial media untuk berkomunikasi kepada pelanggannya seperti instagram, facebook, dan twitter. Pada tugas akhir ini, media sosial yang akan dijadikan bahan uji coba adalah akun twitter *customer service* milik Indosat (@IndosatCare), Telkomsel (@telkomsel), Smartfren (@smartfrencare).

2.4. *Text Processing*

Text processing merupakan tahap awal dimana sistem melakukan seleksi data yang akan di proses untuk mempersiapkan data agar menjadi lebih terstruktur. Berikut merupakan beberapa tahapan dalam *text processing*.

2.4.1. Case Folding

Case Folding adalah sebuah teknik menyamakan gaya penulisan teks dokumen. Semua kalimat pada teks pada dokumen akan diubah menjadi huruf kecil atau menjadi huruf besar. Pada tahap ini, dokumen berupa tweet yang telah didapatkan akan diubah menjadi gaya penulisan yang sama, yaitu semua teks akan diubah menjadi huruf kecil.

2.4.2. Penghapusan Simbol dan Angka

Penghapusan simbol merupakan salah satu tahap *pre-processing* pada *Natural Language Processing*. Cara kerja penghapusan simbol yaitu memfilter / menghapus simbol-simbol pada teks.

2.4.3. Penghapusan URL, Mention, dan Hashtag

Tautan luar dalam URL, mention, dan hastag dalam *tweet* dianggap sebagai informasi yang tidak memiliki arti. Oleh karena itu tautan luar dalam tweet yang berbentuk URL perlu dihapus.

2.4.4. Normalisasi Slang

Slang adalah sebuah kata atau frasa informal yang biasa digunakan oleh suatu kelompok untuk berkomunikasi atau mengekspresikan sesuatu. Contohnya “min” atau “mimin” berarti “admin”, “skuy” atau “kuy” atau “yuk” berarti “ayo”, dan “plis” berarti “tolong”. Tahap ini mengubah kata-kata *slang* menjadi kata yang lebih umum.

2.4.5. Stemming

Stemming merupakan sebuah tahap mengubah suatu token atau kata menjadi bentuk kata dasarnya dengan tujuan untuk mengurangi variasi pada dokumen yang akan diolah.

2.5. POS Tagging

Part of Speech (POS) *Tagging* adalah proses memberi label pada setiap kata dalam kalimat dengan POS atau *tag* yang sesuai dengan kelas kata seperti kata benda, kata kerja, kata keterangan, kata sifat, dan lainnya [8]. Pada tugas akhir ini POS *Tagging* digunakan untuk mencari kelas tiap kelas dari teks *tweet* untuk digunakan dalam proses pencarian lokasi dan *Rule-based Matching*.

2.6. Named Entity Recognition

NER (*Named Entity Recognition*) merupakan salah satu tahap ekstraksi data yang mengklasifikasikan teks atau dokumen yang tidak terstruktur menjadi beberapa kategori yang telah ditentukan sebelumnya, seperti lokasi, perusahaan, kota, dan lain sebagainya. Tujuan yang diharapkan dari proses dalam NER adalah untuk melakukan ekstraksi dan klasifikasi nama ke dalam beberapa kategori dengan mengacu kepada makna yang tepat [9]. Contohnya NER akan mengekstrak entitas lokasi dari sebuah tweet “internet lambat di rumah saya perumahan btn rejomulyo 2 / 57” yaitu “btn rejomulyo 2/57”.

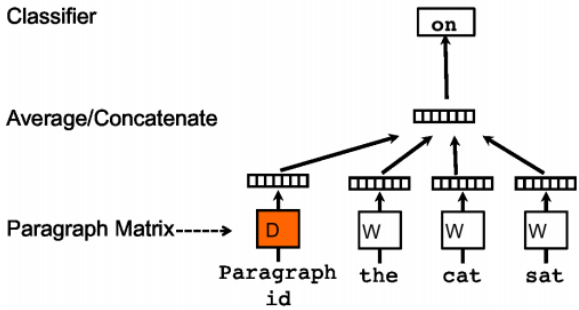
2.7. *Rule-based Matching*

Rule-based Matching membantu pengguna untuk mencocokkan token, frasa, entitas kata, dan kalimat berdasarkan beberapa pola yang ditentukan. Proses ini dilakukan bersama

dengan fitur lain seperti *part-of-speech* [10]. Pada tugas akhir ini *Rule-based Matching* digunakan untuk mencari entitas lokasi dari teks tweet berdasarkan aturan atau pola yang telah dibuat.

2.8. Doc2Vec

Doc2Vec adalah sebuah konsep yang bertujuan untuk merepresentasikan dokumen menjadi bentuk angka sehingga dapat diolah dan dipahami oleh komputer. Doc2Vec merupakan perbaikan dari Word2Vec dengan menambahkan satu input layer berupa ID dokumen seperti yang dapat dilihat pada Gambar 2.1 [11]. Keuntungan dari Doc2Vec diklaim lebih baik dari *bag-of-words* karena dapat menyimpan *semantic* dari kata layaknya Word2Vec. Contohnya, kata "kuat" memiliki keterkaitan dengan kata "tangguh" yang lebih erat dibandingkan kata "lemah". Selain itu, Doc2Vec juga menyimpan urutan kata. Konsep ini pertama kali diusulkan oleh Quoac Le dan Tomas Mikolov.

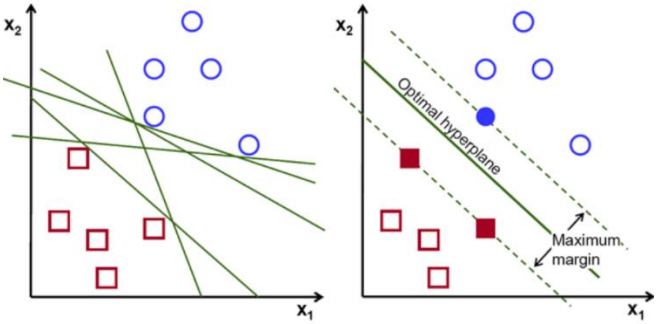


Gambar 2.1 Konsep Doc2Vec

2.9. Support Vector Machine

Support Vector Machine (SVM) adalah metode pembelajaran *supervised* yang menganalisis data dan pola yang biasa digunakan untuk mengklasifikasikan data menjadi beberapa kategori atau kelas tertentu. SVM bekerja dengan memetakan data latih yang telah diberi label menjadi titik-titik dapa ruang. Pemetaan data dipisahkan oleh sebuah garis pemisah sehingga memisahkan data menjadi beberapa kelas seperti Gambar 2.2.

Garis pemisah yang digunakan adalah garis yang memiliki jarak maksimal antar data di tiap tiap kelas. Garis tersebut disebut *hyperplane.*



Gambar 2.2 Contoh gambar Support Vector Machine

2.10. Decision Tree

Decision Tree adalah sebuah metode klasifikasi untuk mendapatkan kesimpulan dari sejumlah data berdasarkan aturan-aturan yang ditentukan [12]. Model Decision Tree berbentuk hierarki pohon seperti akar, batang, dan daun yang merepresentasikan keputusan yang dibuat. Contoh decision tree dapat dilihat pada Gambar 2.3.

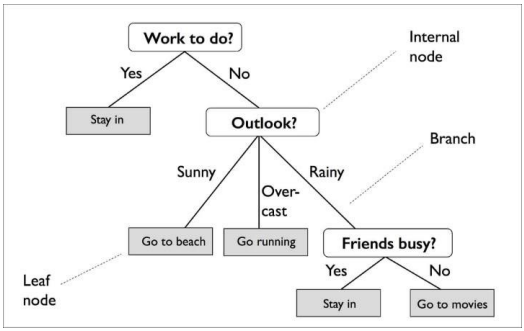
2.11. Tweepy

Tweepy adalah sebuah pustaka python yang digunakan untuk mengakses API Twitter [13]. Tweepy memberikan fitur yang cukup lengkap mulai dari filter atau search hingga *stream listener*. Tweepy akan digunakan sebagai alat untuk mempermudah akses API pada Twitter.

2.12. Sastrawi Python

Sastrawi Python adalah sebuah pustaka online yang digunakan untuk mengubah kata-kata Bahasa Indonesia kedalam bentuk dasarnya. Sastrawi Python dibuat sebagai penghubung

program dan translasi pustaka asli Sastrawi yang menggunakan bahasa pemrograman PHP [14].



Gambar 2.3 Contoh Decision Tree

2.13. Prodigy

Prodigy merupakan alat anotasi yang efisien sehingga penggunanya dapat melakukan anotasi sendiri. Prodigy dapat membantu pengguna dalam melatih dan mengevaluasi model dengan lebih cepat. Fitur yang ada dalam Prodigy yaitu *Named Entity Recognition*, klasifikasi teks, dan visi komputer [15]. Pada tugas akhir ini Prodigy digunakan sebagai alat bantu untuk melakukan anotasi NER pada tweet.

2.14. SpaCy

SpaCy adalah pustaka *open-source* gratis dalam bahasa Python yang digunakan untuk *Natural Language Processing* (NLP). SpaCy dapat digunakan untuk membangun ekstraksi informasi atau praproses teks untuk *artifical intelligence*. Beberapa fitur yang didukung spaCy antara lain tokenisasi, POS *Tagging*, *Named Entity Recognition*, dan *Rule-based Matching* [16]. Pada tugas akhir ini spaCy digunakan untuk melakukan anotasi NER, POS *Tagging*, dan *Rule-based Matching*.

2.15. FuzzyWuzzy

FuzzyWuzzy adalah pustaka *open source* dalam bahasa Python yang digunakan untuk otomasi pencarian dan pencocokan kata atau kalimat dengan menghitung jarak kata dengan algoritma Levenshtein. Cara kerja algoritma Levenshtein cukup sederhana, yaitu dengan menghitung berapa banyak perubahan huruf pada kalimat atau kata terhadap kalimat yang akan dicari dari korpus yang telah ada [17]. Pada tugas akhir ini, FuzzyWuzzy akan digunakan untuk membuat Gazetter pada subbab 3.4.1 untuk mencocokkan hasil ekstraksi lokasi dengan korpus data daerah-daerah di Indonesia.

2.16. Gensim

Gensim adalah pustaka *open-source* gratis dalam bahasa Python yang digunakan untuk *unsupervised topic modelling* dan *Natural Language Processing* menggunakan kecerdasan komputasional [18]. Pada tugas akhir ini, Gensim digunakan untuk membangun model Doc2Vec untuk melakukan ekstraksi fitur pada teks tweet.

2.17. Sklearn

Sklearn (sci-kit learn) adalah sebuah pustaka *machine learning* online berbahasa Python yang memiliki banyak model klasifikasi, regresi dan *clustering*. Selain itu sklearn juga mendukung *preprocessing* seperti penyederhanaan dimensi dan seleksi model [19]. Pada tugas akhir ini sklearn digunakan untuk menyimpan model data LIBSVM kedalam bentuk file sehingga dapat digunakan oleh Apache Flink.

2.18. Scipy

Scipy adalah pustaka *open-source* berbahasa Python yang digunakan untuk komputasi sains. Scipy mendukung fitur seperti aljabar linear, optimasi, pemrosesan sinyal, pemrosesan gambar, dan lain sebagainya [20]. Pada tugas akhir ini, Scipy digunakan untuk mengubah bentuk *array* menjadi format data yang dapat dikenali oleh Apache Flink, yaitu matrix csr.

2.19. Twitter Text Python

Twitter Text Python (TTP) adalah pengurai teks *tweet* untuk mendapatkan *hashtag*, *user*, dan URL. Selain itu, TTP juga dapat membentuk entitas-entitas pada *tweet* sebagai HTML untuk ditampilkan [21]. Pada tugas akhir ini, TTP digunakan untuk melakukan membersihkan entitas yang dianggap tidak penting seperti URL dan mendapatkan entitas user untuk pemrosesan lebih lanjut.

2.20. Flask

Flask merupakan salah satu kerangka kerja aplikasi web Python yang paling populer. Flask memberikan kebebasan kepada pengembang untuk memilih alat dan pustaka yang ingin digunakan. Serta ada banyak ekstensi yang disediakan sehingga penambahan fungsionalitas baru menjadi lebih mudah [22]. Pada tugas akhir ini Flask digunakan untuk membuat aplikasi web.

2.21. MongoDB

MongoDB adalah sebuah *database* berbasis dokumen beformat JSON yang dibuat oleh MongoDB Inc [23]. Terdapat beberapa cara untuk menggunakan MongoDB, yaitu dengan menggunakan distribusi API seperti PyMongo atau berinteraksi dengan GUI menggunakan aplikasi MongoDB Compass [24]. Pada tugas akhir ini, MongoDB dipilih untuk memudahkan penyimpanan data *tweet* dari Tweepy dan sistem aplikasi yang akan dibuaat karena memiliki format yang sama.

2.22. PyMongo

PyMongo adalah pustaka online berbahasa Python yang digunakan untuk mengakses MongoDB. Selain itu, PyMongo merupakan pustaka Python yang direkomendasikan oleh MongoDB [25]. Fitur yang disediakan PyMongo cukup lengkap untuk melakukan CRUD (*create, read, update, delete*) data. Pada tugas akhir ini, PyMongo digunakan sebagai penghubung antara sistem aplikasi yang akan dibuat dengan basis data MongoDB.

2.23. Apache Kafka

Apache Kafka adalah platform *streaming* terdistribusi yang biasa digunakan untuk menghubungkan antar sistem atau aplikasi [26]. Cara kerjanya cukup sederhana, yaitu dengan berlanggan atau *subscribe* pada *broker* atau topik yang disediakan oleh *producer* yang telah dibuat sebelumnya. Pada tugas akhir ini, Apache Kafka digunakan untuk menghubungkan aliran data antar sistem yang akan dibuat.

2.24. Pykafka

Pykafka merupakan sebuah pustaka online berbahasa Python yang digunakan untuk mengakses API kafka *client* [27]. Fitur utama PyKafka adalah akses *KafkaProducer* dan *KafkaConsumer* yang memungkinkan pengguna untuk menghubungkan aliran data satu platform dengan platform lain menggunakan bahasa Python.

2.25. Apache Flink

Apache Flink merupakan sebuah platform kerangka kerja *open source* untuk pemrosesan data terdistribusi baik berbasis *datastream* maupun *batch* [28]. Berbeda dengan Apache Spark, Apache Flink berjalan menggunakan memori dan didesain untuk dapat menjalankan aplikasi berskala kecil maupun besar dengan toleransi kesalahan dan manajemen memori yang dioptimisasi. Pada tugas akhir ini, Apache Flink digunakan untuk mengklasifikasikan data *tweet* menggunakan model LIBSVM yang disediakan.

2.26. *Confusion Matrix*

Confusion matrix (CF) merupakan salah satu metode yang dapat digunakan untuk mengukur kinerja suatu metode klasifikasi. Pada dasarnya, *confusion matrix* mengandung informasi yang membandingkan hasil klasifikasi yang dilakukan oleh sistem dengan hasil klasifikasi yang seharusnya.

Pada pengukuran kinerja menggunakan *confusion matrix*, terdapat 4 (empat) istilah sebagai representasi hasil proses klasifikasi. Keempat istilah tersebut adalah *True Positive* (TP),

True Negative (TN), *False Positive* (FP), dan *False Negative* (FN) [29].

True Positive (TP) menyatakan jumlah kebenaran antara hasil klasifikasi dengan jumlah seluruh data. *True Negative* (TN) menyatakan jumlah dari kesamaan hasil klasifikasi dan yang sesungguhnya adalah salah. *False Positive* (FP) menyatakan jumlah hasil klasifikasi yang diindikasikan benar, tetapi sesungguhnya salah. *False Negative* (FN) menyatakan jumlah dari hasil klasifikasi yang diindikasikan salah, tetapi sesungguhnya benar. Adapun kriteria benar dan salah pada sistem deteksi lokasi tugas akhir ini dapat dilihat pada Tabel 2.2. Sedangkan kriteria kebenaran sistem klasifikasi pada model klasifikasi dapat dilihat pada Tabel 2.3.

Tabel 2.2 Kriteria penilaian confusion matrix pada deteksi lokasi

Nilai	**Kriteria**	**Contoh**	**CF**
Benar	Apabila hasil deteksi lokasi sesuai dengan tempat yang dicantumkan (jika ada) pengguna pada tweet keluhannya	Min, lagi gangguan ya? Deteksi: -	TN
		Min, di surabaya lagi gangguan ya? Deteksi: Surabaya	TP
Salah	Apabila hasil deteksi lokasi tidak sesuai dengan tempat yang dicantumkan (jika ada) pengguna pada tweet.	Min, lagi gangguan ya? Deteksi: Surabaya	FP
		Min, di surabaya lagi gangguan ya? Deteksi: -	FN
		Min, di surabaya lagi gangguan ya? Deteksi: Malang	FP

Tabel 2.3 Kriteria penilaian confusion matrix pada model klasifikasi

Nilai	**Kriteria**	**Contoh**	**CF**	
			Jaringan	**Produk**
Benar	Apabila hasil klasifikasi sesuai dengan konteks keluhan pengguna	Min, jaringannya kok lemot ya? Konteks: Jaringan Klasifikasi: Jaringan	TP	TN
		Min, cara aktivasi paket promo 100K gimana ya? Konteks: Produk Klasifikasi: Produk	TN	TP
Salah	Apabila hasil klasifikasi tidak sesuai dengan konteks keluhan pengguna	Min, jaringannya kok lemot ya? Konteks: Jaringan Klasifikasi: Produk	FN	FP
		Min, cara aktivasi paket promo 100K gimana ya?	FP	FN

		Konteks: Produk Klasifikasi: Jaringan		

Metrik evaluasi *accuracy*, *precision*, *recall*, dan *F-Measure* secara lebih rinci dijelaskan sebagaimana berikut:

a. *Accuracy*

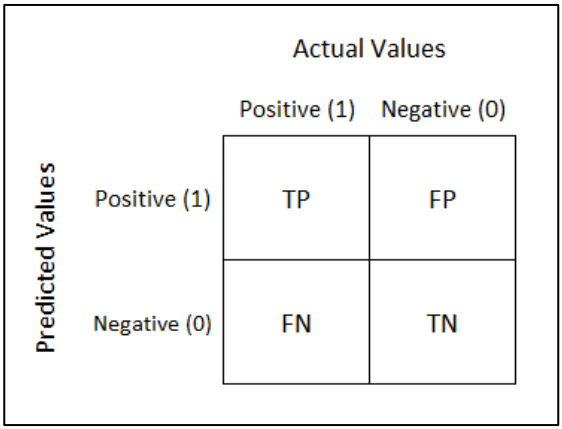
Accuracy adalah adalah nilai rasio data yang diklasifikasikan benar dari jumlah total data.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{2.15}$$

b. *Precision*

Precision adalah nilai total data positif yang diklasifikasikan dengan benar dibagi dengan hasil prediksi data positif.

$$Precision = \frac{TP}{TP + FP} \tag{2.16}$$



Gambar 2.4 Confusion Matrix

c. *Recall*

Recall adalah nilai total data positif yang diklasifikasikan dengan benar dibagi dengan jumlah data positif.

$$Recall = \frac{TP}{TP + FN} \tag{2.17}$$

d. *F-Measure*

F-Measure adalah nilai yang didapatkan dari *precision* dan *recall* menggunakan *Harmonic Mean*.

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{2.18}$$

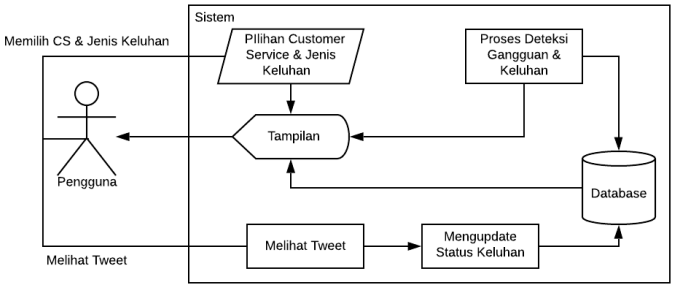
Kemudian, hasil dari masing-masing perspektif pada setiap kelas dirata-rata untuk mendapatkan hasil akhir dari *accuracy*, *precision*, *recall*, dan *F-Measure*.

BAB III
DESAIN DAN PERANCANGAN SISTEM

Bab ini membahas tentang desain dan perancangan aplikasi deteksi keluhan dan gangguan secara *real-time*. Pembahasan yang akan dilakukan meliputi analisis fitur yang dibutuhkan dan perancangan perangkat lunak.

3.1. Analisis Metode Secara Umum

Pada tugas akhir ini akan dibangun sebuah aplikasi yang dapat memberikan informasi kepada penyedia layanan telekomunikasi terkait keluhan pelanggan dan deteksi gangguan jaringan dengan menggunakan pengolahan data secara *real-time* pada media sosial Twitter.



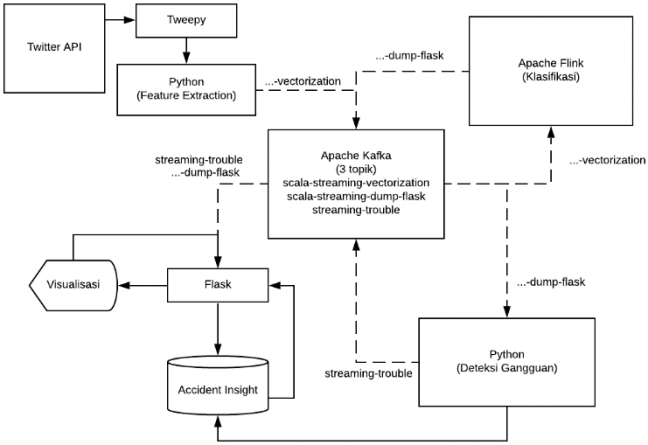
Gambar 3.1 Gambaran interaksi sistem dengan pengguna

Proses pengolahan data pada implementasi sistem ini meliputi pengambilan data tweet, pra proses teks, pembuatan model *Part-of-Speech* (POS) *Tagger* Bahasa Indonesia, pembuatan Model *NER (Named Entity Recognition)* Bahasa Indonesia, pembuatan model pembobotan model Doc2Vec, pembuatan model klasifikasi, dan lain sebagainya. Alur interaksi sistem dengan pengguna secara umum dapat dilihat pada Gambar 3.1. Arsitektur sistem dapat dilihat pada Gambar 3.2 dan diagram alir seluruh tahapan dapat dilihat pada Gambar 3.3.

Pada arsitektur sistem digambarkan bahwa *datastream* berupa tweet akan diolah untuk mendapatkan tweet bersih beserta

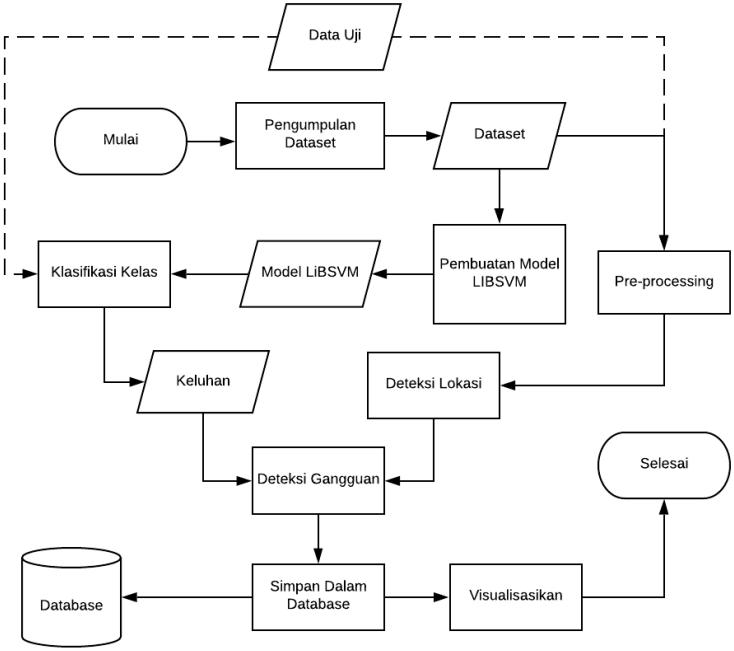
fitur dalam bentuk *vector*. Selanjutnya data hasil pengolahan berupa tweet, *processed*-tweet, *vector*, lokasi, dan detail lainya dikirimkan untuk diklasifikasikan menggunakan Apache Flink melalui Kafka dalam bentuk JSON. Setelah itu, hasil klasifikasi akan diteruskan kembali melalui Kafka dengan topik penghubung yang berbeda, *scala-streaming-dump-flask*, untuk diproses dan ditampilkan oleh masing-masing *subscriber*.

Model-model yang dipakai dalam pengolahan data seperti model NER, POSTAG, dan SVM telah dibuat terlebih dahulu menggunakan data tweet yang telah dikumpulkan sebelumnya



Gambar 3.2 Arsitektur sistem

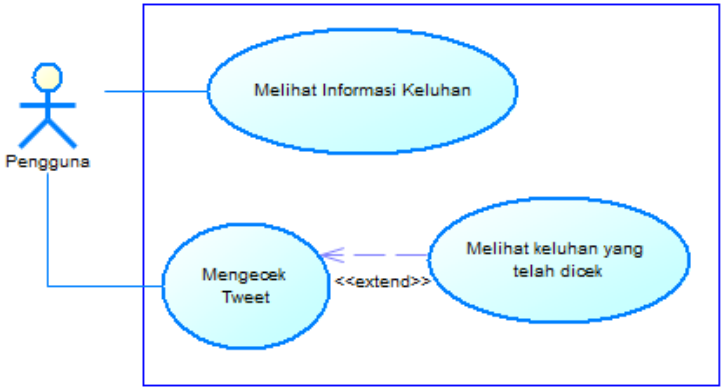
seperti yang telah digambarkan pada diagram alir seluruh tahapan. Lokasi terjadinya gangguan keluhan diekstrak menggunakan kombinasi model NER, POSTAG, dan *Rule Based Matching* (RBS) dengan mengambil perolehan suara terbanyak atau voting.



Gambar 3.3 Diagram alir seluruh tahapan

3.2. Kasus Penggunaan Sistem

Diagram kasus pengunaan dapat dilihat pada Gambar 3.4. Terdapat satu aktor yang terlibat dan berinteraksi langsung dengan sistem. Pengguna merupakan karyawan penyedia layanan *customer service* yang berasal dari berbagai profesi ataupun latar belakang yang berbeda-beda. Selain aktor, pada diagram tersebut juga digambarkan empat kasus penggunaan yang dijabarkan pada Tabel 3.1.



Gambar 3.4 Diagram kasus penggunaan

Tabel 3.1 Daftar kasus penggunaan

Kode	Nama Kasus Penggunaan
UC01	Melihat Informasi Keluhan
UC02	Mengecek keluhan
UC03	Melihat keluhan yang telah dicek

3.2.1. Melihat Informasi Keluhan

Pada kasus penggunaan ini, pengguna dapat melihat informasi keluhan sesuai penyedia layanan telekomunikasi dan jenis keluhan yang dipilih. Penyedia telekomunikasi yang dapat dipilih adalah Telkomsel, Indosat, dan SmartFren. Halaman yang dapat dipilih adalah Dashboard, Jaringan, Produk, Produk & Jaringan, dan Lain-Lain. Pada halaman tersebut terdapat informasi mengenai keluhan pengguna. Spesifikasi kasus penggunaan dapat dilihat pada Tabel 3.2.

Tabel 3.2 Spesifikasi kasus penggunaan UC01

Kode Kasus Penggunaan	UC01
Nama Kasus Penggunaan	Melihat Informasi Keluhan
Aktor	Pengguna
Deskripsi	Pengguna dapat melihat informasi keluhan pengguna

	sesuai *provider* dan jenis keluhan
Relasi	*Directed Association*
Kondisi Awal	Pengguna sedang membuka salah satu halaman *website* sistem
Kondisi Akhir	Sistem menampilkan informasi keluhan
Alur Normal	
Aktor	Sistem
1. Pengguna memilih satu diantara beberapa penyedia layanan yang tersedia 2. Pengguna memilih satu halaman yang akan dituju	3. Sistem mencari data sesuai penyedia layanan dan halaman yang dipilih 4. Sistem tetap menampilkan laman informasi keluhan sesuai penyedia layanan dan jenis keluhan yang dipilih
Alur Alternatif	
-	
Eksepsi	
-	

3.2.2. Mengecek keluhan

Pada kasus penggunaan ini, pengguna dapat mengecek keluhan yang telah diklasifikasikan sehingga pengguna dapat merespon keluhan tersebut pada halaman asli Twitter. Spesifikasi kasus penggunaan dapat dilihat pada Tabel 3.3.

Tabel 3.3 Spesifikasi kasus penggunaan UC02

Kode Kasus Penggunaan	UC02
Nama Kasus Penggunaan	Mengecek keluhan
Aktor	Pengguna
Deskripsi	Pengguna dapat mengecek keluhan
Relasi	*Directed Association*
Kondisi Awal	Pengguna sedang membuka tab "Baru" pada halaman Jaringan, Produk, Jaringan & Produk, atau Lain-Lain.
Kondisi Akhir	Sistem mengubah status tweet menjadi "sudah dilihat"
Alur Normal	
Aktor	Sistem
1. Pengguna menekan tombol "Lihat di Twitter" pada keluhan yang akan dicek	2. Sistem membuka keluhan pada halaman asli twitter 3. Sistem mengubah status keluhan terpilih pada database
Alur Alternatif	
-	
Eksepsi	
-	

3.2.3. Melihat keluhan yang telah dicek

Pada kasus penggunaan ini, pengguna dapat melihat keluhan yang telah dicek. Spesifikasi kasus penggunaan dapat dilihat pada Tabel 3.4.

Tabel 3.4 Spesifikasi kasus penggunaan UC03

Kode Kasus Penggunaan	UC03
Nama Kasus Penggunaan	Melihat keluhan yang telah dicek
Aktor	Pengguna
Deskripsi	Pengguna dapat melihat kembali daftar keluhan yang telah dicek
Relasi	*Extend Association*
Kondisi Awal	Pengguna sedang membuka tab "Baru" pada salah satu diantara halaman Jaringan, Produk, Jaringan & Produk, atau Lain-Lain.
Kondisi Akhir	Sistem menampilkan daftar keluhan yang telah dicek
Alur Normal	
Aktor	Sistem
1. Pengguna menekan tombol "Sudah dilihat" pada keluhan yang akan dicek	2. Sistem mencari keluhan yang telah dicek 3. Sistem menampilkan daftar keluhan yang telah dicek
Alur Alternatif	
-	
Eksepsi	
-	

3.3. Perancangan Data

Pada subbab ini akan menjelaskan proses perancangan data. Data yang digunakan adalah data tweet pada media sosial Twitter yang ditujukan kepada pelayanan pelanggan (CS) penyedia

telekomunikasi, yaitu Telkomsel (@telkomsel), Indosat (@indosatcare), dan Smartfren (@smartfrencare). Data didapatkan dengan menggunakan Twitter *crawler*, Tweepy. Jumlah data yang digunakan sebagai model klasifikasi berjumlah sekitar 4900, adapun model NER menggunakan 1074 tweet yang dianotasi secara manual menggunakan Prodigy. Model POSTAG menggunakan dataset *Indonesian Manually Tagged Corp* yang dapat diunduh dan diakses secara bebas [30]. Contoh Tweet yang digunakan dapat dilihat pada Tabel 3.5.

Data hasil pemrosesan akan disimpan dengan format JSON pada No-SQL *database*. Contoh bentuk data yang akan disimpan pada database dapat dilihat pada Lampiran G:.

3.4. Perancangan Proses

Pada subbab ini akan dijelaskan mengenai perancangan proses yang dilakukan untuk setiap tahap pembuatan sistem tugas akhir ini berdasarkan analisis metode yang dapat dilihat pada Gambar 3.2 dan Gambar 3.3.

Tabel 3.5 Contoh data masukan Tweet

Tweet
@indosatcare kok jaringan im3 di kota medan ga stabil dan hilang hilang?
@telkomsel min, mau nanya, kalau paket yang ini voucher shellfirenya dapat digunakan didalam gamenya ya? https://t.co/tst3m6nogn
@indosatcare jadi gabisa belajar padahal besok ujiann:(
@telkomsel posted... https://t.co/jipfalkq1y
@smartfrencare terus klo saya beli voucher smartfren gosok per giga itu bisa ga min? atau paket saya yg unlimited bakal ilang?

3.4.1. *String Matching* Gazetter

Gazetter merupakan nama kelas yang penulis gunakan untuk membangun sistem pencocokan kalimat. Karena bentuk dan bahasa pada *tweet* yang beragam, pencocokan kalimat ini digunakan untuk mendapatkan hasil lokasi yang lebih formal berdasarkan korpus data daerah-daerah di Indonesia. Hasil

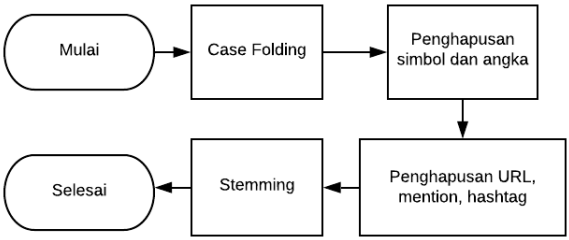
Gazetter ini nantinya digunakan sebagai hasil akhir ekstraksi lokasi pada tweet. Contoh rancangan masukan dan keluaran Gazetter dapat dilihat pada Tabel 3.6. Diagram alir proses pencocokan kalimat dengan Gazetter dapat dilihat pada Gambar 3.6.

Tabel 3.6 Contoh masukan dan keluaran Gazetter

Contoh *Input*	**Contoh *Output* Gazetter**
Kec kputih	Kecamatan Keputih
Srbya	Kota Surabaya

3.4.2. Preprocessing

Preprocessing adalah tahapan yang berisikan teknik-teknik untuk menormalkan data menjadi kedalam bentuk yang lebih seragam. Pada tuga akhir ini, teknik-teknik yang digunakan untuk melakukan *preprocessing* adalah *case folding*, penghapusan entitas yang dianggap kurang penting seperti simbol dan url, normalisasi *slang word*, serta *stemming*. *Preprocessing* dilakukan pada tahapan pembuatan model SVM dan klasifikasi kelas yang akan dibahas pada subbab 3.4.6 dan 3.4.7. Adapun diagram alir *preprocessing* dapat dilihat pada Gambar 3.5. Contoh hasil masukkan dan keluarah *preprocessing* dapat dilihat pada Tabel 3.7.



Gambar 3.5 Diagram alir preprocessing

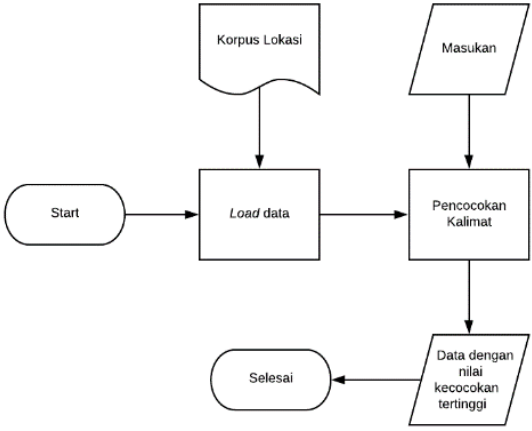
Tabel 3.7 Contoh masukkan dan keluaran Preprocessing

Contoh *Input*	**Contoh *Output* Preprocessing**
@indosatcare min, KENAPA jaringan internet saya HILANG MUNCUL HILANG MUNCUL	admin kenapa jaringan internet saya hilang mncul hilang muncul lagi ya saya

lg ya... saya sedang di daerah cipinang muara jati negara??https://t.co/CurKHGFgj0	sedang di daerah cipinang muara jadi negara

3.4.3. POS Tagging

Pembuatan model POS Tagging menggunakan dataset *online* yang dapat diunduh pada tautan berikut: https://github.com/famrashel/idn-tagged-corpus. Data tersebut disesuaikan terlebih dahulu sesuai dengan format POS *Tagging* pada spaCy dengan daftar *tag* yang dapat dilihat pada Tabel 3.8. Diagram alir pembuatan model POS *Tagging* dapat dilihat pada Gambar 3.7. Contoh rancangan masukan dan keluaran POS *Tagging* dapat dilihat pada Tabel 3.9. Hasil dari POS *Tagging* selanjutnya juga akan digunakan untuk pemrosesan *Rule-based Matching* yang akan dijelaskan pada subbab 3.3.2.



Gambar 3.6 Diagram alir proses pencocokan kalimat lokasi

Tabel 3.8 Daftar tag

Tag	Keterangan	Contoh
ADJ	*adjective* (kata sifat)	panjang, lemot, manis, lambat
ADP	*adposition* (kata depan)	di, ke, dalam, pada

ADV	*adverb* (kata keterangan)	sangat, hanya, segera
AUX	*auxiliary verb* (verba bantu)	boleh, harus, sudah
CONJ	*coordinating conjuction* (konjungsi koordinatif)	dan, tetapi, atau
DET	*determiner* (artikel)	para, sebuah
INTJ	*interjection* (interjeksi)	ayo, mari, wah
NOUN	*noun* (nomina/kata benda)	pulsa, paket, nomor
NUM	*numeral* (numeralia)	satu, kedua, banyak
PART	*particle* (partikel)	pun, -lah, -kah
PRON	*pronoun* (pronomina)	anda, kita
PROPN	*proper noun*	Surabaya, Jawa Timur, Indonesia
PUNCT	*punctuation* (tanda baca)	“…”, ?, .
SCONJ	*subordinating conjunction* (konjungtor subordinatif)	jika, supaya, maka
SYM	*symbol* (simbol)	+, %
VERB	*verb* (verba/kata kerja)	mengganggu, melambat
X	*other* (belum diketahui kategorinya secara pasti)	-

Tabel 3.9 Contoh rancangan Masukkan dan keluaran POSTAG

Contoh *Input*	**Contoh *Output* POSTAG**
@indosatcare kenapa jaringan internet saya hilang muncul hilang muncul lg ya... saya sedang di daerah cipinang muara jati negara.	[('@indosatcare', 'CD', 'NUM'), ('kenapa', 'IN', 'ADP'), ('jaringan', 'NN', 'NOUN'), ('internet', 'NN', 'NOUN'), ('saya', 'PRP', 'PRON'), ('hilang', 'NN', 'NOUN'), ('muncul', 'VB', 'VERB'), ('hilang', 'NN', 'NOUN'), ('muncul', 'VB', 'VERB'), ('lg', 'NN', 'NOUN'), ('ya', 'SC', 'SCONJ'), ('...', 'Z', 'PUNCT'), ('saya', 'PRP', 'PRON'), ('sedang', 'MD', 'AUX'), ('di', 'NNP', 'NOUN'), ('daerah', 'NN', 'NOUN'), ('cipinang', 'NN', 'NOUN'), ('muara', 'NN',

	'NOUN'), ('jati', 'NN', 'NOUN'), ('negara', 'NN', 'NOUN'), ('.', 'Z', 'PUNCT')]

3.4.4. Rule Based Matching

Rule-based Matching (RBM) adalah teknik pencocokan kata atau frasa menggunakan *rule* / aturan-aturan tertentu yang telah ditentukan sebelumnya. Proses ini nantinya digunakan untuk mendapatkan kata atau frasa penting berupa lokasi terjadinya keluhan yang mungkin belum terdekteksi oleh NER. Hasil dari POS *Tagging* akan digunakan sebagai acuan pemrosesan RBS. Daftar *rule* yang digunakan dapat dilihat pada Tabel 3.11. Contoh rancangan masukan dan keluaran RBS dapat dilihat pada Tabel 3.10.

Tabel 3.10 Contoh rancangan Input Output Rule Based Matching

Contoh *Input*	**Contoh *Output* RBM**
@indosatcare kenapa jaringan internet saya hilang muncul hilang muncul lg ya... saya sedang di daerah cipinang muara jati negara.	daerah cipingan mura jati negara

Tabel 3.11 Daftar Rule pada RBM

Rule	**Fungsi**
ADP + PROPN	Mendeteksi Lokasi pada Tweet
daerah + PROPN + PROPN	
desa + PROPN + PROPN	
ds. + PROPN + PROPN	
kota + PROPN + PROPN	
kecamatan + PROPN + PROPN	
kec. + PROPN + PROPN	
kabupaten + PROPN + PROPN	
kab. + PROPN + PROPN	

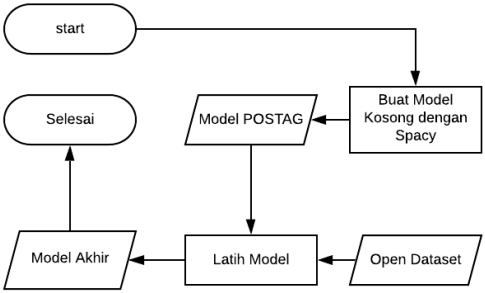
3.4.5. Named Entity Recognition (NER)

NER digunakan untuk mendeteksi detail gangguan seperti keterangan, lokasi, waktu, dan informasi tambahan terkait keluhan

jaringan dan produk. Label yang digunakan dalam pembuatan model NER dapat dilihat pada Tabel 3.12.

Tabel 3.12 Daftar label NER

Nama Label	Keterangan
Lokasi	Lokasi yang dikeluhkan pengguna
Waktu	Waktu kejadian
Produk	Kata yang berkaitan dengan produk seperti: paket data, pulsa, dan promo
Service	Kata yang berkaitan dengan masalah jaringan seperti: sinyal, 3g, dan 4g.
Keterangan	Keterangan penting lainnya



Gambar 3.7 Diagram alir pembuatan POSTAG

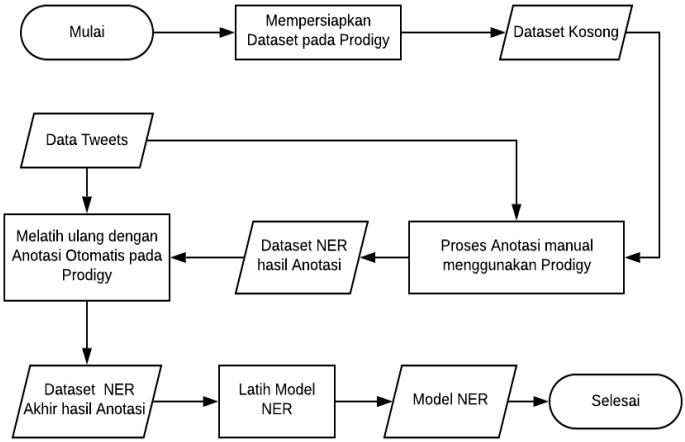
Pembuatan model NER menggunakan dataset dari kumpulan tweet yang dianotasi secara manual menggunakan Prodigy. Dalam prosesnya, *blank* model digunakan sebagai tahap awal pembuatan karena spaCy belum mendukung model berbahasa Indonesia. Proses anotasi dilakukan dalam 2 tahap, yaitu proses anotasi secara manual berdasarkan entitas-entitas yang telah didefinisikan dalam Tabel 3.12. Selanjutya dataset tersebut diverifikasi menggunakan data tweet lama dan baru. Hasil anotasi kemudian diekspor untuk dilatih menggunakan spaCy. Diagram alir pembuatan model NER dapat dilihat pada Gambar 3.8. Contoh rancangan masukan dan keluaran NER dapat dilihat pada Tabel 3.13.

Tabel 3.13 Contoh rancangan Input Output NER

Contoh *Input*	Contoh *Output*
@indosatcare kenapa jaringan internet saya hilang muncul hilang muncul lg ya... saya sedang di daerah cipinang muara jati negara.	[(‘jaringan internet, SERVICE), (‘hilang muncul hilang muncul’, KETERANGAN), (‘daerah cipinang muara jati negara’, LOKASI)]

3.4.6. Model Ekstraksi Fitur

DOC2VEC merupakan salah satu *word embedding* yang digunakan sebagai metode ekstraksi fitur pada teks sehingga dapat dikenali dan diolah oleh mesin. Hasil dari ekstraksi fitur



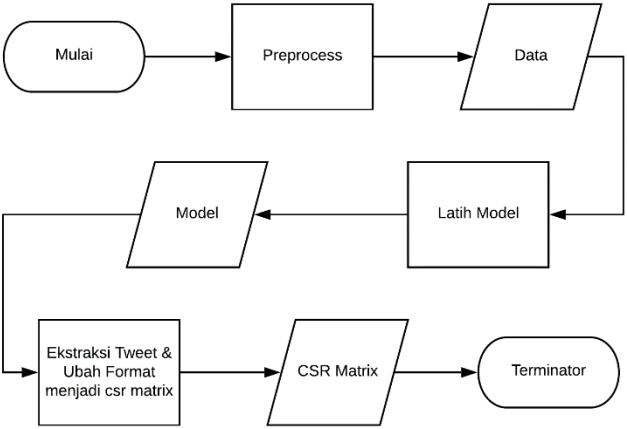
Gambar 3.8 Diagram alir pembuatan NER

menggunakan Doc2Vec akan digunakan sebagai data latih dan masukan model Support Vector Machine untuk diklasifikasikan. Proses pembuatan model Doc2Vec dapat dilihat pada Gambar 3.9.

3.4.7. Klasifikasi menggunakan SVM

SVM merupakan teknik klasifikasi *supervised* yang akan digunakan untuk mengkategorikan tweet yang ditujukan kepada *customer service*. Hasil dari klasifikasi tweet menggunakan SVM akan dijadikan masukkan untuk mendeteksi gangguan pada suatu

lokasi dan mengkategorikan tweet untuk diproses lebih lanjut oleh *customer service*. Tweet akan diklasifikasikan menjadi 2 kelas jaringan dan produk. Contoh tweet yang merupakan kelas jaringan biasanya memiliki kata kunci seperti: lemot, sinyal, 3g, 4g, dan internet tidak aktif. Sedangkan pada kelas produk biasanya memiliki kata kunci seperti: paket data, kuota, dan pulsa.

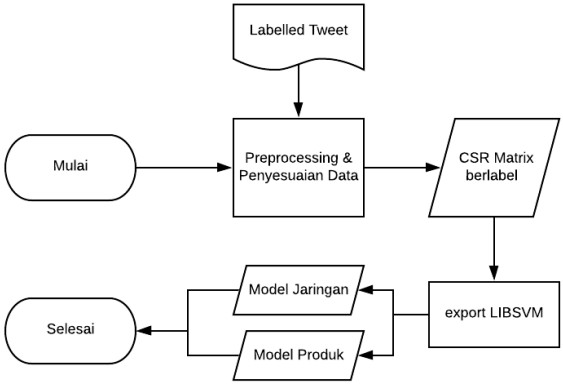


Gambar 3.9 Diagram alir pembuatan DOC2VEC

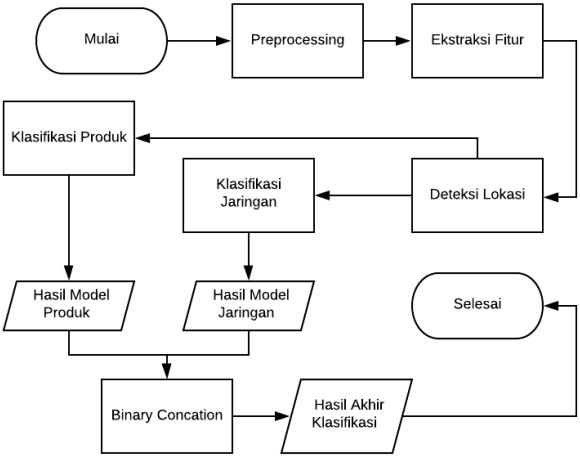
Berdasarkan dokumentasi SVM pada Apache Flink, model yang akan dibuat menggunakan skema *one-againts-all*, yang berarti hanya akan mendeteksi apakah suatu data merupakan kelas yang ditentukan (+) atau bukan (-). Sehingga untuk mendeteksi 2 kelas, jaringan atau produk, diperlukan 2 model. Hasil kedua model tersebut kemudian digabungkan untuk mendapatkan hasil akhir klasifikasi. Penggabungan dilakukan berdasarkan Tabel 3.14. Diagram alir pembuatan model SVM dapat dilihat pada Gambar 3.10. Contoh rancangan masukan dan keluaran SVM dapat dilihat pada Tabel 3.15. Sedangkan diagram alir tahapan klasifikasi dapat dilihat pada Gambar 3.11.

Tabel 3.14 Penggabungan hasil klasifikasi dengan Binary Concation

		Model Jaringan	
		+	-
Model Produk	+	Keduanya	Produk
	-	Jaringan	Lain Lain



Gambar 3.10 Diagram alir pembuatan model SVM



Gambar 3.11 Diagram alir tahapan klasifikasi

Tabel 3.15 Contoh rancangan Masukkan dan keluaran SVM

Contoh *Input*	**Contoh *Output SVM***
@indosatcare kenapa jaringan internet saya hilang muncul hilang muncul lg ya... saya sedang di daerah cipinang muara jati negara.	Jaringan
@indosatcare min sinyal internet aku kok ga bisa ya, dari tadi data seluler udah diaktifin muncul logo 4g tapi muncul lalu hilang.	Jaringan
@smartfrencare ni kenapa bebrapa hari ini kalo siang lemot terus sih? bikin gedeg aja	Jaringan
@indosatcare ada yg sama sama oe ga nggk bisa maketin data? pulsa aja malah kepotong 10rb	Produk
@telkomsel poin dari pembayaran terakhir tagihan kartu halo gak bertambah, harusnya ada tambahan setiap selesai pembayaran kan?	Produk
@telkomsel min kenapa kok saya gak bisa beli paket data 50GB yang lagi promosi ya?	Produk

3.4.8. Deteksi Lokasi

Deteksi lokasi menggunakan ekstraksi teks dipilih karena koordinat GPS yang terdapat pada objek *tweet* bisa saja tidak merepresentasikan lokasi terjadi gangguan sebenarnarnya [31].

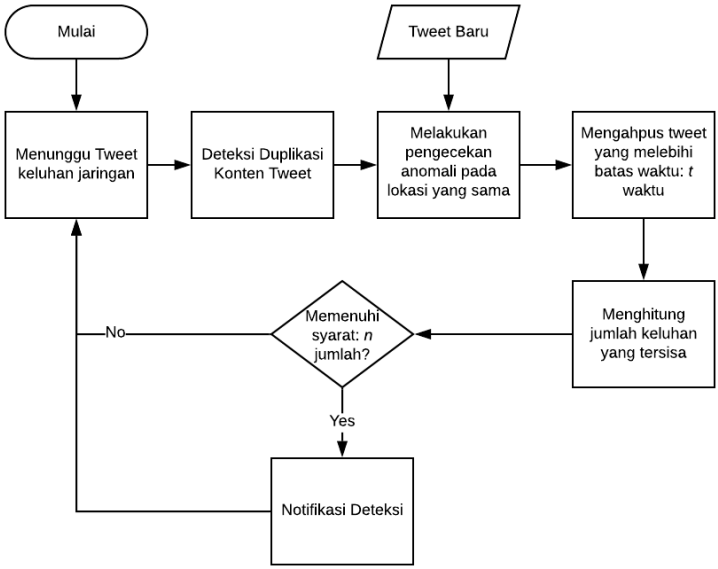
Ketika mengamati data yang diambil, kebanyakan pengguna mengutarakan keluhan jaringan dengan mencantumkan lokasi pada *tweet*-nya. Oleh karena itu, penulis menggunakan NER, POSTAG, dan *Rule-based Matching* untuk mengekstrak lokasi dari teks tweet. Ketiga hasil masing-masing metode tersebut dinormalkan berdasarkan korpus lokasi menggunakan *Lavenshtein Distance*. Selanjutnya, keluaran metode-metode tersebut divoting untuk mendapatkan hasil akhir lokasi.

3.4.9. Deteksi Gangguan

Gangguan jaringan adalah suatu hal yang menyebabkan distorsi pada sarana telekomunikasi sehingga menghambat proses komunikasi antar sistem. Gangguan jaringan biasanya dirasakan oleh beberapa masyarakat dalam area tertentu. Selain masalah dari pihak provider, gangguan juga dapat terjadi pada seseorang karena anomali atau kondisi khusus seperti sedang berada pada ruangan tertutup. Sehingga diperlukan metode untuk memverifikasi kejadian tersebut. Menghitung jumlah pesan atau *posting* dalam kurun waktu tertentu merupakah salah satu contoh untuk mendeteksi dan memverifikasi ada atau tidaknya suatu kejadian. Cara ini merupakan salah satu cara yang kurang efektif dan relatif terhadap tiap-tiap studi kasus tertentu [32]. Namun, metode ini merupakan salah satu metode yang mudah untuk diimplementasikan serta waktu komputasi yang cukup singkat karena tidak memerlukan pelatihan model dan komputasi yang kompleks. Sehingga cocok diimplementasikan pada pemrosesan yang lebih mengutamakan waktu komputasi yang cepat. Oleh karena itu, metode tersebut dipilih untuk mendeteksi ada atau tidaknya gangguan jaringan dengan memberikan pengguna akhir untuk mendefinisikan jumlah n keluhan dan kurun waktu t.

Pada tugas akhir ini gangguan jaringan didefinisikan sebagai n keluhan jaringan pada kurun waktu t. Contoh kondisi yang menandakan gangguan menggunakan metode ini digambarkan pada Gambar 3.13. Cara ini dapat diimplementasikan dengan mengelompokkan data tweet yang memiliki kelas keluhan jaringan kedalam satu grup yang sama berdasarkan lokasinya.

Untuk menghemat sumber daya waktu dan komputasi pada pemrosesan *streaming*, pengecekan pemenuhan kriteria hanya akan dicek ketika terdapat tweet baru masuk sehingga tidak perlu melakukan pengecekan setiap detiknya. Ketika terdapat sebuah tweet baru, data masing-masing tweet pada grup tersebut akan dicek apakah memiliki jarak waktu *posting* lebih dari *t* satuan waktu dari waktu *posting* tweet terbaru. Apabila telah melebihi, maka tweet tersebut akan dikeluarkan dari grup dan dinyatakan sebagai bukan gangguan jaringan atau anomali. Setelah grup bersih dari anomali gangguan, lalu jumlah keluhan yang tersisa dihitung untuk mengetahui apakah telah memenuhi *n* kejadian yang nantinya dapat disimpulkan ada atau tidaknya gangguan jaringan. Karena sistem merupakan premrosesan berbasis *streaming*, sehingga tahapan tersebut diulang kembali terus menerus. Diagram alir metode deteksi gangguan dapat dilihat pada Gambar 3.12.



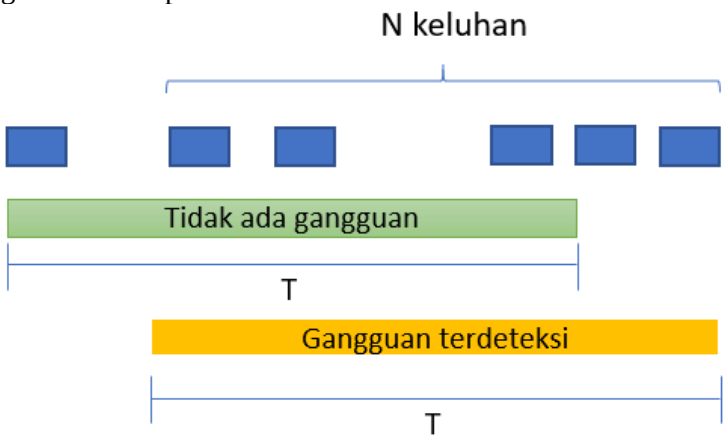
Gambar 3.12 Diagram alir deteksi gangguan

3.5. Perancangan Antarmuka Sistem

Antarmuka sistem pada tugas akhir ni digunakan untuk menampilkan demo aplikasi yang dapat menampilkan *datastream* tweet hasil akhir pemrosesan deteksi gangguan dan keluhan pengguna. Selain itu, antarmuka sistem juga dapat memberikan umpan balik berupa status tweet dari pengguna. Antarmuka dirancang menggunakan Flask dan VueJs.

3.6. Evaluasi dan Uji Coba

Pada tahap uji coba dilakukan evaluasi untuk model NER, model LIBSVM, dan kecepatan pemrosesan. Untuk menghitung nilai *precision*, *recall*, dan *f-measure*, model NER dievaluasi menggunakan bantuan pustaka spaCy sedangkan model LIBSVM dengan metode *Cross Validation* pada Apache FLink. Kecepatan premosesan terhitung sesaat setelah tweet diterima hingga akan disimpan dalam database untuk dihitung rata-rata kecepatan dan kemampuan Apache Flink menggunakan uji coba *stress-test*, yaitu mengirimkan data tweet dalam jumlah besar dalam waktu yang singkat pada perangkat dan lingkungan implementasi yang digunakan oleh penulis.



Gambar 3.13 Konsep deteksi gangguan

BAB IV
IMPLEMENTASI SISTEM

Bab ini akan menjelaskan tentang implementasi tugas akhir berdasarkan rancangan perangkat lunak. Proses implementasi mengacu pada rancangan perangkat yang telah dilakukan sebelumnya, namun juga dimungkinkan terjadinya perubahan-perubahan jika dirasa perlu. Implementasi dilakukan dalam bahasa Scala, Python, dan Javascript.

4.1. Lingkungan Implementasi

Pada tugas akhir ini digunakan beberapa perangkat serta pustaka yang digunakan untuk mempermudah pengerjaan tugas akhir. Perang keras dan sistem operasi yang digunakan selama tugas akhir adalah:

- Processor Intel Core i7-4720HQ 2.60GHz
- RAM 12 GB (11.9GB)
- Disk 256GB SSD + 1TB SATA
- Sistem Operasi Windows 10 Education
- Jenis Sistem 64-bit operating system, x64-based
- Apache Flink Versi *standalone* 1.7.2

Sedangkan untuk alat pembantu yang digunakan selama tugas akhir diuraikan pada Tabel 4.1.

Tabel 4.1 Daftar alat pembantu dalam pengerjaan tugas akhir

No	Alat Penunjang	Keterangan
1	Python	Bahasa Python digunakan untuk menangani task Natural Language Processing (NLP).
2	Scala	Bahasa Scala digunakan untuk menangani pengolahan klasifikasi data pada Apache Flink.
3	Apache Flink	Kerangka kerja pemrosesan data berbasis *streaming*.

4	Apache Kafka	Platform *stream-processing* untuk menangani pengiriman data *stream.*
5	MongoDB	Database untuk menyimpan data berformat JSON.
6	MongoDB Compass Community	Alat untuk mengakses database MongoDB dalam proses validasi dan *debugging*.
7	Tweepy	Pustaka python untuk mengakses API Twitter seperti *scrapping.*
8	pykafka	Pustaka python untuk mengakses Apache Kafka.
9	pymongo	Pustaka python untuk mengakses MongoDB.
10	Sastrawi	Pustaka python untuk melakukan *stemming*
11	sklearn	Pustaka *machine learning* pada python yang nantinya digunakan untuk mengekspor model SVMLIB
12	scipy	Pustaka optimasi perhitungan matematis pada Python yang nantinya digunakan untuk mengubah bentuk *list* menjadi *csr matrix*
13	Twitter Text Python	Pustaka python untuk mengekstrak entitas entias pada *tweet*
14	spaCy	Pustaka untuk melakukan NER, POS Tagging, dan Rule-based Matching.
15	Gensim	Pustaka python untuk melakukan ekstraksi fitur pada teks.
16	fuzzywuzzy	Pustaka python untuk mencari *string* atau kata dengan pola atau korpus tertentu.
17	Flask	Kerangka kerja aplikasi web Python.
18	Visual Studio Code, IntelliJ	Aplikasi yang digunakan untuk penulisan program.

19	Ms. Word dan Ms. Excel	Aplikasi untuk mengolah data.

4.2. Persiapan dan Pengambilan Data

Data tweet keluhan dari Twitter yang ditujukan kepada *Customer Service* Indosat (@IndosatCare), Telkomsel (@Telkomsel), SmartFren (@smartfrencare) diambil dengan tweepy secara otomatis.

4.3. Implementasi Proses

Implementasi proses dilakukan beradasarkan perancangan proses yang dijelaskan pada bab Analisis dan Perancangan Sistem.

4.3.1. Implementasi *String Matching* Gazetter

Implementasi *String Matching* Gazetter menggunakan pustaka online FuzzyWuzzy. Data lokasi berupa nama provinsi, kota/kecamatan, kelurahan, dan desa di Indonesia yang diambil dari repositori online milik Cahya DSN [33]. Implementasi kelas Gazetter dapat dilihat pada Kode Sumber 4.1. Pada kode sumber tersebut, terdapat beberapa fungsi yang masing-masing keguanaannya dijelaskan pada Tabel 4.2.

Tabel 4.2 Daftar fungsi kelas Gazetter

Nama Fungsi	Parameter	Kegunaan
__init__	places	Menyimpan data tempat dari parameter *places* ketika instansiasi kelas
match_gazetter	target	Mencari hasil kalimat lokasi dan nilai yang paling cocok dengan parameter target pada korpus data

match_gazetter_ score	target	Mencari nilai jarak yang paling cocok dengan parameter target pada korpus data
find_place	target	Mencari kalimat lokasi dan nilai yang paling cocok dengan parameter target dengan menambahkan satu persatu kata lokasi secara bertahap
parse_place	text	Mengganti singkatan pada lokasi sehingga bentuknya normal dan sesuai dengan korpus data
get_location	words, tags	Mencari teks lokasi dan nilai yang paling cocok sesuai dengan hasil POSTAG

```
1.  from fuzzywuzzy import fuzz
2.  from fuzzywuzzy import process
3.
4.
5.  class Gazetter:
6.    def __init__(self, places):
7.      self.places = places
8.      print("There are %d places used."%(len(places))
    )
9.    def match_gazetter(self,target):
10.     if(len(target)==0):
11.       return ("",-1)
12.     place, score = process.extractOne(target, self.
    places, scorer=fuzz.token_sort_ratio)
13.     return (place,score)
```

```
14.
15.   def match_gazetter_score(self,target):
16.     if(len(target)==0):
17.       return ("",-1)
18.     place, score = process.extractOne(target, self.
    places, scorer=fuzz.token_sort_ratio)
19.     return score
20.
21.   def find_place(self,target):
22.     split = target.split(",")
23.     result = ("",0)
24.     if(len(split)==1):
25.       split = target.split(" ")
26.       for tobefind in split:
27.         curr_val = self.match_gazetter(tobefind)
28.         if(curr_val[1]> result[1]):
29.           result = curr_val
30.       return result
31.
32.     else:
33.       temp = ""
34.       for tobefind in split:
35.         temp = temp + " " + self.parsePlace(tobefin
    d)
36.         curr_val = self.match_gazetter(temp)
37.         if(curr_val[1]> result[1]):
38.           result = curr_val
39.       return result
40.
41.   def parsePlace(self,text):
42.     if(text=="kec." or text =="kec"):
43.       return "KECAMATAN"
44.     elif(text=="kab." or text == "kab"):
45.       return "KABUPATEN"
46.     elif(text == "ds." or text == "ds"):
47.       return "DESA"
48.     return text
49.
50.   def get_location(self,words,tags):
51.     idx_s = 0
52.     idx_e = 0
53.
54.     place = ""
```

```
55.     cur_score = 0
56.
57.     for i in range(len(tags)):
58.       if(i != 0 and tags[i-1] == "IN" ):
59.         if(tags[i] == 'NNP' or tags[i] == 'NN'):
60.           if(i == 0):
61.             idx_s = 0
62.             idx_e = 0
63.           elif(tags[i-1] == 'NNP'):
64.             idx_e = i
65.           else:
66.             idx_s = i
67.             idx_e = i
68.         elif(i != 0 and (tags[i] == 'NNP' or tags[i
    ] == 'NN')):
69.           if(idx_e - idx_s + 1 > 1):
70.             query = ' '.join(words[idx_s:idx_e+1])

71.             query = self.parsePlace(query)
72.             score = self.match_gazetter_score(query
    )
73.             if score > cur_score:
74.               cur_score = score
75.               place = query
76.
77.
78.         if((i == len(tags)-
    1) and (tags[i] == 'NNP' or tags[i] == 'NN')):
79.           if(idx_e - idx_s + 1 > 1):
80.             query = ' '.join(words[idx_s:idx_e+1])

81.             query = self.parsePlace(query)
82.             score = self.match_gazetter_score(query
    )
83.             if score > cur_score:
84.               cur_score = score
85.               place = query
86.
87.       return place
```

Kode Sumber 4.1 Implementasi kelas Gazetter

4.3.2. Implementasi Pembuatan Model POSTAG

Implementasi pembuatan model POSTAG dilakukan menggunakan pustaka spaCy. Data latih diambil dari Indonesian Manually Tagged Corpus [30]. Cara implementasi POS *Tagging* dapat dilihat pada Kode Sumber 4.2. Pada kode sumber tersebut, format tag pada data dikonversi agar sesuai dengan format tag pada spaCy. Selanjutnya *load* blank model dan *load* data latih. Lalu setiap *tag* pada *part-of-speech* akan dilatih. Hasil latih berupa model POS *Tagging* yang akan disimpan kedalam *disk*. Di akhir kode sumber, model POS *Tagging* diuji coba dengan masukan teks baru untuk mengetahui model yang dibuat apakah sudah dapat mengenali setiap kelas kata dalam kalimat.

```
1.  …
2.  TAG_MAP = { "CC": {"pos": "CONJ"}, "CD": {"pos": "N
    UM"}, "DT": {"pos": "DET"},"FW": {"pos": "X"}, "IN"
    : {"pos": "ADP"},"JJ": {"pos": "ADJ"}, "MD": {"pos"
    : "AUX"}, "NEG": {"pos": "X"}, "NN": {"pos": "NOUN"
    }, "NND": {"pos": "NOUN"}, "NNP": {"pos": "PROPN"},
     "OD": {"pos": "NUM"}, "PR": {"pos": "PRON"}, "PRP"
    : {"pos": "PRON"}, "RB": {"pos": "ADV"}, "RP": {"po
    s": "PART"}, "SC": {"pos": "SCONJ"},"SYM": {"pos":
    "SYM"},"UH": {"pos": "INTJ"}, "VB": {"pos": "VERB"}
    , "WH": {"pos": "X"}, "XX": {"pos": "X"}, "Z": {"po
    s": "PUNCT"} }
3.
4.  TRAIN_DATA = []
5.  cv = open("indonesian-manually-tagged.pkl","rb")
6.  # data saved in pickle format to keep it ease
7.  TRAIN_DATA = pickle.load(cv)
8.  print("Let's train %d strings!"%(len(TRAIN_DATA)))

9.  @plac.annotations(
10.   lang=("ISO Code of language to use", "option", "l
    ", str),
11.   output_dir=("Optional output directory", "option"
    , "o", Path),
12.   n_iter=("Number of training iterations", "option"
    , "n", int),
```

```
13. )
14. def main(lang="id", output_dir=None, n_iter=25):
15.   nlp = spacy.blank(lang)
16.   # add the tagger to the pipeline
17.   tagger = nlp.create_pipe("tagger")
18.   for tag, values in TAG_MAP.items():
19.       tagger.add_label(tag, values)
20.   nlp.add_pipe(tagger)
21.
22.   optimizer = nlp.begin_training()
23.   for i in range(n_iter):
24.     random.shuffle(TRAIN_DATA)
25.     losses = {}
26.
27.     batches = minibatch(TRAIN_DATA, size=compoundin
    g(4.0, 32.0, 1.001))
28.     for batch in batches:
29.       texts, annotations = zip(*batch)
30.       nlp.update(texts, annotations, sgd=optimizer,
     losses=losses)
31.     print("Losses", losses)
32.
33.   test_text = "Min @IndosatCare, di deket gandaria
    jakarta surabaya kok ngelag ya?"
34.   doc = nlp(test_text)
35.   print("Tags", [(t.text, t.tag_, t.pos_) for t in
    doc])
36.
37.   nlp.to_disk("./pos-tag")
38.   print("Saved model to", output_dir)
39.   print("Loading from", output_dir)
40.   nlp2 = spacy.load("./pos-tag")
41.   doc = nlp2(test_text)
42.   print("Tags", [(t.text, t.tag_, t.pos_) for t in
    doc])
43.
44. if __name__ == "__main__":
45.   plac.call(main)
46. …
```

Kode Sumber 4.2 Pembuatan Model POSTAG

4.3.3. Implementasi Rule Based Matching

Implementasi *Rule Based Matching* menggunakan model POS *Tagging* (sub bab 4.3.1) yang dapat dilihat pada Kode Sumber 4.4. Pada kode sumber tersebut, model *Matcher* dibuat dengan menambahkan *rule* yang telah didefinisikan pada subbab 3.4.1. Kemudian *tweet* yang akan diproses diolah menggunakan POS *Tagging* untuk mendapatkan entitas-entitas penting. Selanjutnya hasil pengolahan POS *Tagging* dicocokkan dengan model *Matcher* yang telah dibuat. Lalu, hasil yang sesuai akan diproses kembali menggunakan *string matcher* menggunakan Gazetter karena terdapat kemungkinan bahwa bentuk kata lokasi yang telah diekstrak dari *tweet* belum normal seperti berupa singkatan, terdapat huruf yang hilang, salah tulis, dan lain sebagainya.

```
1.  …
2.  def loadMatcher(self):
3.      matcher = Matcher(self.__mPostag.vocab)
4.      pattern = [{"POS": "ADP", "OP": "+"},{"LOWER":
    "daerah", "OP": "?"},{"LOWER": "desa", "OP": "?"},{
    "LOWER": "kota", "OP": "?"},{"LOWER": "kecamatan",
    "OP": "?"},{"LOWER": "kabupaten", "OP": "?"}, {"POS
    ": "NOUN", "OP": "+"}]
5.      pattern1 = [{"LOWER": "daerah", "OP": "+"}, {"P
    OS": "NOUN", "OP": "+"}]
6.      pattern2 = [{"LOWER": "desa", "OP": "+"}, {"POS
    ": "NOUN", "OP": "+"}]
7.      pattern3 = [{"LOWER": "kota", "OP": "+"}, {"POS
    ": "NOUN", "OP": "+"}]
8.      pattern4 = [{"LOWER": "ds", "OP": "+"},{"POS":"
    PUNCT","OP":"?"},{"POS": "NOUN", "OP": "+"}]
9.      pattern5 = [{"LOWER": "kecamatan", "OP": "+"},
    {"POS": "NOUN", "OP": "+"}]
10.     pattern6 = [{"LOWER": "kec", "OP": "+"},{"POS":
    "PUNCT","OP":"?"}, {"POS": "NOUN", "OP": "+"}]
11.     pattern7 = [{"LOWER": "kabupaten", "OP": "+"},
    {"POS": "NOUN", "OP": "+"}]
12.     pattern8 = [{"LOWER": "kab", "OP": "+"},{"POS":
    "PUNCT","OP":"?"}, {"POS": "NOUN", "OP": "+"}]
```

```
13.     matcher.add("LOKASI", None,pattern,pattern2,pat
    tern3,pattern4,pattern5,pattern6,pattern7,pattern8)
14.     return matcher
15.   def getLocationBasedOnMatcher(self,tweet):
16.     locations = {
17.       "desa" : [],
18.       "kecamatan" : [],
19.       "kabupaten" : [],
20.       "kota" : [],
21.       "others" : [],
22.     }
23.     array_of_text = self.getTokenMatchEntities(twee
    t)
24.     if(len(array_of_text)==0):
25.       return ""
26.     for ent in array_of_text:
27.       holder = ent
28.       if(holder.lower().find("desa") >= 0 or holder
    .lower().find("ds.") >= 0):
29.         locations['desa'].append(holder)
30.       elif(holder.lower().find("kecamatan") >= 0 or
     holder.lower().find("kec.") >= 0):
31.         locations['kecamatan'].append(holder)
32.
33.       elif(holder.lower().find("kabupaten") >= 0 or
     holder.lower().find("kab.") >= 0):
34.         locations['kabupaten'].append(holder)
35.       elif(holder.lower().find("kota") >= 0):
36.         locations['kota'].append(holder)
37.       else:
38.         locations['others'].append(holder)
39.
40.     loc_result = ("",0)
41.     for loc_input in locations['desa']:
42.       curr_val = self.__gazetter.find_place(loc_inp
    ut)
43.       if(curr_val[1] > loc_result[1]):
44.         loc_result = curr_val
45.
46.     if(loc_result[0] != ""):
47.       return loc_result[0]
48.
49.     for loc_input in locations['kecamatan']:
```

```
50.         curr_val = self.__gazetter.find_place(loc_inp
    ut)
51.         if(curr_val[1] > loc_result[1]):
52.           loc_result = curr_val
53.
54.       if(loc_result[0] != ""):
55.         return loc_result[0]
56.
57.       for loc_input in locations['kabupaten']:
58.         curr_val = self.__gazetter.find_place(loc_inp
    ut)
59.         if(curr_val[1] > loc_result[1]):
60.           loc_result = curr_val
61.
62.       if(loc_result[0] != ""):
63.         return loc_result[0]
64.
65.       for loc_input in locations['kota']:
66.         curr_val = self.__gazetter.find_place(loc_inp
    ut)
67.         if(curr_val[1] > loc_result[1]):
68.           loc_result = curr_val
69.
70.       if(loc_result[0] != ""):
71.         return loc_result[0]
72.
73.       for loc_input in locations['others']:
74.         curr_val = self.__gazetter.find_place(loc_inp
    ut)
75.         if(curr_val[1] > loc_result[1]):
76.           loc_result = curr_val
77.
78.       return loc_result[0]
79. …
```

Kode Sumber 4.3 Pembuatan model rule based matching

4.3.4. Implementasi Pembuatan Model NER

Implementasi pembuatan model NER menggunakan bantuan pustaka Prodigy dan spaCy.

4.3.4.1. Mempersiapkan *dataset* pada Prodigy

Untuk melakukan anotasi manual pada Prodigy, diperlukan *dataset* sehingga pengguna dapat mengelompokkan hasil anotasinya. *Dataset* dibuat dengan perintah pada Kode Sumber 4.4. Pada kode sumber tersebut, *dataset* baru diberi nama 'cs_ner'.

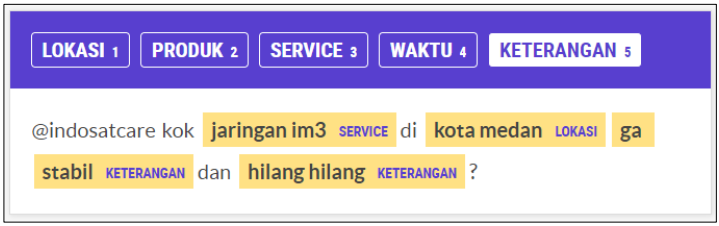
```
python -m prodigy dataset cs_ner
```

Kode Sumber 4.4 Pembuatan dataset pada Prodigy

4.3.4.2. Proses Anotasi manual pada Prodigy

Anotasi NER secara manual dilakukan untuk memberikan label secara manual pada entitas kata yang dianggap penting. Cara menganotasi NER manual pada Prodigy dengan menggunakan perintah *ner.manual* seperti yang telah ditunjukkan pada Kode Sumber 4.5. Pada kode sumber tersebut, *file* data yang akan dianotasi memiliki nama "prodigy_input.txt" dan label yang digunakan ada 5 macam, yaitu 'LOKASI", "PRODUK", "SERVICE", "WAKTU", dan "KETERANGAN". Penjelasan masing-masing label dapat dilihat pada subbab 3.4.5. Contoh bentuk data yang akan dianotasikan ditunjukkan oleh Gambar 4.3.

Setelah menjalankan perintah pada Kode Sumber 4.5, *web server* Prodigy akan menyala dan penulis dapat menganotasikan NER secara manual pada data yang telah disiapkan melalui *browser*. Tampilan *Prodigy* sebelum dilakukan anotasi (tampilan awal) dapat dilihat pada Gambar 4.2 dan

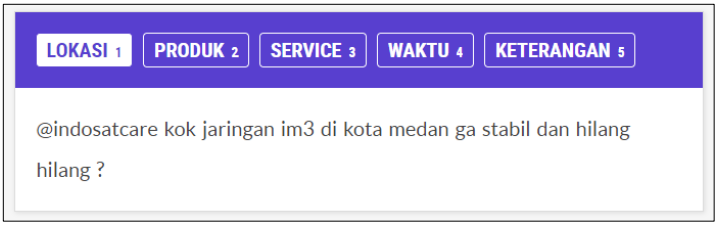


Gambar 4.1 Contoh tampilan Prodigy setelah anotasi

tampilan Prodigy setelah melakukan anotasi dapat dilihat pada Gambar 4.1 Penulis melakukan anotasi terhadap 1572 *tweet.*

```
python -m prodigy ner.manual cs_ner “D:\Dokumen\Tugas
Akhir\models" prodigy_input.txt --label "LOKASI, PRODUK,
SERVICE, WAKTU, KETERANGAN"
```

Kode Sumber 4.5 Proses anotasi manual NER dengan Prodigy



Gambar 4.2 Contoh tampilan Prodigy sebelum anotasi

```
1   @indosatcare kok jaringan im3 di kota medan ga stabil dan hilang
    hilang?
2   @smartfrencare @disaa_xx apa gua juga harus ganti ya,tapi apakah
    smartfren udah optimal di daerah saya ya
3   @indosatcare min seharian indosat di daerah situbondo sinyalnya kok
    ilang2an ya, bener2 nggak bisa make lhoo.. kan rugi ya udah daftar
    paketan tapi nggak bisa kepake.. gimana dong..
```

Gambar 4.3 Contoh data masukan anotasi NER

4.3.4.3. Pembuatan Model *Pre-Trained* NER

Pembuatan model dilakukan dengan menggunakan kode program yang diambil dari GitHub ManivannanMurugavel [34]. Model *pre-trained* dilatih mengunakan hasil anotasi manual sebelumnya yang telah diekspor dengan Kode Sumber 4.6. Model ini nantinya akan dijadikan model awal untuk memperkaya *dataset* dan model akhir. Cara membuat model *pre-trained* NER dapat dilihat pada Kode Sumber 4.7. Awalnya muat *dataset* anotasi manual dan *blank* model. Kemudian tambahkan label sesuai entitas NER. Setiap entitas NER selanjutnya akan dilatih dengan iterasi sebanyak jumlah data. Hasil latih berupa model NER akan disimpan kedalam *disk*. Di akhir kode, model NER diuji coba dengan memasukkan data teks baru berupa *tweet* untuk mengetahui

apakah model yang dibuat sudah dapat mengenali entitas lokasi, *service*, produk, waktu, dan keterangan.

```
python -m prodigy ner.gold-to-spacy  cs_ner
".\my_ner_fix.json"
```

Kode Sumber 4.6 Mengeskpor hasil anotasi NER

```
1.  import spacy
2.  import random
3.  import json
4.
5.  TRAIN_DATA = []
6.  with open('my_ner_fix.json') as file:
7.    datas = file.readlines()
8.    for data in datas:
9.      TRAIN_DATA.append(json.loads(data))
10.
11.
12. def train_spacy(data,iterations):
13.   TRAIN_DATA = data
14.   nlp = spacy.blank('id')
15.   if 'ner' not in nlp.pipe_names:
16.     ner = nlp.create_pipe('ner')
17.     nlp.add_pipe(ner, last=True)
18.
19.   for _, annotations in TRAIN_DATA:
20.     for ent in annotations.get('entities'):
21.       ner.add_label(ent[2])
22.
23.   other_pipes = [pipe for pipe in nlp.pipe_names if
     pipe != 'ner']
24.   with nlp.disable_pipes(*other_pipes):  # only tra
    in NER
25.     optimizer = nlp.begin_training()
26.     for itn in range(iterations):
27.       print("Statring iteration " + str(itn))
28.       random.shuffle(TRAIN_DATA)
29.       losses = {}
30.       for text, annotations in TRAIN_DATA:
31.         nlp.update(
32.           [text],
33.           [annotations],
```

```
34.             drop=0.2,
35.             sgd=optimizer,
36.             losses=losses)
37.         print(losses)
38.   return nlp
39.
40.
41. prdnlp = train_spacy(TRAIN_DATA, 500)
42.
43. # Save our trained Model
44. modelfile = "model-for-location"
45. prdnlp.to_disk(modelfile)
46.
47. #Test your text
48. test_text = input("Masukkan test text: ")
49. doc = prdnlp(test_text)
50. for ent in doc.ents:
51.   print(ent.text, ent.start_char, ent.end_char, ent
    .label_)
```

Kode Sumber 4.7 Pembuatan model pre-trained NER

4.3.4.4. Proses Anotasi otomatis pada Prodigy

Anotasi NER secara otomatis dilakukan untuk memperkaya *dataset* anotasi manual dengan bantuan model *pre-trained*. Cara menganotasi NER otomatis pada Prodigy menggunakan perintah *ner.make-gold* pada Kode Sumber 4.8 dan *ner.teach* pada Kode Sumber 4.9. Pada kode sumber tersebut, *file* data yang akan dianotasi memiliki nama "prodigy_input_advanced.txt". Model yang akan digunakan berada pada direktori ".\model-for-location" yang telah dibuat sebelumnya. Sedangkan bentuk data dan label yang digunakan sama seperti proses anotasi manual yang telah dijelaskan pada subbab 4.3.4.2.

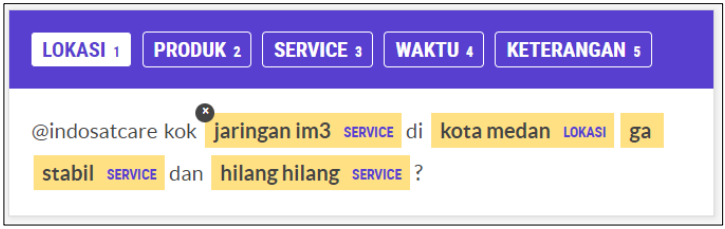
```
python -m prodigy ner.make-gold cs_ner ".\model-for-
location" prodigy_input_advanced.txt --label "LOKASI,
JARINGAN, SERVICE, WAKTU, KETERANGAN"
```

Kode Sumber 4.8 Proses anotasi NER dengan ner.make-gold

```
python -m prodigy ner.teach cs_ner ".\model-for-
location" prodigy_input_advanced.txt
```

Kode Sumber 4.9 Proses anotasi NER dengan ner.teach

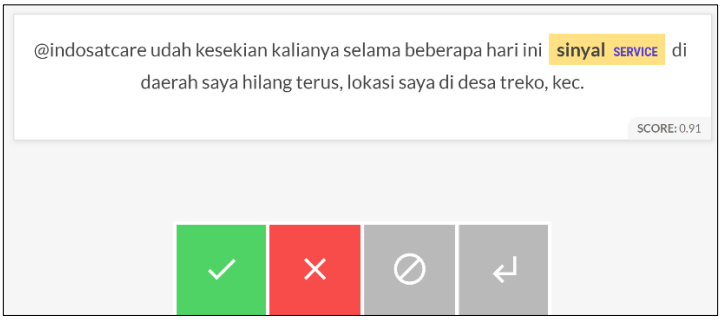
Setelah menjalankan perintah salah satu perintah kode sumber tersebut, *web server* Prodigy akan menyala dan pengguna dapat menganotasikan NER pada data yang telah disiapkan melalui *browser* dengan memvalidasi hasil anotasi otomatis oleh Prodigy. Pada proses *ner.make-gold* pengguna dapat mengoreksi dengan menambahkan atau menghapus saran entitas label anotasi pada teks *tweet* seperti yang terlihat pada Gambar 4.4. Sedangkan pada proses *ner.teach* pengguna dapat memvalidasi hasil anotasi otomatis dengan menekan tombol "✓" yang berati benar, "×" yang berarti salah, "⊘" diabaikan (tidak ditambahkan pada *dataset*), atau "↵" seperti yang terlihat pada Gambar 4.5.



Gambar 4.4 Contoh tampilan ner.make-gold pada Prodigy

4.3.4.1. Latih *dataset* NER

Hasil anotasi NER yang telah diperkaya perlu diekspor dan dilatih kembali dengan perintah *ner.batch-train* seperti yang telah ditunjukkan pada Kode Sumber 4.10. Hasil ekspor model NER akan disimpan pada direktori "model-for-location-final". Prodigy akan mengekspor hasil model latih terbaik ke direktori *output* dan menyertakan *file* data latih dan evaluasi dengan format JSONL. Pengaturan awal jumlah data evaluasi adalah 50% untuk data dibawah 1000 dan 20% untuk data lebih dari 1000. Namun, pengguna juga dapat menentukan sendiri presentase data untuk proses evaluasi.



Gambar 4.5 Contoh tampilan ner.teach pada Prodigy

```
python -m prodigy ner.batch-train dataset_ner
".\model-for-location " --output ".\model-for-
location-final"
```

Kode Sumber 4.10 Latih dan ekspor model NER

4.3.5. Implementasi Pembuatan Model DOC2VEC

Pembuatan model DOC2VEC dilakukan dengan bantuan pustaka Gensim. Data yang diambil adalah data teks *tweet* dari dataset yang telah dibuat. Contoh data latih ditunjukkan pada Tabel 5.1. Cara implementasi pembuatan model DOC2VEC dapat dilihat pada fungsi `createDoc2Vectorizer` pada Kode Sumber 4.11. Setelah data yang telah dilabeli dimuat, data kemudian dipraproses untuk dijadikan model DOC2VEC dengan 100 fitur data. Lalu data baru yang telah diproses diubah menjadi bentuk *csr matrix* agar dapat dikenali oleh Apache Flink yang dapat dilihat pada fungsi `getVector`. Adapun kode sumber lengkap dapat dilihat di Lampiran B:.

```
1.  …
2.    def __init__(self, vectorizer = True, vectorizer_
    type = "doc2vec"):
3.      self.__mNer = self.loadNERModel()
4.      self.__mPostag = self.loadPosTagModel()
5.      self.__gazetter = Gazetter(self.loadPlaces())
6.      self.__matcher = self.loadMatcher()
7.      self.__pcs = Preprocessor()
8.      self.__vector_type = vectorizer_type
```

```
9.      if(vectorizer):
10.       if(vectorizer_type == "doc2vec"):
11.         print("Creating doc2vec modal...")
12.         self.__vectorizer = self.createDoc2Vectoriz
    er()
13.     else:
14.       self.__vectorizer = self.createVectorizer()
15.
16.     def getVector(self,tweet):
17.       if(self.__vector_type=="doc2vec"):
18.         d2v_vectors = [self.__vectorizer.transform(
    [tweet])[0]]
19.         return sparse.csr_matrix(d2v_vectors)
20.       else:
21.         return self.__vectorizer.transform([tweet])
22.
23.     def createDoc2Vectorizer(self):
24.       tweets = pickle.load(open("FIXED-
    DATA\processed-tweets-extended-for-vector-abc-
    balanced.pkl","rb"))
25.       gensim_doc2vec = D2VTransformer(size=100)
26.       processed_tweets = [self.__pcs.preprocess(x)
    for x in tweets]
27.       vector = gensim_doc2vec.fit(processed_tweets)
28.       print("Vector model created!")
29.       return vector
30. …
```

Kode Sumber 4.11 Pembuatan model DOC2VEC

4.3.6. Implementasi Pembuatan Model SVM

Dalam membuat model SVM terdapat 2 proses inti yaitu, *preprocessing* dan penyesuaian data yang dapat dilihat pada Kode Sumber 4.12 serta pelatihan model dengan Apache Flink yang dapat dilihat pada Kode Sumber 4.13. Pada proses penyesuaian data, *dataset* yang berisi data teks *tweet* dan label dipisahkan. Lalu label dibagi lagi menjadi 2, yaitu label untuk jaringan dan produk karena SVM yang didukung Apache Flink saat ini hanya memiliki konsep *one-againts-all*, berarti sebuah model yang hanya bisa menentukan 1 kelas saja. Sehingga diperlukan setidaknya 2 model untuk menentukan apakah data teks merupakan kelas Jaringan,

Produk, atau bukan keduanya. Data teks kemudian ditransformasikan menggunakan model DOC2VEC dan diubah menjadi *csr matrix* agar dapat disimpan kedalam file bersama label. Data disimpan dalam format SVMLIB karena format tersebut didukung oleh Python dan Apache Flink. Terdapat 2 data latih yang digunakan untuk membuat model. Detail data latih yang digunakan dapat dilihat pada Tabel 5.1. Contoh data latih dapat dilihat pada Tabel 5.2. Adapun kode sumber lengkap dapat dilihat di Lampiran C: dan Lampiran D:.

```
1.  …
2.  dfs = pd.read_excel("FIXED-DATA/LABELED-DATA-
    EXTENDED.xlsx")
3.  prepcs = Preprocessor()
4.  def first():
5.    label_holder = dfs.label.to_numpy()
6.    labels = []
7.    labels_2 = []
8.    tweets = [prepcs.preprocess(x) for x in dfs.tweet
    .to_numpy()]
9.    real_tweets = []
10.   _d1 = 0
11.   _j1 = 0
12.   _p1 = 0
13.   for index,label in enumerate(label_holder):
14.     if(pd.isnull(label)):
15.       continue
16.     if(label.replace(" ","").replace("\n","") == "D
    "):
17.       _d1 += 1
18.       real_tweets.append(tweets[index])
19.       labels.append(-1)
20.       labels_2.append(-1)
21.     elif(label.replace(" ","").replace("\n","") ==
    "J"):
22.       _j1 += 1
23.       real_tweets.append(tweets[index])
24.       labels.append(1)
25.       labels_2.append(-1)
```

```
26.     elif(label.replace(" ","").replace("\n","") ==
    "P"):
27.       real_tweets.append(tweets[index])
28.       _p1 += 1
29.       labels.append(-1)
30.       labels_2.append(1)
31.
32.   gensim_doc2vec = D2VTransformer(size=100)
33.   d2v_model = gensim_doc2vec.fit(real_tweets)
34.   d2v_vectors = [d2v_model.transform([x])[0] for x
    in real_tweets]
35.   d2v_vectors = sparse.csr_matrix(d2v_vectors)
36.   print("Jaringan has %d +1 , %d -1(P) , %d -
    1(D)"%(_j1,_p1,_d1))
37.   print("Produk has %d +1 , %d -1(J) , %d -
    1(D)"%(_p1,_j1,_d1))
38.   print(d2v_vectors.shape)
39.   pickle.dump(tweets, open('processed-tweets-
    extended-for-vector-abc-unbalanced.pkl', 'wb'))
40.   dump_svmlight_file(d2v_vectors,labels,'extended-
    data-labeled-jaringan-d2v-abc-
    unbalanced.svmlib',zero_based=False,multilabel=Fals
    e)
41.   dump_svmlight_file(d2v_vectors,labels_2,'extended
    -data-labeled-produk-d2v-abc-
    unbalanced.svmlib',zero_based=False,multilabel=Fals
    e)
42.
43. first()
44. …
```

Kode Sumber 4.12 Preprocessing dan penyesuaian data latih SVM

Selanjutnya data yang telah ditransformasikan dimuat dengan Apache Flink untuk digunakan sebagai data latih. Lalu diakhir Kode Sumber 4.13 hasil prediksi kedua model digabungkan dengan konsep *binary concation* untuk mendapatkan hasil kelas akhir.

```
1.  …
2.  class Model{
3.    private var svm_jaringan :SVM = new SVM()
4.    private var svm_produk : SVM = new SVM()
5.    val env = ExecutionEnvironment.getExecutionEnvir
    onment
6.    println("Loading Dataset for training!")
7.    val trainingDS_jaringan: DataSet[LabeledVector]
    = env.readLibSVM("F:\\Dokumen\\Tugas Akhir\\skrips
    hiittt\\FIXED-MODEL\\SVM\\JARINGAN.svmlib")
8.    val trainingDS_produk: DataSet[LabeledVector] =
    env.readLibSVM("F:\\Dokumen\\Tugas Akhir\\skripshi
    ittt\\FIXED-MODEL\\SVM\\PRODUK.svmlib")
9.    println("Dataset Loaded")
10.   train()
11.
12.   def train(): Unit ={
13.     svm_jaringan.fit(trainingDS_jaringan)
14.     svm_produk.fit(trainingDS_produk)
15.   }
16.   def predict_produk(value : DataSet[Vector]): Dat
    aSet[(Vector,Double)]  ={
17.     println("Value accepted, start predicting - Pr
    oduk")
18.     var temp_produk = svm_produk.predict(value)
19.     return temp_produk
20.   }
21.
22.   def predict_jaringan(value : DataSet[Vector]): D
    ataSet[(Vector,Double)]  ={
23.     println("Value accepted, start predicting - Ja
    ringan")
24.     var temp_jaringan = svm_jaringan.predict(value
    )
25.     return temp_jaringan
26.   }
27. }
28. …
```

Kode Sumber 4.13 Pembuatan model pada Apache Flink

4.3.7. Implementasi Sistem

Implementasi sistem dilakukan berdasarkan perancangan sistem dan arsitektur yang dijelaskan pada bab Analisis dan Perancangan Sistem.

4.3.7.1. Implementasi Penghubung Data

Data antar sistem dan proses yang dilakukan dihubungkan dengan sebuah logika seperti penghubung data dengan konsep *subscribe*. Penghubungan data dilakukan dengan bantuan Apache Kafka. Cara pembuatan *penghubung* data dapat dilihat pada Kode Sumber 4.14. Setiap *penghubung* data harus diberi nama unik. Nama *penghubung* dapat diisikan dengan mengganti "`[name]`" dengan nama lainnya seperti yang terlihat di akhir kode sumber. Pada sistem ini, terdapat 3 *penghubung* data yang dijelaskan pada Tabel 4.3.

Tabel 4.3 Daftar penghubung antar data dan platform pemrosesan

Nama	Keterangan
scala-streaming-vectorization	Menghubungkan data awal dari *producer* ke Apache Flink
scala-streaming-dump-flask	Menghubungkan data dari Apache Flink ke tampilan dan proses deteksi
streaming-trouble	Menghubungkan proses deteksi ke tampilan

```
kafka-topics.bat --create --zookeeper localhost:2181 --
replication-factor 1 --partitions 1 --topic [name]

kafka-topics.bat --create --zookeeper localhost:2181 --
replication-factor 1 --partitions 1 --topic scala-
streaming-vectorization
```

Kode Sumber 4.14 Pembuatan penghubung data

4.3.7.2. Implementasi Twitter Listener

Implementasi Twitter Listener atau *producer* dilakukan dengan bantuan pustaka Tweepy. *Producer* akan mendengarkan *event tweet* secara *realtime* dan mengirimkan data menuju tahap selanjutnya untuk diklasifikasikan. Data yang didapatkan oleh *producer* merupakan objek *tweet*. Detail komponen objek *tweet* dapat dilihat di halaman dokumentasi Twitter untuk *developer* [35]. Cara implementasi *producer* dapat dilihat pada Kode Sumber 4.15. Pada kode sumber tersebut, buat objek Apache Kafka dan Tweepy terlebih dahulu. Selanjutnya Tweepy menyaring teks *tweet* yang akan didengarkan sesuai dengan kata kunci yang penulis tentukan, yaitu @IndosatCare, @Telkomsel, @smartfrencare. Kemudian data diubah dari format JSON untuk diambil data yang diperlukan. Lalu data yang didapat diproses untuk mendapatkan entitas-entitas penting seperti lokasi, NER, dan transformasi data teks *tweet* dalam bentuk *vector*. Fungsi praproses dan transformasi *vector* ditulis dalam sebuah kelas yang akan dibahas pada subbab 4.3.7.3 dan subbab 4.3.7.4.

```
1.  access_token = "#access-token"
2.  access_secret = "#access-secret"
3.  consumer_key = "#consumer-key"
4.  consumer_secret = "#consumer-secret"
5.
6.  auth = tweepy.OAuthHandler(consumer_key,consumer_s
    ecret)
7.  auth.set_access_token(access_token,access_secret)

8.
9.  api = tweepy.API(auth,wait_on_rate_limit=True,wait
    _on_rate_limit_notify=True)
10. pc = Processor()
11.
12. _kafkaclient = KafkaClient(hosts='127.0.0.1:9092')

13. _producer_classifier = _kafkaclient.topics['scala-
    streaming-vectorization']
14.
```

```
15. class MyStreamListener(tweepy.StreamListener):
16.   def on_status(self,status):
17.     status=status._json
18.     obj = {}
19.     obj['_ner'] = displacy.parse_deps(pc.getNerEnt
    ities(obj['full_text']))
20.     obj['_received'] = time.time()
21.
22.     print("New tweet received...")
23.     obj['tweet_id'] = status['id_str']
24.     obj['screen_name'] = status['user']['screen_na
    me']
25.     obj['mentions'] = status['entities']['user_men
    tions']
26.     obj['directed_to'] = []
27.     obj['_time'] = status['created_at']
28.     for mention in obj['mentions']:
29.       if(mention['screen_name'].lower() in ["smart
    frencare","indosatcare","telkomsel"]):
30.         obj['directed_to'].append(mention['screen_
    name'].lower())
31.     if "extended_tweet" in status:
32.       obj['full_text'] = status['extended_tweet'][
    'full_text'].replace("\n"," ").encode("utf-8")
33.     elif "full_text" in status:
34.       obj['full_text'] = status['full_text'].repla
    ce("\n"," ").encode("utf-8")
35.     else:
36.       obj['full_text'] = status['text'].replace("\
    n"," ").encode("utf-8")
37.     obj['full_text'] = str(obj['full_text'])
38.     obj['processed_text'] = obj['full_text']
39.     target = pc.getVector(obj['processed_text'])
40.     array_of_tups = []
41.     for index,value in enumerate(target.toarray().
    tolist()[0]):
42.       array_of_tups.append((index,value))
43.     obj['vector'] = array_of_tups
44.     obj['_address'] = pc.getAddress(obj['full_text
    '])
45.     with _producer_classifier.get_sync_producer()
    as producer:
```

```
46.         producer.produce(json.dumps(obj).encode("utf
    -8"))
47.
48.   def on_error(self, status_code):
49.     if status_code == 420:
50.       print ("Limit Excedeed")
51.       return False
52. myStreamListener = MyStreamListener()
53. myStream = tweepy.Stream(auth=api.auth , listener=
    myStreamListener, tweet_mode='extended')
54. myStream.filter(track=["@IndosatCare","@Telkomsel"
    ,"@smartfrencare"])
```

Kode Sumber 4.15 Twitter listener atau producer data

4.3.7.3. Implementasi Preprocessor

Preprocessor merupakan implementasi rancangan *preprocessing*. Implementasi kelas *preprocessor* dilakukan dengan bantuan pustaka Twitter Text Python (TTP) dan Sastrawi. Kelas ini memiliki beberapa fungsi yang dijelaskan pada Tabel 4.4. Fungsi utama kelas dapat dilihat pada Kode Sumber 4.12. Adapun cara implementasi kelas dapat dilihat pada Lampiran A:.

Tabel 4.4 Daftar fungsi preprocessor

Nama	Parameter	Keterangan
__init__	-	Intansiasi kelas sastrawi dan TTP dan memuat data *slang*
preprocess	tweet , mentions (opsional)	Fungsi utama untuk praproses *tweet*.
Stemming	tweet	Melakukan *stemming* pada pada tweet

loadSlangFromJson	-	Memuat data *slang* yang dipanggal saat instansiasi kelas
slang_change	tweet	*Slang* pada Teks *tweet* diganti berdasarkan korpus data yang telah dimuat
remove_symbol_and_number	tweet	Menghapus simbol dan angka menggunakan *regex*
remove_link_hashtag_mentions	tweet, mention (opsional)	Menghapus entitas URL dan *mention* dengan menggunakan TTP, jika pengguna mengirimkan parameter mention, maka selain menggunakan TTP untuk menghapus mention, program akan memanggil *remove_mentions_provided*
remove_mentions_provided	tweet, mentions	Menghapus mention pada tweet yang disediakan

```
1.  …
2.   def preprocess(self,tweet, mentions = None):
3.       holder = tweet
4.       holder = self.remove_link_hastag_mentions(holder.replace("\n","").lower(),mentions)
5.
6.       holder = self.remove_symbol_and_number(holder)

7.
8.       holder = self.slang_change(holder)
9.
10.      holder = self.stemming(holder)
11.
```

```
12.     return holder
13. …
```

Kode Sumber 4.16 Fungsi utama implementasi preprocessing

4.3.7.4. Implementasi Ekstraksi Fitur dan Deteksi Lokasi

Fitur ekstraksi fitur dan deteksi lokasi yang telah dirancang pada bab 3 diimplementasikan ke dalam kelas *processor*. Implementasi dilakukan dengan bantuan pustaka online spaCy, scipy, dan gensim serta kelas Gazetter yang telah dijelaskan di subbab 4.3.1. Kelas ini memiliki beberapa fungsi yang dijelaskan pada Tabel 4.5. Implementasi fungsi ekstraksi fitur dan deteksi lokasi dapat dilihat pada Kode Sumber 4.17. Adapun implementasi kelas dapat dilihat pada Lampiran B: dan Lampiran C:.

Tabel 4.5 Daftar fungsi processor

Nama	Parameter	Keterangan
__init__	vectorizer (opsional), vectorizer_type (opsional)	Melakukan instansiasi model-model yang dibutuhkan. Parameter vectorizer menentukan apakah perlu melakukan instansiasi model *vectorizer* untuk pembobotan teks atau tidak. Sedangkan vectorizer_type mendefinisikan jenis model *vectorizer* (tf-idf / Doc2Vec).
getVector	tweet	Melakukan transformasi data tweet dengan model *vectorizer* menjadi fitur

		data yang dapat diolah oleh komputer.
createDoc2Vectorizer	-	Fungsi untuk memuat model Doc2Vec
createVectorizerr	-	Fungsi untuk memuat model TF-IDF
loadMatcher	-	Fungsi untuk memuat aturan aturan yang akan digunakan untuk *Rule-based Matching*
loadPlaces	-	Fungsi untuk memuat data lokasi untuk melakukan *string matching* yang telah dijelaskan pada Subbab 4.3.1
getAddress	tweet	Melakukan ekstraksi lokasi berdasarkan voting hasil dari NER, POSTAG, dan *Rule-based Matching*.
getLocationBasedOnNER	tweet	Mengekstrak lokasi dari teks *tweet* menggunakan NER
getLocationBasedOnPostag	tweet	Mengekstrak lokasi dari teks *tweet* menggunakan NER

getLocationBasedOnMatcher	tweet	Mengekstrak lokasi dari teks *tweet* menggunakan POSTAG dan Gazetter
getNerEntities	tweet	Mengekstrak entitas NER pada *tweet*
getPosTagEntitiesSplit	tweet	Mengekstrak entitas POSTAG pada *tweet*
getTokenMatchEntities	tweet	Mengekstrak lokasi dari *Tag* teks *tweet* menggunakan *Rule-Based Matching*
loadPosTagModel	-	Fungsi untuk memuat model POSTAG
loadNERModel	-	Fungsi untuk memuat model NER
getNerForWeb	tweet	Fungsi untuk membuat tampilan NER dengan HTML dan CSS yang telah didefinisikan

```
1.  …
2.     def getVector(self,tweet):
3.      if(self.__vector_type=="doc2vec"):
4.        d2v_vectors = [self.__vectorizer.transform([t
    weet])[0]]
5.        return sparse.csr_matrix(d2v_vectors)
6.      else:
7.        return self.__vectorizer.transform([tweet])

8.    def getAddress(self,tweet):
```

```
9.      _ner = self.getLocationBasedOnNER(tweet).replac
    e("\n","")
10.     _postag = self.getLocationBasedOnPosTag(tweet).
    replace("\n","")
11.     _matcher = self.getLocationBasedOnMatcher(tweet
    ).replace("\n","")
12.
13.     holder = {}
14.     if(len(_ner)!=0):
15.       holder[_ner] = 1
16.
17.     if(_postag in holder.keys()):
18.       holder[_postag] = holder[_postag] + 1
19.     elif(len(_postag)!=0):
20.       holder[_postag] = 1
21.
22.     if(_matcher in holder.keys()):
23.       holder[_matcher] = holder[_matcher] + 1
24.     elif(len(_matcher)!=0):
25.       holder[_matcher] = 1
26.     print(holder)
27.     ma = -1
28.     ke = ""
29.     for key in holder.keys():
30.       print("key %s : %d"%(key,holder[key]))
31.       if(holder[key] > ma):
32.         ma = holder[key]
33.         ke = key
34.         print(ma)
35.         print(ke)
36.
37.     if(ma == 1 and len(_ner)!=0):
38.       return _ner
39.
40.     return ke
41. …
```

Kode Sumber 4.17 Fungsi ekstraksi fitur dan deteksi lokasi

4.3.7.5. Implementasi Klasifikasi

Implementasi klasifikasi menggunakan bantuan *platform* Apache Flink yang dibuat khusus untuk pemrosesan *datastream*. Klasifikasi diimplementasikan dengan menggunakan bahasa scala.

Pemrosesan dibagi menjadi beberapa kelas yang dapat dilihat pada Lampiran C: dan Lampiran D:. Adapun potongan fungsi klasifikasi pada program dapat dilihat pada Kode Sumber 4.18. Diagram kelas yang dibuat digambarkan pada Gambar 4.6. Tiap tahapan harus dibagi menjadi beberapa proses karena pengolahan *datastream* pada Apache Flink memetakan logika pada tiap tiap data melalui *flatMap*. Sehingga untuk menjalankan program yang kompleks seperti klasifikasi memerlukan sebuah kelas-kelas baru yang didedikasikan khusus yang mewarisi kelas *FlatMapFunction* untuk mengolah data. Sedangkan kelas *CheckPointedFunction* diperlukan agar status model SVM yang akan dibuat dapat disimpan dan dijalankan oleh Apache Flink (*serializable*). Proses dimulai ketika Flink mendapatkan data, lalu memicu proses penyesuaian data dan klasifikasi yang dilanjutkan dengan mengembalikan bentuk data kedalam bentuk JSON untuk dikirimkan. Fungsi masing-masing kelas dijelaskan pada Tabel 4.6.

Tabel 4.6 Daftar kelas pada Aplikasi Flink

Kelas	**Keterangan**
Main	Kelas utama yang memanggil memetakan data yang diterima dan mengirimkan data ke *platform* lain
JsonStringAndClassify	Digunakan untuk menyesuaikan data yang diterima dalam format JSON dan format-format lain seperti Array menjadi data yang dapat diolah oleh Apache Flink. Kelas ini juga berfungsi untuk mengembalikan data hasil pengolahan menjadi format JSON.

map	Digunakan untuk merangkum hasil klasifikasi kelas Model.
Model	Digunakan untuk mengklasifikasikan *tweet* dengan model Jaringan dan Produk.

Untuk menjalankan aplikasi Flink, penulis memilih menggunakan *executeable file* dalam bentuk JAR karena proses pembuatan dan pengaturan yang mudah. Proses pembuatan *executeable file* dibuat dengan skema *maven repository*. Sebelum menjalankan *executeable file* Aplikasi Flink yang telah dibuat, *server* Apache Flink harus dinyalakan dengan perintah pada Kode Sumber 4.19. Terakhir aplikasi dapat dijalankan dengan Kode Sumber 4.20.

```
1.  …
2.
3.   def flatMap2(value: DataSet[Vector], out: Collect
    or[DataSet[(Vector, Double)]]): String = {
4.       println("prediction is being processed... ")
5.       var temp_jaringan : DataSet[(Vector,Double)] =
     model.predict_jaringan(value)
6.       var temp_produk : DataSet[(Vector,Double)] = m
    odel.predict_produk(value)
7.       var result_jaringan : Seq[(Vector,Double)] = t
    emp_jaringan.collect()
8.       var result_produk : Seq[(Vector,Double)] = tem
    p_produk.collect()
9.       var result_akhir : String = ""
10.      if(result_jaringan(0)._2  == 1.0  && result_pr
    oduk(0)._2 == 1.0 ){
11.        result_akhir = "keduanya"
12.      }else if(result_jaringan(0)._2  == 1.0  && res
    ult_produk(0)._2 == -1.0){
13.        result_akhir = "jaringan"
14.      }else if(result_jaringan(0)._2  == -
    1.0  && result_produk(0)._2 == 1.0){
15.        result_akhir = "produk"
```

```
16.     }else{
17.       result_akhir = "nihil"
18.     }
19.     println(result_jaringan(0)._2)
20.     println(result_produk(0)._2)
21.     println(result_akhir)
22.     return result_akhir
23.   }
24. …
```

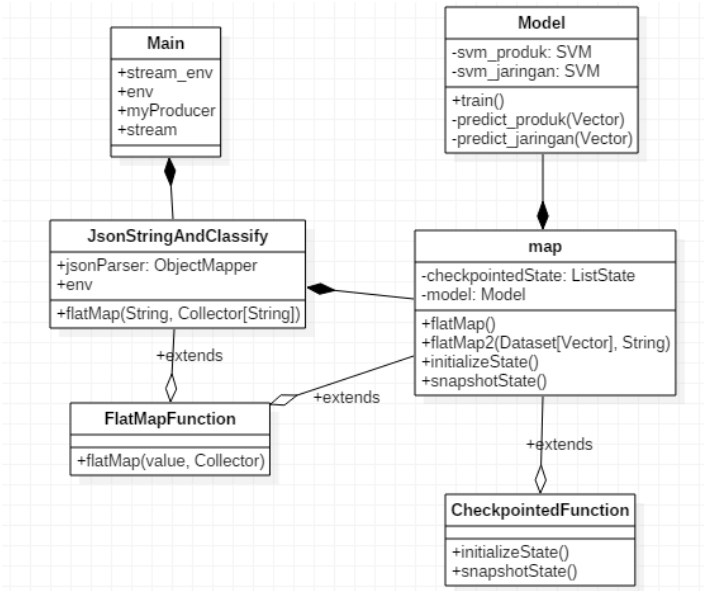
Kode Sumber 4.18 Implementasi Binary Concation untuk klasifikasi Akhir

```
start-cluster
```

Kode Sumber 4.19 Cara menyalakan Apache Flink

```
flink run "F:\Dokumen\Tugas Akhir\JARS\testing.jar"
```

Kode Sumber 4.20 Cara menjalankan Aplikasi Flink



Gambar 4.6 Diagram Kelas program klasifikasi pada Apache Flink

4.3.7.6. Implementasi Deteksi Gangguan Jaringan

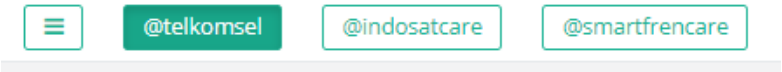
Implementasi pembuatan deteksi gangguan jaringan dapat dilihat pada Lampiran E:. Data *tweet* yang diproses adalah

teks *tweet* dengan hasil klasifikasi “jaringan”. Masing-masing waktu unggahan *tweet* disimpan dan dikumpulkan berdasarkan lokasi yang telah diekstrak pada proses yang dijelaskan di subbab 4.3.7.2. Kumpulan data tersebut kemudian diupdate berkala sesuai rentang waktu *t* yang ditentukan. Sistem akan mendeteksi gangguan jaringan apabila dalam rentang waktu tersebut, terdapat *n* buah *tweet* dalam satu tempat yang sama. Pengaturan awal rentang waktu *t* dan *n* adalah 10 menit dan 5 buah *tweet*. Namun, pengguna dapat mengubahnya ketika menginstansisi kelas Trouble Filter.

4.3.8. Implementasi Antarmuka

Implementasi antarmuka sistem dibuat untuk mempermudah uji coba dan evaluasi. Implementasi dibuat dengan menggunakan kerangka kerja Flask, Apache Kafka, dan VueJs. Pada halaman depan seperti yang ditunjukkan pada Gambar 4.10, terdapat beberapa bagian utama untuk mengoperasikan laman web. Pada Gambar 4.11 terdapat bagian navigasi halaman untuk berganti halaman sesuai jenis *tweet*. Sedangkan ada 3 tombol pada bagian atas atau *topnavbar* halaman untuk mengganti hasil klasifikasi berdasarkan *customer service* seperti yang ditunjukkan pada Gambar 4.8.

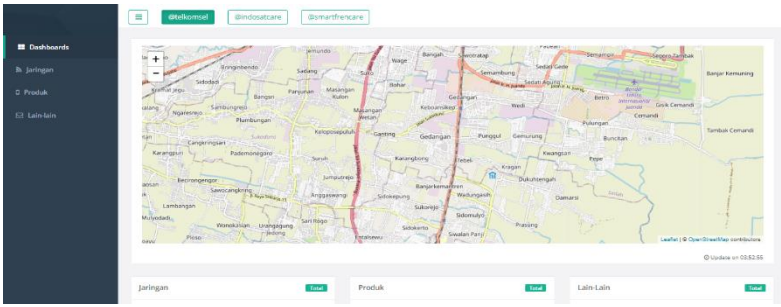
Gambar 4.7 Tampilan daftar tweet hasil pemrosesan



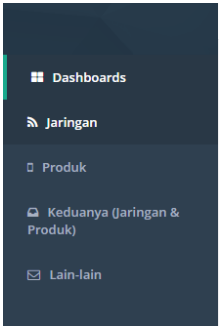
Gambar 4.8 Tampilan tombol pemilihan customer service

Pengguna dapat melihat daftar hasil klasifikasi *tweet* dengan menekan salah satu tombol navigasi halaman. Tampilan daftar *tweet* dapat dilihat pada Gambar 4.7. Setiap komplain yang ditujukan pada *customer service* ditampilkan bersama hasil deteksi lokasi, ekstraksi entitas NER, dan sebuah tombol untuk menuju ke halaman asli *tweet* di Twitter. *Tweet* yang telah dilihat akan berpindah ke bagian "Sudah dilihat" untuk memudahkan pengguna melakukan penyelesaian komplain. Selain itu, pengguna juga dapat melihat hasil deteksi gangguan berdasarkan lokasi pada *maps* seperti pada Gambar 4.9 yang disediakan di halaman utama sistem.

Gambar 4.9 Tampilan hasil deteksi lokasi



Gambar 4.10 Tampilan dashboard sistem



Gambar 4.11 Tampilan navigasi halaman

BAB V
PENGUJIAN DAN EVALUASI

Bab ini membahas pengujian dan evaluasi pada aplikasi yang dikembangkan. Pengujian fungsionalitas mengacu pada kasus penggunaan pada bab tiga. Pengujian kegunaan program dilakukan dengan mengetahui tanggapan dari pengguna terhadap sistem. Hasil evaluasi menjabarkan tentang rangkuman hasil pengujian pada bagian akhir bab ini.

5.1. Lingkungan Pengujian

Lingkungan penguijan sistem pada pengerjaan tugas ini dilakukan pada lingkungan dan alat kakas pada berikut:

- Processor Intel® Core™ i7-4720HQ CPU @ 2.60GHz
- Installed RAM 12.0 GB (11.9 GB usable)
- System Type 64-bit operating system, x64-based processor
- Windows Edition Windows 10 Education
- Apache Flink Versi *standalone* 1.7.2

5.2. Data Uji Coba

Data yang dilakukan dalam uji coba adalah data latih dan data tweet diluar data latih yang diambil menggunakan API resmi Twitter pada September – Oktober 2019. Data latih merupakan data hasil *crawling* yang dilabeli secara manual. Daftar data latih dapat dijelaskan pada Tabel 5.1. Contoh data latih dapat dilihat pada Tabel 5.2.

Tabel 5.1 Daftar data latih

Nama	Detail Data	Keterangan
Data Latih 1	Total 4915 dari 3 jenis kelas, yaitu jaringan, produk, dan bukan keduanya	Merupakan data yang dilabeli manual yang terdiri dari 1433 Jaringan, 1503 Produk, dan 1979 bukan keduanya.
Data Latih 2	Total 2936 dari 2 jenis kelas, yaitu	Merupakan data yang dilabeli manual yang

	jaringan dan produk	terdiri dari 1433 Jaringan, dan 1503 Produk.

Tabel 5.2 Contoh data latih

Teks tweet	Label
@indosatcare min mohon segera diproses ya komplain di dm nya. terima kasih.	D (Bukan Keduanya)
ini jaringan @indosatcare lagi bapuk banget deh dari tadi pagi!!!	J (Jaringan)
@smartfrencare kartu sy tdk bsa aktivasi paket. tlg d bantu. sy sdh dm. tq,xiaobells	P (Produk)

5.3. Skenario Pengujian

Subbab ini akan menjelaskan skenario uji yang telah dilakukan. Terdapat 5 skenario uji coba yang dilakukan. Berikut merupakan penjelasan setiap skenario pengujian:

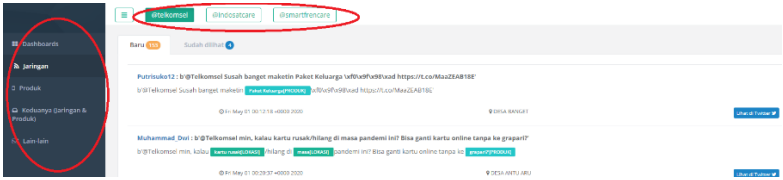
1. Skenario Pengujian 1
   Dalam seknario ini akan dilakukan pengujian terhadap kasus penggunaan sistem yang telah dirancang pada subbab 3.2. Pengujian dilakukan dengan menjalankan sistem secara *blackbox* sebagai pengguna sistem.
2. Skenario Pengujian 2
   Dalam skenario ini akan dilakukan pengujian terhadap kebenaran apakah sistem dapat bekerja dengan baik. Pengujian dilakukan dengan menjalankan sistem klasifikasi dan deteksi gangguan. Selama pengujian, penulis mengamati proses masing-masing tahapan untuk memastikan bahwa sistem telah berjalan sesuai dengan yang direncanakan pada bab 3.
3. Skenario Pengujian 3
   Dalam skenario ini akan dilakukan pengujian terhadap model SVM. Evaluasi model SVM dilakukan dengan menghitung

nilai *accuracy*, *precision*, *recall*, dan *f-measure* untuk setiap hasil klasifikasi. Selain itu, pengujian dilakukan terhadap metode-metode yang digunakan untuk mengekstrak lokasi dari sebuah *tweet*.

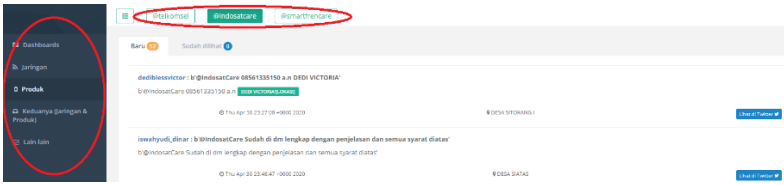
4. Skenario Pengujian 4
   Dalam skenario ini akan dilakukan pengujian dengan melakukan *stress test* untuk mengukur kecepatan sistem dalam memroses data.

5.3.1. Skenario Pengujian 1

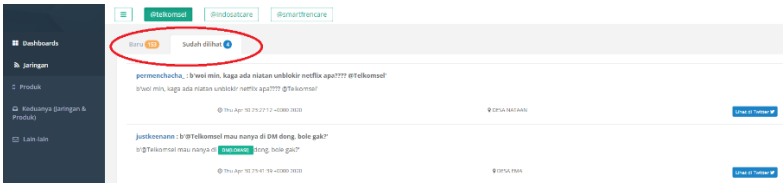
Pada skenario ini dilakukan evaluasi sistem dengan menguji kasus penggunaan yang telah dirancang pada subbab 3.2. Evaluasi dilakukan dengan mengamati kebenaran penggunaan sistem. Dari pengujian yang dilakukan, pengguna dapat menggunakan sistem sesuai kasus penggunaan yang telah dirancang seperti yang digambarkan hasil antarmuka sistem pada Gambar 5.1 - Gambar 5.4.



Gambar 5.1 Hasil pengujian UC01



Gambar 5.2 Hasil Pengujian UC01



Gambar 5.3 Hasil Pengujian UC03

Gambar 5.4 Hasil Pengujian UC02

5.3.2. Skenario Pengujian 2

Pada skenario ini dilakukan evaluasi sistem dengan mengamati jalannya proses klasifikasi dan deteksi gangguan secara *real-time*. Evaluasi proses klasifikasi dilakukan dengan memberikan keluaran tiap tiap proses yang dijelaskan pada Gambar 3.11.

Dari pengujian yang dilakukan, sistem dapat melakukan proses klasifikasi yang dapat dilihat keluaran internal prosesnya pada Gambar 5.5 - Gambar 5.7. Untuk lebih jelasnya, hasil keluaran 3 proses pertama dapat dilihat pada Tabel 5.3, Tabel 5.4, dan Lampiran F:. Adapun hasil klasifikasi dapat dilihat pada Tabel 5.5. Sedangkan pengujian deteksi gangguan tidak bisa dilakukan karena tidak lokasi yang terdeteksi.

Tabel 5.3 Keluaran preprocessing

Tweet sebelum preprocessing	Setelah preprocessing
@Telkomsel Tolong cek dm lagi ya min, sudah dibalas. Terima kasih	tolong cek dm lagi ya min sudah balas terima kasih

```
New tweet received...
b'@Telkomsel Tolong cek dm lagi ya min, sudah dibalas. Terima kasih'
[(0, 0.004488218557516019), (1, -0.003549988381564617), (2, 0.001261401921510696 4), (3, -0.00028355600079521537), (
4, -0.00061236845795506), (5, 0.000286042195511918 4), (6, -0.001868748455308377), (7, 9.17992438189685 3e-05), (
8, -0.004494702909141779), (9, 0.003398259636014 7), (10, 0.004826879594475031), (11, -0.00029529773746617 14), (12,
0.004694464150816202), (13, -0.004035390447825193), (14, -0.003023895202204585), (15, -0.00029297420405278 76), (16,
 -0.00060586840845644 47), (17, -0.004450909327715635), (18, -2.5442081096116453e-05), (19, 0.0013379890006035566),
(20, -0.002281236927956342 7), (21, 0.000213354127481579 78), (22, -0.001432527671568095 7), (23, 0.00114740757271647 4
5), (24, 0.003152522025629878), (25, -0.00017884463886730373), (26, -0.00025790499057620764), (27, 0.00088949705241
24801), (28, 0.004196892026811838), (29, -0.002484969974927604 2), (30, -0.004512969870120287), (31, -0.0006922286120
24337), (32, -0.003008622210472822), (33, 0.004979896824806929), (34, -0.00139610131736844 78), (35, 3.5373182072362
397e-06), (36, 0.001359525951556861 4), (37, 0.004095901735126972), (38, 0.003629321232438075), (39, -0.00087373878
34116817), (40, 0.003782538697123527 5), (41, -0.000799491419456899 2), (42, -0.001804306288249790 7), (43, 0.00227454
4909596443), (44, 0.002288896590471267 7), (45, 0.001521100639365613 5), (46, 0.003185692708939314), (47, 0.004321603
570133448), (48, -0.003870298853144049 6), (49, -0.004211900755763054), (50, 0.004982391837938423 5), (51, -0.0026439
12797793746), (52, 0.001841821125708520 4), (53, 0.004411335568875074), (54, -0.001078050932846963 4), (55, 0.0008836
402557790279), (56, -0.002192856976762414), (57, 0.002156961942091584), (58, 0.000600164174102246 8), (59, 0.0036927
917972207071), (60, -0.000687439285684377), (61, 0.004095903465896845), (62, -0.001048400705171227 46), (63, -0.001527
4450415745378), (64, -0.001031304924439669), (65, 0.004557535052299495), (66, -0.003286183113232255), (67, -0.004
899676423519885), (68, -0.001550821820273995 4), (69, -0.001045812969096005), (70, -0.001002908335067331 8), (71, 0.00
321646546944975 85), (72, 0.000253038073424249 9), (73, 0.002395302057266235 4), (74, 0.004948440939188003 5), (75, -9.
444341412745416e-05), (76, -0.003886624472215771 7), (77, -0.001689163618721068), (78, 0.002050216076895946), (79,
0.004487547557801008), (80, -0.001222249586135149), (81, -0.001112328609451651 6), (82, 0.002008705632761121), (83,
0.00081011565634980 8), (84, -0.004695161245763302), (85, -0.003619502298533916 5), (86, -0.001061790622770786 3), (87
, -0.001169423805549740 8), (88, -0.004974120762199163 4), (89, -0.003991978708654642), (90, -0.00025717329117469 49),
 (91, -0.004059544298797846), (92, 0.004082484636455774), (93, 0.001857859198935533), (94, -0.00016192789189517498),
 (95, -0.001848145737312734 1), (96, -0.004513651132583618), (97, -0.004241369664669037), (98, -0.001846986007876694
2), (99, -0.001169270602986216 5)]
Processing Address
Address from NER:
Address from POSTAG:
Address from RBM:
Nilai vote masing-masing metode:
{}
FINAL LOCATION:  || Took 0.014876 seconds
```

Gambar 5.5 Keluaran internal tweet listener, preprocessing, ekstraksi fitur, dan deteksi lokasi

```
{"_received": 1593907860.4851568, "tweet_id": "1127430783291805286 4", "screen_name": "roziMF", "mentions": [{"screen_name":
 "Telkomsel"}], "directed_to": {"telkomsel"}, "_time": "Sat Jun 20 11:47:17 +0000 2020", "full_text": "b'@Telkomsel Tolong
cek dm lagi ya min, sudah dibalas. Terima kasih'", "processed_text": "b tolong cek dm lagi ya min sudah balas terima kasih"
, "preprocess_time": 0.031509399414062 5, "vector": [(0, 0.001647904748097062 1), (1, 0.001728427596390247 3), (2, 0.002894950
332120061), (3, -0.001586972502991557 1), (4, 0.004265718627721871), (5, 0.003605183200761676), (6, 0.002275197301059613),
(7, -0.003731474280357361), (8, 0.002278872067108750 3), (9, -0.003758589969947934), (10, -0.001141579705290496 3), (11, 0.00
443025259301066 4), (12, 0.003227602690458297 7), (13, 0.004454377179266884 9), (14, -0.00037486970540106742), (15, 0.001122325
8916288614), (16, 0.001325779128819704), (17, 0.000178200323716373 3), (18, 0.003790494908587846 76), (19, -0.00015847846225 3
69722), (20, -0.001489106216467917), (21, -0.000222073140321299 43), (22, -0.000111543653474655), (23, 0.001264611375518143
), (24, -0.003086593467742047), (25, 0.004270324949175119), (26, 0.001180900002797544), (27, 0.003288030857220292), (28, 0.
002980067285115465), (29, -0.002953832503401939 6), (30, -0.002299739513546228 4), (31, 0.004524546209722757), (32, 0.00300 2
984621971846), (33, -0.004758044146001339), (34, -7.802936306888935e-05), (35, -0.001866230159066175), (36, -0.00434908503
6665201), (37, 0.000142825374496169 4), (38, 0.003104143077507615), (39, -0.004260172601789236), (40, 0.001806886284612192)
, (41, -0.003414837717842855 5), (42, -0.004323133267462254), (43, -0.000598442275077104 6), (44, 0.000647124834358922), (45
, 0.001556067029014229 8), (46, -0.003019222291186452), (47, -0.000406364066293463 1), (48, 0.004439061936980485 9), (49, -0.00
189066916650533683), (50, 0.000623029598500579 6), (51, -0.000601484731305390 6), (52, -0.000809912162672545), (53, 0.0041683
97746980819), (54, -0.002564430935308337), (55, -0.000304476969176093 37), (56, -0.002335733405764818), (57, 0.004791471715844
9438), (58, 0.002136794151738286), (59, 0.004217254463583231), (60, 0.003990577799950242), (61, -0.004499819595366716), (62,
 -0.003780637169256806 4), (63, 0.001837716205045809), (64, -0.004637375939063742 6), (65, 0.002010293770581484), (66, -0.003
7874369882047176), (67, -0.003978535990861264 2), (68, 0.002567402319982648), (69, 0.003404018701985478 4), (70, 0.00196913420
22269964), (71, -0.003303528530523181), (72, 0.001281508943066001), (73, -0.001996633131105702 4), (74, 0.002774593653157353 4
), (75, 0.004220111761242151), (76, -0.000110072862298693 51), (77, 0.002327261958271265), (78, 0.004374284574687481), (79,
9.981544280890375e-05), (80, -0.004137963987886906), (81, -0.004073546268045902), (82, -0.004451349377632141), (83, -0.0038
166434969753027), (84, -0.001129405340179801), (85, 0.004455327522009611), (86, 0.003552620531991124), (87, -0.004184225574
13578), (88, -0.004187422804534435), (89, -0.001676393556048541), (90, 0.004288604821298122), (91, 0.00073387076742947), 
 (92, -0.002166519174352882), (93, 0.000578955096433103 1), (94, 0.003473787102848291 4), (95, 0.003686937270680304), (96,
0.002883622655645013), (97, -0.008809064409336447 7), (98, 0.003288922365754842 8), (99, -0.000615452532656490 8)], "_address"
: "", "_address_time": 0.020944118409755586, "_ner": "b'@Telkomsel Tolong cek dm lagi ya min, sudah dibalas. Terima kasih'",
"_svm_time": 1593907860.5496905}
Starting Prediction
prediction is being processed...
Loading Dataset for training!
Dataset Loaded
Value accepted, start predicting - Jaringan
Value accepted, start predicting - Produk
1.0
1.0
keduanya
Prediction Ended. The class is keduanya
```

Gambar 5.6 Keluaran internal Apache Flink ketika menerima data dan hasil klasifikasinya

Tabel 5.4 Deteksi Lokasi

Metode	Hasil Lokasi
NER	-
POSTAG	-
Rule Based Matching	-
Hasil Akhir Lokasi	-

Tabel 5.5 Keluaran klasifikasi Apache Flink

Model	Hasil
Jaringan	1 Jaringan
Produk	1 Produk
Hasil Akhir	1 Keduanya

Gambar 5.7 Tampilan hasil pengolahan pada web

5.3.3. Skenario Pengujian 3

Pada skenario ini dilakukan evaluasi model SVM menggunakan *cross-validation*. Evaluasi model SVM dilakukan dengan cara menghitung akurasi, *precision*, *recall*, dan *f-measure* untuk setiap model yang telah dibuat pada subbab 4.3.6. Selain itu, hasil akhir klasifikasi akan diukur dan diuji secara manual dengan memverifikasi kualitas hasil klasifikasi dan deteksi lokasi sistem. Hasil pengujian deteksi lokasi dapat dilihat pada Tabel 5.8.

Data uji coba yang digunakan dalam pengujian *cross validation* dapat dilihat pada Tabel 5.1. Selain itu pengujian dilakukan dengan menggunakan 200 *tweet* diluar data latih. Model akan dievaluasi menggunakan data uji dengan dan tanpa *preprocessing* untuk mengetahui pengaruh tahapan tersebut terhadap metris penilaian yang digunakan. Selain menggunakan Apache Flink, pengujian dilakukan dengan membandingkan hasil klasifikasi menggunakan SVM Linear dengan pustaka sklearn. Dari uji coba yang dilakukan, didapatkan hasil tertinggi menggunakan sklearn sebesar 71% sedangkan 50.9% menggunakan Apache Flink. Adapun hasil akhir klasifikasi menggunakan *binary concation* dengan data uji diluar data latih mendapatkan akurasi sebesar 43%.

Tabel 5.6 Hasil evaluasi masing-masing model dengan preprocessing

Model	**Data Latih**	**Alat**	***Cross Validation 10 Fold***			
			Acc	***Prec***	***Rec***	***FMea***
Jaringan	1	Flink	49%	28%	48%	35.5%
		Sklearn	**71%**	**70%**	**50%**	**42.3%**
	2	Flink	49%	48%	49%	48%
		Sklearn	52%	52%	52%	48%
Produk	1	Flink	50.8%	31%	50.7%	38.7%
		Sklearn	**69%**	**39.7%**	**50%**	**41%**
	2	Flink	48.6%	49%	48%	49%
		Sklearn	52%	52%	52%	55%

Tabel 5.7 Hasil evaluasi masing-masing model tanpa preprocessing

Model	**Data Latih**	**Alat**	***Cross Validation 10 Fold***			
			Acc	***Prec***	***Rec***	***FMea***
Jaringan	**1**	Flink	50%	30%	53%	38.6%
		Sklearn	**70%**	**55%**	**50%**	**41.9%**
	2	Flink	50.9%	49.8%	50.5%	50%
		Sklearn	50%	50%	50%	46%
Produk	**1**	Flink	49.5%	30%	48.8%	37%
		Sklearn	**69%**	**34%**	**50%**	**40.9%**
	2	Flink	50%	51%	50.8%	51%
		Sklearn	50%	50.7%	50.7%	54.6%

Tabel 5.8 Hasil evaluasi deteksi lokasi

Nama	**Akurasi**	**Precision**	**Recall**	**Fmeasure**
NER	**91%**	**38.8%**	**53.8%**	**45%**
POSTAG	94%	-	0%	-
RBM	69%	10%	87.5%	19.1%
Hasil Akhir	68%	13.8%	90%	24%

```
λ python -W ignore python-kfold.py
Jaringan1 Processed
Acc: 0.710477 , Prec: 0.705030, Rec: 0.503899, FMeas : 0.423383

Produk1 Processed
Acc: 0.694202 , Prec: 0.397141, Rec: 0.500187, FMeas : 0.410392
```

Gambar 5.8 Contoh screenshoot hasil matrix evaluasi Sklearn

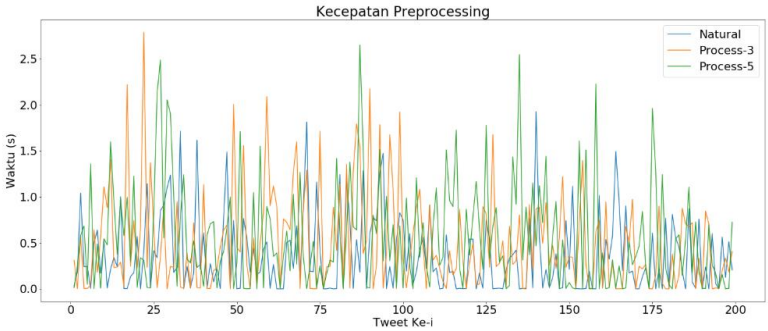
```
JARINGAN
 AVG ACC : 0.4910446857441559 AVG PREC : 0.29084006203482615
 AVG RECALL : 0.49210866779323437 AVG fmeas :0.36494266191568187

Produk
 AVG ACC : 0.5120603157748955 AVG PREC : 0.330891603299991
 AVG RECALL : 0.501193025825993 AVG fmeas :0.398220151422762
```

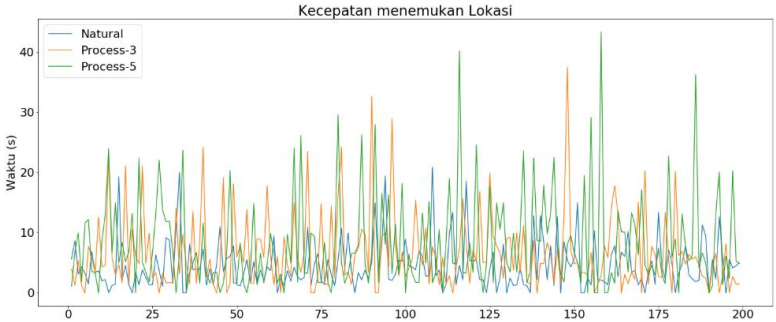
Gambar 5.9 Contoh screenshoot hasil matrix evaluasi Apache Flink

5.3.4. Skenario Pengujian 4

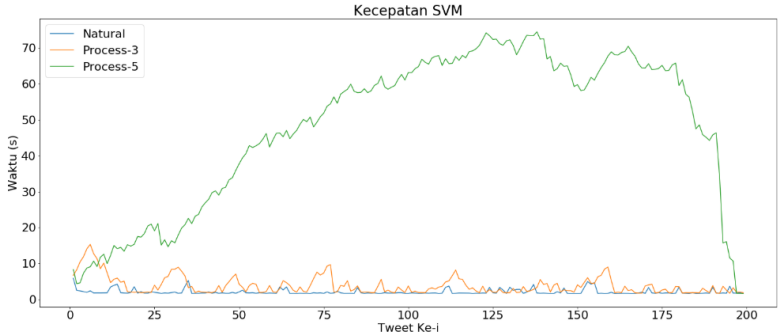
Pada skenario ini dilakukan evaluasi kecepatan sistem dalam memroses data. Evaluasi model SVM dilakukan dengan cara mengukur kecepatan *preprocesing*, deteksi lokasi, klasifikasi, dan sistem secara keseluruhan. Pengukuran dimulai sesaat *tweet* diterima oleh sistem untuk diteruskan kedalam sistem. Kecepatan sistem diukur dan dibandingkan skema *multiprocessing* pada tahap *preprocessing* dan penyesuaian data dengan Python karena semakin banyak *preprocessor* maka semakin banyak data pula yang diterima Apache Flink untuk diklasifikasikan pada saat yang bersamaan. Sehingga penulis dapat mengetahui performa Apache Flink pada pengaturan *default* terhadap jumlah data yang masuk pada tiap detiknya. Pengujian dilakukan dengan 199 *tweet* yang diambil secara acak dari datalatih. Dari hasil uji coba didapatkan lama waktu *preprocessing* 0,1 – 2,8 detik. Sedangkan waktu terlama yang dibutuhkan untuk mendeteksi lokasi adalah 42 detik. Adapun total waktu yang dibutuhkan sistem untuk menjalankan pengujian adalah 380 – 1000 detik.



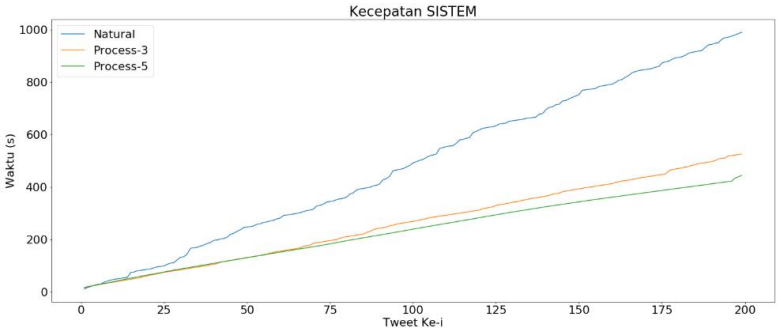
Gambar 5.10 Hasil pengukuran kecepatan preprocessing



Gambar 5.11 Hasil pengukuran kecepatan deteksi lokasi



Gambar 5.12 Hasil pengukuran kecepatan klasifikasi Apache Flink



Gambar 5.13 Hasil pengukuran kecepatan sistem

5.4. Pembahasan

Dari hasil pengujian yang dapat dilihat pada Tabel 5.6 dan Tabel 5.7, meskipun Apache Flink unggul pada dataset tanpa *preprocessing* dan Sklearn lebih baik pada dataset dengan *preprocessing*, tidak terdapat perbedaan yang signifikan antara Apache Flink dan Sklearn pada Data Latih 2. Sedangkan pada Data Latih 1, sklearn memiliki hasil yang lebih baik dibandingkan Apache Flink karena model klasifikasi Apache Flink tidak hanya dipengaruhi oleh jumlah data, tapi juga keseimbangan jumlahnya. Hal ini dapat dilihat dengan akurasi dan precision Apache Flink yang cukup rendah apabila dibandingkan dengan Sklearn pada Data Latih 1. Selain itu, dari kedua hasil tersebut, dapat dilihat bahwa Apache Flink dapat bekerja lebih baik dengan melakukan klasifikasi tanpa preprocessing sedangkan Sklearn memiliki efek yang sebaliknya.

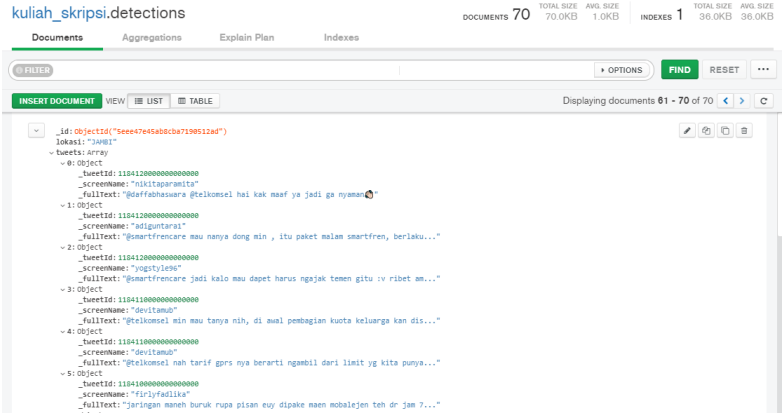
Dari model terbaik pada Apache Flink, didapatkan hasil akurasi akhir sebesar 43% pada pengujian 200 data diluar data latih. Dari hasil tersebut dapat dilihat terdapat penurunan sebesar 7% dibandingkan pengujian cross validation yang telah dilakukan. Hal ini disebabkan terdapat beberapa kata atau tweet pada data latih yang cukup dominan sehingga menyebabkan overfitting. Meskipun cukup rendah, hasil klasifikasi yang didapatkan menggunakan Apache Flink dengan model pembobotan Doc2Vec lebih unggul apabila dibandingkan dengan model pembobotan TF-

IDF. Kemungkinan, hal ini dikarenakan terkadang pembobotan teks dengan model TF-IDF memiliki banyak *sparse matrix* atau nilai yang bernilai kosong sehingga hasil klasifikasi rendah. Adapun hasil klasifikasi model SVM Jaringan dan Produk menggunakan pembobotan TF-IDF dapat dilihat pada Tabel 5.9. Tabel 5.9 Hasil evaluasi terbaik masing-masing model klasifikasi menggunakan pembobotan TF-IDF

Model	**Data Latih**	***Cross Validation***			
		Acc	**Prec**	**Rec**	**FMea**
Jaringan	1	26%	26.5%	87%	40%
	2	21%	24%	31%	27%
Produk	1	24%	24%	71%	35.9%
	2	25%	15%	12.5%	13.5%

Sedangkan pengujian deteksi lokasi terhadap 200 tweet yang dirangkum pada Tabel 5.8, dapat dilihat bahwa akurasi deteksi lokasi sistem cukup rendah yaitu 68%. Hal ini dikarenakan hasil metode *Rule-based Matching* sangat memengaruhi hasil pada saat melakukan voting hasil ekstraksi lokasi masing-masing metode. Dalam implementasinya, aturan RBM disesuaikan dengan hasil dataset yaitu mengganti POS PROPN dengan NOUN untuk mendapatkan hasil yang lebih baik. Meskipun demikian, hampir semua data *tweet* yang diawali POS ADP seperti ”di dalam”, ”di kerjakan”, ”dari pagi”, dan lain sebagainya dianggap memiliki lokasi padahal seharusnya tidak. Sedangkan metode POSTAG gagal menemukan lokasi pada data uji. Hasil POSTAG tidak memiliki nilai TP dan FP sehingga nilai *precision* dan *fmeasure*-nya tidak terdefinisi karena semua *tweet* yang seharusnya memiliki lokasi tidak dapat terdeteksi sama sekali. Metode ekstraksi lokasi terbaik pada pengujian ini adalah NER.

Dari 5580 *tweet* yang telah diujikan, sistem berhasil menemukan 70 gangguan seperti yang dapat dilihat pada Gambar 5.14. Sistem hanya mampu mendeteksi 70 gangguan karena dari 5580 data, hanya terdapat 150 lokasi yang tercantum setidaknya 5 kali pada tweet. Hal ini disebabkan sedikitnya jumlah tweet keluhan yang mengandung kata-kata lokasi.



Gambar 5.14 Screenshoot deteksi lokasi

Dari Gambar 5.13, dapat dilihat bahwa pemrosesan penyesuaian data dengan satu *processor* memiliki kecepatan yang lebih rendah dibandingkan tiga dan 5 *processor*. Hal ini karena setiap *tweet* yang telah diterima oleh Tweepy menunggu *tweet* sebelumnya selesai di-*preprocess* dan dikirim ke Apache Flink. Dengan kata lain terjadi *bottleneck* pada tahap *preprocessing* meskipun perbedaan waktu *preprocessing* dan penyesuaian data masing-masing tweet tidak terlalu jauh sekitar 0.5 detik hingga 2.5 detik seperti yang ditunjukkan pada Gambar 5.10 dan Gambar 5.11. Sedangkan dengan *multiprocessing*, data yang masuk akan diproses secara paralel sesuai dengan jumlah *processor* sehingga menghasilkan waktu *preprocessing* yang lebih cepat.

Dari Gambar 5.12, dapat dilihat bahwa semakin banyak jumlah data yang akan diklasifikasikan, semakin tinggi juga waktu yang diperlukan untuk mengklasifikasikan masing-masing *tweet*. Hal ini karena jumlah tweet yang dapat diklasifikasikan Apache Flink per detik tidak sebanding dengan jumlah data tweet yang masuk. Meskipun hasil pemrosesan sistem dengan jumlah 5 *preprocessor* pada komputer berspesifikasi Intel Core i7-4720HQ lebih baik dibandingkan jumlah *preprocessor* lainnya, dapat dilihat Apache Flink kewalahan ketika jumlah data yang masuk terlampau banyak dalam waktu yang bersamaan. Hal ini karena, *bottleneck*

sistem yang terjadi pada *preprocessing* telah tidak ada. Data yang masuk bisa langsung di-*preprocess* dan dikirimkan ke Apache Flink secara paralel sehingga menyebabkan ketidakseimbangan antara jumlah data yang masuk dengan jumlah data yang dapat diklasifikasikan.

Volume *datastream* yang masuk dapat diproses dengan baik dan optimal oleh Apache Flink adalah saat *preprocessor* berjumlah 3. Apache Flink dapat mengklasifikasikan masing-masing tweet dengan jumlah waktu yang tidak jauh berbeda dengan 1 *preprocessor* meskipun jumlah tweet yang diterima dalam waktu yang bersamaan lebih tinggi.

[Halaman ini sengaja dikosongkan]

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini akan diberikan kesimpulan yang diperoleh selama pengerjaan tugas akhir dan saran mengenai pengembangan yang dapat dilakukan terhadap Tugas Akhir ini di masa yang akan datang.

6.1. Kesimpulan

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

1. Keluhan pelanggan penyedia layanan telekomunikasi pada Twitter dapat diklasifikasikan menjadi 2 kelas jaringan atau produk dengan merepresentasikannya kedalam vektor menggunakan algoritma pembobotan seperti Doc2Vec. Setelah itu, representasi teks *tweet* diklasifikasikan dengan algoritma SVM yang telah dilatih menggunakan dataset yang dibuat secara manual dengan melabeli masing-masing tweet kedalam kelas Jaringan, Produk, atau Bukan Keduanya.
2. Sistem deteksi gangguan secara *realtime* dapat dirancang dengan mengirimkan *datastream* tweet ke Apache Flink untuk diklasifikasikan. *Datastream* didapatkan dengan berlangganan atau mendengarkan *stream* API Twitter menggunakan Tweepy. Data yang akan dikirim diproses terlebih dahulu untuk mendapatkan informasi penting didalamnya. Informasi penting tersebut seperti lokasi, representasi teks *tweet*, entitas NER, dan lain sebagainya.
3. Dengan spesifikasi i7-4720HQ, Apache Flink dapat mengklasifikasikan *datastream* 3 tweet per 2,5 - 1 detik pada waktu yang bersamaan. Seiring bertambahnya jumlah tweet yang datang pada waktu yang sama, Apache Flink mengalami pertambahan *delay* yang cukup tinggi.

4. Pada 5580 tweet, total gangguan yang dapat dideteksi adalah 70 kasus. Sistem dapat mendeteksi gangguan dengan melihat jumlah komplain terkait jaringan dalam 10 menit secara *asynchronous*. Sistem akan mengirimkan setiap *tweet* dengan kelas jaringan untuk dipetakan dan disimpan berdasarkan lokasi terjadinya. Apabila sebuah *tweet* telah disimpan melebihi waktu yang ditentukan dan belum memenuhi 5 keluhan jaringan maka sistem menyimpulkan tidak ada gangguan pada lokasi tersebut.
5. Lokasi gangguan atau komplain keluhan pengguna dapat dideteksi dengan menggunakan metode NER, POSTAG, dan RBM. Dengan menggunakan metode tersebut didapatkan beberapa entitas, salah satunya adalah LOKASI. Selanjutnya, entitas lokasi tersbut dicocokkan dengan korpus data menggunakan metode *string matching* dengan algoritma Levenshtein untuk mengurangi perbedaan penulisan teks tempat akibat bentuk kata *tweet* yang kurang formal.

6.2. Saran

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

1. Menambah data latih model SVM untuk kelas Jaringan dan Produk agar mamberikan nilai akurasi dan presisi yang lebih baik.
2. Mengganti *string matching* dengan API Google Maps untuk mendapatkan hasil yang lebih akurat dan kebebasan dalam memilih cakupan gangguan seperti provinsi, kota, kecamatan, dan seterusnya. Namun perlu pertimbangan dan perhitungan matang terkait jumlah *tweet* yang akan dicari dengan Maps API karena biaya layanan yang cukup mahal.

3. Memperbarui versi Apache Flink serta menyusun arsitektur Apache Flink yang terdistribusi karena pengujian yang dilakukan menggunakan versi *standalone*. Selain itu, mengganti pengaturan *default* Apache Flink seperti memperbesar jumlah virtual memori, *jobmanager*, dan *worker*.
4. Data latih yang digunakan untuk pembuatan model POSTAG disesuaikan dengan topik permasalahan agar model POSTAG dapat memberikan hasil yang lebih akurat dalam mendeteksi kelas kata sehingga mendapatkan hasil lokasi yang lebih baik.
5. Mengoptimalkan entitas-entitas pada NER selain lokasi seperti service, produk, waktu, dan keterangan untuk mendeteksi dan klasifikasi *tweet*.

[Halaman ini sengaja dikosongkan]

DAFTAR PUSTAKA

[1] We Are Social Inc, "Digital in 2019," [Online]. Available: https://wearesocial.com/global-digital-report-2019. [Accessed 19 June 2020].

[2] R. D. L. P, C. Fatichah and D. Purwitasari, "Deteksi Gempa Berdasarkan Data Twitter Menggunakan Decision Tree, Random Forest, dan SVM," *Jurnal Teknik ITS*, vol. 6, 2017.

[3] I. Mabruri, A. Z. Arifin and C. Fatichah, Sistem Deteksi Kecelakaan Melalui Pemanfaatan Sosial Media Twitter Berbasis Deep Learning, Surabaya: ITS, 2019.

[4] A. Rozen, "Tweeting Made Easier," 7 November 2017. [Online]. Available: https://blog.twitter.com/official/en_us/topics/product/2017/tweetingmadeeasier.html. [Accessed 23 April 2020].

[5] Indosat, "LEADing Through Trust:Laporan Tahunan Indosat 2018," Indosat, 2018.

[6] Telkomsel, "YOUR GATEWAY TO THE DIGITAL WORLD: Laporan Tahunan Telkomsel 2018," Telkomsel, 2018.

[7] Smartfren, "Go Unlimited:Laporan Tahunan Smartfren 2018," Smartfren, 2018.

[8] Y. A. N. N. W. Agus Mulyanto, "Penyelesaian Kata Ambigu Pada Proses POS Tagging Menggunakan Algoritma Hidden Markov Model(HMM)," *Prosiding Seminar Nasional Metode Kuantitatif*, 2017.

[9] L. S. A. A. M. Alireza Mansouri, "Named Entity Recognition Approach," *International Journal of Computer Science and Network Security*, vol. 8, 2008.

[10] A. KS, "Rule-Based Matching with spaCy," 19 May 2019. [Online]. Available:

https://medium.com/@ashiqgiga07/rule-based-matching-with-spacy-295b76ca2b68. [Accessed 23 April 2020].

[11] T. M. Quoc Le, "Distributed Representations of Sentences and Documents," 2014.

[12] S. L. S. Carl Kingsford, "What are decision trees?," *Nature Biotechnology*, vol. 26, pp. 1011-1013, 2008.

[13] "Tweepy," Tweepy, [Online]. Available: https://github.com/tweepy/tweepy. [Accessed 23 April 2020].

[14] H. A. Robbani, "PySastrawi," [Online]. Available: https://github.com/har07/PySastrawi. [Accessed 23 April 2020].

[15] "Prodigy - An Annotation tool for AI, Machine Learning & NLP," Prodigy, [Online]. Available: https://prodi.gy/. [Accessed 23 April 2020].

[16] "spaCy 101: Everything you need to know," spaCy, [Online]. Available: https://spacy.io/usage/spacy-101. [Accessed 23 April 2020].

[17] J. Keller, "Text Matching with Fuzzy Wuzzy," 7 May 2019. [Online]. Available: https://medium.com/@jmcneilkeller/text-matching-with-fuzzywuzzy-6600eb32c530. [Accessed 23 April 2020].

[18] R. Rehurek, "Gensim - Topic Modelling for Human," 1 November 2019. [Online]. Available: https://radimrehurek.com/gensim/. [Accessed 23 April 2020].

[19] "sci-kit : machine learning in python," [Online]. Available: https://scikit-learn.org/stable/. [Accessed 23 April 2020].

[20] [Online]. Available: https://www.scipy.org/. [Accessed 23 April 2020].

[21] E. Burnett, "twitter-text-python," [Online]. Available: https://github.com/edmondburnett/twitter-text-python. [Accessed 23 April 2020].

[22] "Flask," 4 April 2020. [Online]. Available: https://pypi.org/project/Flask/. [Accessed 23 April 2020].

[23] MongoDB Inc, "What Is MongoDB?," [Online]. Available: https://www.mongodb.com/what-is-mongodb. [Accessed 23 April 2020].

[24] MongoDB Inc, "MongoDB Compass," [Online]. Available: https://www.mongodb.com/products/compass. [Accessed 23 April 2020].

[25] MongoDB Inc, "PyMongo Documentation," [Online]. Available: https://pymongo.readthedocs.io/en/stable/index.html. [Accessed 23 April 2020].

[26] Apache, "Apache Kafka - Introduction," [Online]. Available: https://kafka.apache.org/intro. [Accessed 23 April 2020].

[27] "PyKafka," [Online]. Available: https://pykafka.readthedocs.io/en/latest/. [Accessed 23 April 2020].

[28] "What is Apache Flink? - Architecture," Apache, [Online]. Available: https://flink.apache.org/flink-architecture.html. [Accessed 23 April 2020].

[29] S. Narkhede, "Understanding Confusion Matrix," 9 May 2018. [Online]. Available: https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62. [Accessed 23 April 2020].

[30] A. D. F. R. A. L. Rulli Manurung, "idn-tagged-corpus," 9 September 2016. [Online]. Available: https://github.com/famrashel/idn-tagged-corpus. [Accessed 23 April 2020].

[31] W. Cui, P. Wang, Y. Du, X. Chen, D. Guo, J. Li and Y. Zhou, "An algorithm for event detection based on social media data," *Neurocomputing*, vol. 254, pp. 53-58, 2017.

[32] A. Z. Arifin, Interviewee, *Information Retrieval Class.* [Interview]. September 2019.

[33] C. DSN, "Wilayah," [Online]. Available: https://github.com/cahyadsn/wilayah. [Accessed 23 April 2020].

[34] M. Murugavel, "Spacy Ner Annotator," 22 April 2019. [Online]. Available: https://github.com/ManivannanMurugavel/spacy-ner-annotator. [Accessed 23 April 2020].

[35] Twitter Inc, "Tweet Objects," [Online]. Available: https://developer.twitter.com/en/docs/tweets/data-dictionary/overview/tweet-object. [Accessed 29 April 2020].

[36] Inet, "UI (User Interface)," Informasi Internet, 23 November 2016. [Online]. Available: http://www.informasi-internet.com/2016/11/ui-user-interface.html. [Accessed November 2018].

[37] "Universal POS tags," [Online]. Available: https://universaldependencies.org/docs/u/pos/index.html. [Accessed 23 April 2020].

[38] Apache, "What is Apache Flink? - Architecture," [Online]. Available: https://flink.apache.org/flink-architecture.html. [Accessed 23 April 2020].

LAMPIRAN

Lampiran A: Implementasi Preprocessor

```
14. import re
15. from ttp import ttp
16. from Sastrawi.Stemmer.StemmerFactory import Stemmer
    Factory
17. from json import dumps
18. import json
19.
20. class Preprocessor:
21.   def __init__(self):
22.     self.__slangs = self.loadSlangFromJson()
23.     self.__symbolre = re.compile("[ ](?=[ ])|[^-
    _,A-Za-z0-9 ]+")
24.     self.__ttp_parser = ttp.Parser()
25.     self.__stemmer = StemmerFactory().create_stemme
    r()
26.
27.   def preprocess(self,tweet, mentions = None):
28.     holder = tweet
29.     holder = self.remove_link_hastag_mentions(holde
    r.replace("\n","").lower(),mentions)
30.
31.     holder = self.remove_symbol_and_number(holder)

32.
33.     holder = self.slang_change(holder)
34.
35.     holder = self.stemming(holder)
36.
37.     return holder
38.
39.
40.   def stemming(self, tweet):
41.     return self.__stemmer.stem(tweet)
42.
43.   def loadSlangFromJson(self):
44.     datas = None
45.     with open('DATA\slang.json') as json_file:
46.       datas = json.load(json_file)
47.     return datas
```

```
48.
49.   def slang_change(self,tweet):
50.     holder = tweet
51.     for replace_with in self.__slangs:
52.       for slang in self.__slangs[replace_with]:
53.         if(holder.find(" "+slang+" ")>0):
54.           holder = holder.replace(" "+slang+" "," "
    +replace_with+" ")
55.           break
56.     return holder
57.
58.   def remove_symbol_and_number(self,tweet):
59.     return self.__symbolre.sub("",tweet)
60.
61.   def remove_link_hastag_mentions(self,tweet,mentio
    n = None):
62.     holder = tweet
63.     if(mention != None):
64.         holder = self.remove_mentions_provided(hold
    er,mention)
65.     entity = self.__ttp_parser.parse(tweet)
66.
67.     for mention in entity.users:
68.         holder = holder.replace("@"+mention.lower()
    ,"")
69.
70.     for url in entity.urls:
71.         holder = holder.replace(url.lower(),"")
72.
73.     for tag in entity.tags:
74.         holder = holder.replace("#"+tag.lower(),"")
75.
76.     return holder
77.
78.   def remove_mentions_provided(self,tweet, mentions
    ):
79.     holder = tweet
80.     for mention in mentions:
81.         holder = holder.replace(mention.lower(),"")

82.     return holder
83.
84. if __name__ == "__main__":
```

```
85.   preprocess = Preprocessor()
86.   print(preprocess.preprocess("@IndosatCare Ini sud
    ah registrasi tapi notif sperti ini,. Tp kenapa pul
    sa di sedot semua ya.. https://t.co/CUrKHGFgjO"))
```

Lampiran B: Implementasi Processor

```
42. import spacy
43. import pandas as pd
44. import numpy as np
45. from gazetter import Gazetter
46. from spacy.matcher import Matcher
47. from sklearn.feature_extraction.text import TfidfVe
    ctorizer
48. from scipy import sparse
49. from gensim.sklearn_api import D2VTransformer
50. from preprocessing import Preprocessor
51. import pickle
52. import _thread
53. class Processor:
54.   def __init__(self, vectorizer = True, vectorizer_
    type = "doc2vec"):
55.     self.__mNer = self.loadNERModel()
56.     self.__mPostag = self.loadPosTagModel()
57.     self.__gazetter = Gazetter(self.loadPlaces())
58.     self.__matcher = self.loadMatcher()
59.     self.__pcs = Preprocessor()
60.     self.__vector_type = vectorizer_type
61.     if(vectorizer):
62.       if(vectorizer_type == "doc2vec"):
63.         print("Creating doc2vec modal...")
64.         self.__vectorizer = self.createDoc2Vectoriz
    er()
65.     else:
66.       self.__vectorizer = self.createVectorizer()
67.
68.   def getVector(self,tweet):
69.     if(self.__vector_type=="doc2vec"):
70.       d2v_vectors = [self.__vectorizer.transform([t
    weet])[0]]
71.       return sparse.csr_matrix(d2v_vectors)
```

```
72.     else:
73.       return self.__vectorizer.transform([tweet])
74.
75.   def createDoc2Vectorizer(self):
76.     tweets = pickle.load(open("FIXED-
    DATA\processed-tweets-extended-for-vector-abc-
    balanced.pkl","rb"))
77.     gensim_doc2vec = D2VTransformer(size=100)
78.     processed_tweets = [self.__pcs.preprocess(x) fo
    r x in tweets]
79.     vector = gensim_doc2vec.fit(processed_tweets)
80.     print("Vector model created!")
81.     return vector
82.
83.   def createVectorizer(self):
84.     tweets = pickle.load(open("FIXED-
    DATA\processed-tweets-extended-for-vector-abc-
    balanced.pkl","rb"))
85.     vectorizer = TfidfVectorizer()
86.     vector = vectorizer.fit(tweets)
87.     print("Vector model created!")
88.     return vector
89.
90. def loadMatcher(self):
91.     matcher = Matcher(self.__mPostag.vocab)
92.     pattern = [{"POS": "ADP", "OP": "+"},{"LOWER":
    "daerah", "OP": "?"},{"LOWER": "desa", "OP": "?"},{
    "LOWER": "kota", "OP": "?"},{"LOWER": "kecamatan",
    "OP": "?"},{"LOWER": "kabupaten", "OP": "?"}, {"POS
    ": "NOUN", "OP": "+"}]
93.     pattern1 = [{"LOWER": "daerah", "OP": "+"}, {"P
    OS": "NOUN", "OP": "+"}]
94.     pattern2 = [{"LOWER": "desa", "OP": "+"}, {"POS
    ": "NOUN", "OP": "+"}]
95.     pattern3 = [{"LOWER": "kota", "OP": "+"}, {"POS
    ": "NOUN", "OP": "+"}]
96.     pattern4 = [{"LOWER": "ds", "OP": "+"},{"POS":"
    PUNCT","OP":"?"},{"POS": "NOUN", "OP": "+"}]
97.     pattern5 = [{"LOWER": "kecamatan", "OP": "+"},
    {"POS": "NOUN", "OP": "+"}]
98.     pattern6 = [{"LOWER": "kec", "OP": "+"},{"POS":
    "PUNCT","OP":"?"}, {"POS": "NOUN", "OP": "+"}]
```

```
99.     pattern7 = [{"LOWER": "kabupaten", "OP": "+"},
    {"POS": "NOUN", "OP": "+"}]
100.      pattern8 = [{"LOWER": "kab", "OP": "+"},{"POS
    ":"PUNCT","OP":"?"}, {"POS": "NOUN", "OP": "+"}]
101.      matcher.add("LOKASI", None,pattern,pattern2,p
    attern3,pattern4,pattern5,pattern6,pattern7,pattern
    8)
102.      return matcher
103.
104.    def loadPlaces(self):
105.      places = []
106.      with open("DATA/lokasi-from-
    details.txt","r") as f:
107.        datas = f.readlines()
108.        for data in datas:
109.            places.append(data)
110.      return places
111.
112.    def getAddress(self,tweet):
113.      _ner = self.getLocationBasedOnNER(tweet).repl
    ace("\n","")
114.      _postag = self.getLocationBasedOnPosTag(tweet
    ).replace("\n","")
115.      _matcher = self.getLocationBasedOnMatcher(twe
    et).replace("\n","")
116.      print("Processing Address")
117.      print(">Ner")
118.      print(_ner)
119.      print(">Postag")
120.      print(_postag)
121.      print(">Matcher")
122.      print(_matcher)
123.
124.      holder = {}
125.      if(len(_ner)!=0):
126.        holder[_ner] = 1
127.
128.      if(_postag in holder.keys()):
129.        holder[_postag] = holder[_postag] + 1
130.      elif(len(_postag)!=0):
131.        holder[_postag] = 1
132.
133.      if(_matcher in holder.keys()):
```

```
134.        holder[_matcher] = holder[_matcher] + 1
135.      elif(len(_matcher)!=0):
136.        holder[_matcher] = 1
137.      print(holder)
138.      ma = -1
139.      ke = ""
140.      for key in holder.keys():
141.        print("key %s : %d"%(key,holder[key]))
142.        if(holder[key] > ma):
143.          ma = holder[key]
144.          ke = key
145.          print(ma)
146.          print(ke)
147.
148.      # if the all method have diff value then use
    the NER since it has better accuracy
149.      if(ma == 1 and len(_ner)!=0):
150.        return _ner
151.
152.      return ke
153.
154.    def getLocationBasedOnNER(self,tweet):
155.      entities = self.getNerEntities(tweet)
156.      locations = {
157.        "desa" : [],
158.        "kecamatan" : [],
159.        "kabupaten" : [],
160.        "kota" : [],
161.        "others" : [],
162.      }
163.      for ent in entities.ents:
164.        if(ent.label_ == "LOKASI"):
165.          holder = ent.text
166.          if(holder.lower().find("desa") >= 0 or ho
    lder.lower().find("ds.") >= 0):
167.            locations['desa'].append(ent.text)
168.
169.          elif(holder.lower().find("kecamatan") >=
    0 or holder.lower().find("kec.") >= 0):
170.            locations['kecamatan'].append(ent.text)

171.
```

```
172.          elif(holder.lower().find("kabupaten") >=
    0 or holder.lower().find("kab.") >= 0):
173.            locations['kabupaten'].append(ent.text)

174.
175.          elif(holder.lower().find("kota") >= 0):
176.            locations['kota'].append(ent.text)
177.
178.          else:
179.            locations['others'].append(ent.text)
180.
181.      loc_result = ("",0)
182.      for loc_input in locations['desa']:
183.        curr_val = self.__gazetter.find_place(loc_i
    nput)
184.        if(curr_val[1] > loc_result[1]):
185.          loc_result = curr_val
186.
187.      if(loc_result[0] != ""):
188.          return loc_result[0]
189.
190.      for loc_input in locations['kecamatan']:
191.        curr_val = self.__gazetter.find_place(loc_i
    nput)
192.        if(curr_val[1] > loc_result[1]):
193.          loc_result = curr_val
194.
195.      if(loc_result[0] != ""):
196.        return loc_result[0]
197.
198.      for loc_input in locations['kabupaten']:
199.        curr_val = self.__gazetter.find_place(loc_i
    nput)
200.        if(curr_val[1] > loc_result[1]):
201.          loc_result = curr_val
202.
203.      if(loc_result[0] != ""):
204.          return loc_result[0]
205.
206.      for loc_input in locations['kota']:
207.        curr_val = self.__gazetter.find_place(loc_i
    nput)
208.        if(curr_val[1] > loc_result[1]):
```

```
209.          loc_result = curr_val
210.
211.      if(loc_result[0] != ""):
212.        return loc_result[0]
213.
214.      for loc_input in locations['others']:
215.        curr_val = self.__gazetter.find_place(loc_i
    nput)
216.        if(curr_val[1] > loc_result[1]):
217.          loc_result = curr_val
218.
219.      return loc_result[0]
220.
221.    def getLocationBasedOnPosTag(self,tweet):
222.        holder = self.getPosTagEntitiesSplit(tweet)
223.
224.        return self.__gazetter.get_location(holder[
    0],holder[1])
225.
226.    def getLocationBasedOnMatcher(self,tweet):
227.      locations = {
228.          "desa" : [],
229.          "kecamatan" : [],
230.          "kabupaten" : [],
231.          "kota" : [],
232.          "others" : [],
233.      }
234.      array_of_text = self.getTokenMatchEntities(tw
    eet)
235.      if(len(array_of_text)==0):
236.          return ""
237.      for ent in array_of_text:
238.        holder = ent
239.        if(holder.lower().find("desa") >= 0 or hold
    er.lower().find("ds.") >= 0):
240.          locations['desa'].append(holder)
241.
242.        elif(holder.lower().find("kecamatan") >= 0
    or holder.lower().find("kec.") >= 0):
243.          locations['kecamatan'].append(holder)
244.
245.        elif(holder.lower().find("kabupaten") >= 0
    or holder.lower().find("kab.") >= 0):
```

```
246.            locations['kabupaten'].append(holder)
247.
248.          elif(holder.lower().find("kota") >= 0):
249.            locations['kota'].append(holder)
250.
251.          else:
252.            locations['others'].append(holder)
253.
254.        loc_result = ("",0)
255.        for loc_input in locations['desa']:
256.          curr_val = self.__gazetter.find_place(loc_i
    nput)
257.          if(curr_val[1] > loc_result[1]):
258.            loc_result = curr_val
259.
260.        if(loc_result[0] != ""):
261.          return loc_result[0]
262.
263.        for loc_input in locations['kecamatan']:
264.          curr_val = self.__gazetter.find_place(loc_i
    nput)
265.          if(curr_val[1] > loc_result[1]):
266.            loc_result = curr_val
267.
268.        if(loc_result[0] != ""):
269.          return loc_result[0]
270.
271.        for loc_input in locations['kabupaten']:
272.          curr_val = self.__gazetter.find_place(loc_i
    nput)
273.          if(curr_val[1] > loc_result[1]):
274.            loc_result = curr_val
275.
276.        if(loc_result[0] != ""):
277.          return loc_result[0]
278.
279.        for loc_input in locations['kota']:
280.          curr_val = self.__gazetter.find_place(loc_i
    nput)
281.          if(curr_val[1] > loc_result[1]):
282.            loc_result = curr_val
283.
284.        if(loc_result[0] != ""):
```

```
285.         return loc_result[0]
286.
287.       for loc_input in locations['others']:
288.         curr_val = self.__gazetter.find_place(loc_i
    nput)
289.         if(curr_val[1] > loc_result[1]):
290.           loc_result = curr_val
291.
292.       return loc_result[0]
293.
294.     def getNerEntities(self,tweet):
295.       return self.__mNer(tweet)
296.
297.     def getPosTagEntities(self,tweet):
298.       return self.__mPostag(tweet)
299.
300.     def getPosTagEntitiesSplit(self,tweet):
301.       words = []
302.       tags = []
303.       result =  self.__mPostag(tweet)
304.       for token in result:
305.         words.append(token.text)
306.         tags.append(token.tag_)
307.       return (words, tags)
308.
309.     def getTokenMatchEntities(self,tweet):
310.       doc = self.__mPostag(tweet)
311.       token_match = []
312.       matches = self.__matcher(doc)
313.       for match_id, start, end in matches:
314.         string_id = self.__mPostag.vocab.strings[ma
    tch_id]  # Get string representation
315.         span = doc[start:end]  # The matched span
316.         token_match.append(span.text)
317.       return token_match
318.
319.     def loadPosTagModel(self):
320.       print("Postag model created!")
321.       return spacy.load("FIXED-MODEL/POSTAG/pos-
    tag")
322.
323.     def loadNERModel(self):
324.       print("Ner model created!")
```

```
325.      return spacy.load('FIXED-MODEL/NER/NER90')
326.
327.    def getNerForWeb(self,tweet):
328.      processed = tweet
329.      doc = self.getNerEntities(tweet)
330.      for index,ent in reversed(list(enumerate(doc.
    ents))):
331.        before = processed[:ent.start_char]
332.        after = processed[ent.end_char:]
333.        if(ent.label_ == "SERVICE"):
334.          label = "<span class='label label-
    warning'>%s                                [SERVICE]
    </span>"%(processed[ent.start_char:ent.end_char])
335.        elif(ent.label_ == "LOKASI"):
336.          label = "<span class='label label-
    primary'>%s                                 [LOKASI]
    </span>"%(processed[ent.start_char:ent.end_char])
337.        elif(ent.label_ == "WAKTU"):
338.          label = "<span class='label label-
    plain'>%s                                    [WAKTU]
    </span>"%(processed[ent.start_char:ent.end_char])
339.        elif(ent.label_ == "PRODUK"):
340.          label = "<span class='label label-
    info'>%s                                    [PRODUK]
    </span>"%(processed[ent.start_char:ent.end_char])
341.        elif(ent.label_ == "KETERANGAN"):
342.          label = "<span class='label label-
    plain'>%s                              [KETERANGAN]
    </span>"%(processed[ent.start_char:ent.end_char])
343.        processed = before + label + after
344.      return processed
```

Lampiran C: Implementasi Klasifikasi: Objek Main

```
1.  package main.scala
2.
3.  import java.util.Properties
4.  import java.util
5.  import org.apache.flink.api.scala._
6.  import org.apache.flink.api.common.functions.FlatMa
    pFunction
```

```
7.  import org.apache.flink.ml.math.Vector
8.  import org.apache.flink.ml.common.LabeledVector
9.  import org.apache.flink.ml.classification.SVM
10. import org.apache.flink.ml.RichExecutionEnvironment

11. import org.apache.flink.ml.math.SparseVector
12. import org.apache.flink.api.scala.DataSet
13. import main.scala.map
14. import org.apache.flink.ml.preprocessing.Splitter
15. import org.apache.flink.ml.preprocessing.Splitter.T
    rainTestDataSet
16. import org.apache.flink.ml.regression.MultipleLinea
    rRegression
17. import org.apache.flink.shaded.jackson2.com.fasterx
    ml.jackson.databind.node.ArrayNode
18. import org.apache.flink.shaded.jackson2.com.fasterx
    ml.jackson.databind.{JsonNode, ObjectMapper}
19. import org.apache.flink.streaming.api.environment.S
    treamExecutionEnvironment
20. import org.apache.flink.streaming.connectors.kafka.
    {FlinkKafkaConsumer, FlinkKafkaProducer}
21. import org.apache.flink.streaming.util.serializatio
    n.SimpleStringSchema
22. import org.apache.flink.util.{Collector, Instantiat
    ionUtil}
23.
24. object app{
25.   def main(args: Array[String]): Unit = {
26.     val env = ExecutionEnvironment.getExecutionEnvi
    ronment
27.     val stream_env = StreamExecutionEnvironment.get
    ExecutionEnvironment
28.
29.     val properties = new Properties()
30.     println("Let's test streaming!")
31.     properties.setProperty("bootstrap.servers", "lo
    calhost:9092")
32.
33.     val myProducer = new FlinkKafkaProducer[String]
    (
34.       "localhost:9092",          // broker list
35.       "scala-streaming-dump-
    flask",                // target topic
```

```
36.         new SimpleStringSchema)
37.     val stream = stream_env
38.         .addSource(new FlinkKafkaConsumer[String]("sc
    ala-streaming-
    vectorization", new SimpleStringSchema(), propertie
    s).setStartFromLatest())
39.     stream.print()
40.     val vectors = stream.flatMap(new JsonStringAndC
    lassify).addSink(myProducer)
41.     stream_env.execute("svm-scala-streaming")
42.   }
43.
44.   private class JsonStringAndClassify extends FlatM
    apFunction[String, String] {
45.     println("Initiating parser and environment")
46.     lazy val jsonParser = new ObjectMapper()
47.     lazy val env = ExecutionEnvironment.getExecutio
    nEnvironment
48.     lazy val mapper = new map
49.     override def flatMap(value: String, out: Collec
    tor[String]): Unit = {
50.       val jsonNode : JsonNode = jsonParser.readValu
    e(value, classOf[JsonNode])
51.       val is_valid = jsonNode.has("vector")
52.       val is_text = jsonNode.has("full_text")
53.       val is_id = jsonNode.has("tweet_id")
54.       val is_user = jsonNode.has("screen_name")
55.       val is_address = jsonNode.has("_address")
56.       val is_mentions = jsonNode.has("mentions")
57.       val is_for = jsonNode.has("directed_to")
58.
59.       if(is_valid && is_text && is_id && is_user &&
     is_address && is_mentions && is_for){
60.         val v_node = jsonNode.get("vector")
61.         val v_text = jsonNode.get("full_text")
62.         val v_pretext = jsonNode.get("processed_tex
    t")
63.         val v_id = jsonNode.get("tweet_id")
64.         val v_user = jsonNode.get("screen_name")
65.         val v_address = jsonNode.get("_address")
66.         val v_mentions = jsonNode.get("mentions")
67.         val v_for = jsonNode.get("directed_to")
68.         val v_time = jsonNode.get("_time")
```

```
69.         val v_received = jsonNode.get("_received")

70.         val v_ner = jsonNode.get("_ner")
71.         val v_address_time = jsonNode.get("_address
    _time")
72.         val v_svm = jsonNode.get("_svm_time")
73.
74.         val vectors : Array[(Int,Double)] = new Arr
    ay[(Int,Double)](v_node.size())
75.         var result: String = "nihil"
76.         if(v_node.size()!=0) {
77.           for (index <- 0 until v_node.size()) {
78.               vectors.update(index, (v_node.get(ind
    ex).get(0).asInt(), v_node.get(index).get(1).asDoub
    le()))
79.           }
80.           val testingDS: DataSet[Vector] = env.from
    Elements(SparseVector.fromCOO(100, vectors))
81.           println("Starting Prediction")
82.           var collector: Collector[DataSet[(Vector,
     Double)]] = null
83.           result = mapper.flatMap2(testingDS, colle
    ctor)
84.         }else{
85.           println("Has no vector, thus continuing w
    ithout any classification...")
86.         }
87.         println("Prediction Ended. The class is " +
     result)
88.         val object_node = jsonParser.createObjectNo
    de()
89.         object_node.set("_tweetId",v_id)
90.         object_node.set("_screenName",v_user)
91.         object_node.set("_processedText",v_pretext)

92.         object_node.set("_fullText",v_text)
93.         object_node.set("_address",v_address)
94.         object_node.set("_mentions",v_mentions)
95.         object_node.set("_directed",v_for)
96.         object_node.set("_time",v_time)
97.         object_node.set("_ner",v_ner)
98.         object_node.set("_received",v_received)
99.         object_node.put("_status",0)
```

```
100.            object_node.put("_classification",result)

101.            object_node.set("_address_time",v_address
    _time)
102.            object_node.set("_svm_time",v_svm)
103.            val json_stringify : String = jsonParser.
    writeValueAsString(object_node)
104.            out.collect(json_stringify)
105.          }else{
106.            println("Ignoring stream input ! Wrong da
    ta format!")
107.          }
108.        }
109.      }
110.    }
```

Lampiran D: Implementasi Klasifikasi: Map dan Model

```
1.  package main.scala
2.
3.  import java.util
4.
5.  import org.apache.flink.api.common.functions.FlatMa
    pFunction
6.  import org.apache.flink.api.common.io.OutputFormat

7.  import org.apache.flink.api.common.state.{ListState
    , ListStateDescriptor}
8.  import org.apache.flink.api.common.typeinfo.{TypeHi
    nt, TypeInformation}
9.  import org.apache.flink.api.common.typeutils.TypeSe
    rializer
10. import org.apache.flink.api.java.io.LocalCollection
    OutputFormat
11. import org.apache.flink.api.java.tuple.Tuple
12. import org.apache.flink.api.scala._
13. import org.apache.flink.ml.common.LabeledVector
14. import org.apache.flink.ml.math.Vector
15. import org.apache.flink.util.Collector
16. import org.apache.flink.ml.RichExecutionEnvironment
```

```
17. import org.apache.flink.api.scala.DataSet
18. import org.apache.flink.ml.classification.SVM
19. import org.apache.flink.runtime.state.{FunctionInit
    ializationContext, FunctionSnapshotContext}
20. import org.apache.flink.streaming.api.checkpoint.Ch
    eckpointedFunction
21. import org.apache.flink.util.InstantiationUtil
22.
23. class map extends FlatMapFunction[DataSet[Vector],D
    ataSet[(Vector, Double)]] with CheckpointedFunction
     {
24.   @transient
25.   private var checkpointedState: ListState[Model] =
     _
26.   lazy val model : Model = new Model
27.
28.   def flatMap2(value: DataSet[Vector], out: Collect
    or[DataSet[(Vector, Double)]]): String = {
29.     println("prediction is being processed... ")
30.     var temp_jaringan : DataSet[(Vector,Double)] =
    model.predict_jaringan(value)
31.     var temp_produk : DataSet[(Vector,Double)] = mo
    del.predict_produk(value)
32.     var result_jaringan : Seq[(Vector,Double)] = te
    mp_jaringan.collect()
33.     var result_produk : Seq[(Vector,Double)] = temp
    _produk.collect()
34.     var result_akhir : String = ""
35.     if(result_jaringan(0)._2  == 1.0  && result_pro
    duk(0)._2 == 1.0 ){
36.       result_akhir = "keduanya"
37.     }else if(result_jaringan(0)._2  == 1.0  && resu
    lt_produk(0)._2 == -1.0){
38.       result_akhir = "jaringan"
39.     }else if(result_jaringan(0)._2  == -
    1.0  && result_produk(0)._2 == 1.0){
40.       result_akhir = "produk"
41.     }else{
42.       result_akhir = "nihil"
43.     }
44.     println(result_jaringan(0)._2)
45.     println(result_produk(0)._2)
46.     println(result_akhir)
```

```
47.     return result_akhir
48.   }
49.
50.   override def snapshotState(functionSnapshotContex
    t: FunctionSnapshotContext): Unit = {
51.
52.   }
53.
54.   override def initializeState(functionInitializati
    onContext: FunctionInitializationContext): Unit = {

55.     println("why error")
56.     val descriptor  = new ListStateDescriptor[Model
    ]("model", TypeInformation.of(new TypeHint[Model] {
    }))
57.     checkpointedState = functionInitializationConte
    xt.getOperatorStateStore.getListState(descriptor)
58.
59.     if(functionInitializationContext.isRestored){
60.       println("State is restored, clearing state an
    d re-initiating models!")
61.       checkpointedState.clear()
62.
63.       println("Starting to train!")
64.       model.train()
65.       println("Data trained")
66.       println("*Note : it may took some time for th
    e first prediction!")
67.       checkpointedState.add(model)
68.     }
69.   }
70.
71.   override def flatMap(value: DataSet[Vector], out:
     Collector[DataSet[(Vector, Double)]]): Unit = ???

72. }
73.
74. class Model{
75.   private var svm_jaringan :SVM = new SVM()
76.   private var svm_produk : SVM = new SVM()
77.   val env = ExecutionEnvironment.getExecutionEnviro
    nment
78.   println("Loading Dataset for training!")
```

```
79.   val trainingDS_jaringan: DataSet[LabeledVector] =
     env.readLibSVM("F:\\Dokumen\\Tugas Akhir\\skripshi
    ittt\\FIXED-MODEL\\SVM\\JARINGAN.svmlib")
80.   val trainingDS_produk: DataSet[LabeledVector] = e
    nv.readLibSVM("F:\\Dokumen\\Tugas Akhir\\skripshiit
    tt\\FIXED-MODEL\\SVM\\PRODUK.svmlib")
81.   println("Dataset Loaded")
82.   train()
83.
84.   def train(): Unit ={
85.     svm_jaringan.fit(trainingDS_jaringan)
86.     svm_produk.fit(trainingDS_produk)
87.   }
88.   def predict_produk(value : DataSet[Vector]): Data
    Set[(Vector,Double)]  ={
89.     println("Value accepted, start predicting - Pro
    duk")
90.     var temp_produk = svm_produk.predict(value)
91.     return temp_produk
92.   }
93.
94.   def predict_jaringan(value : DataSet[Vector]): Da
    taSet[(Vector,Double)]  ={
95.     println("Value accepted, start predicting - Jar
    ingan")
96.     var temp_jaringan = svm_jaringan.predict(value)

97.     return temp_jaringan
98.   }
99. }
```

Lampiran E: Implementasi Deteksi Gangguan

```
1. import threading, logging, time
2. from pykafka import KafkaClient
3. from pykafka.common import OffsetType
4. from json import dumps
5. from bson import json_util
6. import pandas as pd
7. import numpy as np
8. import json
```

```
9.  import time
10. from datetime import datetime
11. from processing import Processor
12. import pymongo
13.
14. class TroubleFilter:
15.   def __init__(self,timediff = 10.0 , tweetcount =
    5):
16.     self.telkomsel = {}
17.     self.indosat = {}
18.     self.smartfren = {}
19.     self.count = tweetcount
20.     self.tollerance = timediff
21.
22.   def checkDuplciate(self,x,y):
23.     counter = 0
24.     if(x["_tweetId"] == y["_tweetId"]):
25.       return True
26.
27.     if(SequenceMatcher(None, x['_fullText'], y["_fu
    llText"]).ratio() >= 0.75):
28.       return True
29.
30.     return False
31.
32.   def countDiffInSecond(self,diff):
33.     x = diff.days * 24 * 3600 + diff.seconds
34.     return x
35.
36.   def processData(self,objek):
37.     directeds = objek['_directed']
38.     addr = objek['_address']
39.     trouble_tweet = {
40.     "_tweetId" : objek['_tweetId'],
41.     '_screenName' : objek['_screenName'],
42.     '_fullText' : objek['_fullText']
43.     }
44.     if(addr==""):
45.       return []
46.
47.     time_ = objek['_time']
48.     time = datetime.strptime(time_,'%a %b %d %H:%M:
    %S %z %Y')
```

```
49.     result = []
50.     for directed in directeds:
51.       if ( "indosat" in directed.lower() ):
52.         if( addr in self.indosat.keys()):
53.           duplicate = False
54.           for index,data in enumerate((self.indosat
    [addr]['time'])):
55.             if(self.checkDuplciate(self.indosat[add
    r]['tweets'][index],trouble_tweet)):
56.               duplicate = True
57.               break
58.             diff = self.countDiffInSecond(time-
    data)
59.             if((diff / 60.0)>=self.tollerance):
60.               self.indosat[addr]['time'].pop(index)

61.               self.indosat[addr]['tweets'].pop(inde
    x)
62.           if(not duplicate):
63.             self.indosat[addr]['time'].append(time)

64.             self.indosat[addr]['tweets'].append(tro
    uble_tweet)
65.           if(len(self.indosat[addr]['tweets']) >= s
    elf.count):
66.             result.append((True,directed,(self.indo
    sat[addr]['tweets'])))
67.             self.indosat[addr] = {"tweets":[],"time
    ":[]}
68.         else:
69.           self.indosat[addr] = {"tweets":[],"time":
    []}
70.           self.indosat[addr]['time'].append(time)
71.           self.indosat[addr]['tweets'].append(troub
    le_tweet)
72.           result.append((False,directed))
73.       elif ( "telkomsel" in directed.lower()):
74.         if( addr in self.telkomsel.keys()):
75.           duplicate=False
76.           for index,data in enumerate((self.telkoms
    el[addr]['time'])):
77.             diff = self.countDiffInSecond(time-
    data)
```

```
78.             if(self.checkDuplciate(self.telkomsel[a
    ddr]['tweets'][index],trouble_tweet)):
79.               duplicate = True
80.               break
81.             if((diff / 60.0)>=self.tollerance):
82.               self.telkomsel[addr]['time'].pop(inde
    x)
83.               self.telkomsel[addr]['tweets'].pop(in
    dex)
84.           if(not duplicate):
85.             self.telkomsel[addr]['time'].append(tim
    e)
86.             self.telkomsel[addr]['tweets'].append(t
    rouble_tweet)
87.           if(len(self.telkomsel[addr]['tweets']) >=
     self.count):
88.             result.append((True,directed,(self.telk
    omsel[addr]['tweets'])))
89.             self.telkomsel[addr] = {"tweets":[],"ti
    me":[]}
90.         else:
91.           self.telkomsel[addr] = {"tweets":[],"time
    ":[]}
92.           self.telkomsel[addr]['time'].append(time)

93.           self.telkomsel[addr]['tweets'].append(tro
    uble_tweet)
94.           result.append((False,directed))
95.
96.       elif ( "smartfren" in directed.lower()):
97.         if( addr in self.smartfren.keys()):
98.           duplicate=False
99.           for index,data in enumerate((self.smartfr
    en[addr]['time'])):
100.              diff = self.countDiffInSecond(time-
    data)
101.              if(self.checkDuplciate(self.smartfren
    [addr]['tweets'][index],trouble_tweet)):
102.                duplicate = True
103.                break
104.              if((diff / 60.0)>=self.tollerance):
105.                self.smartfren[addr]['time'].pop(in
    dex)
```

```
106.                self.smartfren[addr]['tweets'].pop(
    index)
107.            if(not duplicate):
108.              self.smartfren[addr]['time'].append(t
    ime)
109.              self.smartfren[addr]['tweets'].append
    (trouble_tweet)
110.            if(len(self.smartfren[addr]['tweets'])
    >= self.count):
111.              result.append((True,directed,(self.sm
    artfren[addr]['tweets'])))
112.              self.telkomsel[addr] = {"tweets":[],"
    time":[]}
113.          else:
114.            self.smartfren[addr] = {"tweets":[],"ti
    me":[]}
115.            self.smartfren[addr]['time'].append(tim
    e)
116.            self.smartfren[addr]['tweets'].append(t
    rouble_tweet)
117.            result.append((False,directed))
118.      return result
119.
120.  if __name__ == "__main__":
121.    _kafkaclient = KafkaClient(hosts='127.0.0.1:909
    2')
122.    _consumer_topic = _kafkaclient.topics['scala-
    streaming-dump-flask']
123.    _producer_trouble = _kafkaclient.topics['stream
    ing-trouble']
124.    _filer = TroubleFilter()
125.    pc = Processor(vectorizer = False)
126.
127.    myclient = pymongo.MongoClient("mongodb://local
    host:27017/")
128.    mydb = myclient["kuliah_skripsi"]
129.    mytweets = mydb["tweets"]
130.    mydetection = mydb["detections"]
131.
132.    print("Now listening for incoming datas!")
133.    for data in _consumer_topic.get_simple_consumer
    (auto_offset_reset=OffsetType.LATEST,reset_offset_o
    n_start=True):
```

```
134.        print("\nNew data received")
135.        text = data.value.decode()
136.        obj = json.loads(text)
137.        obj['_elapsedtime'] = time.time() - float(obj
    ['_received'])
138.        obj['_svm_time'] = time.time() - float(obj['_
    svm_time'])
139.        mytweets.insert_one(obj)
140.        print("Processing detection")
141.        result = _filer.processData(obj)
142.        if(len(result)==0):
143.            print("No trouble because no location det
    ected")
144.        for directTo in result:
145.          if(directTo[0]):
146.            x = {}
147.            x['_address'] = obj['_address']
148.            x['_coordinates'] = pc.getCoordinates(obj
    ['_address'])
149.            x['_time'] = obj["_time"]
150.            x['_lens'] = directTo[2]
151.            print("Whops! Trouble detected!")
152.            print(x)
153.            if("indosat" in directTo[1].lower()):
154.              x['_directed'] = "indosat"
155.              mydetection.insert_one(x)
156.            elif("telkomsel" in directTo[1].lower()):

157.              x['_directed'] = "telkomsel"
158.              mydetection.insert_one(x)
159.            elif("smartfren" in directTo[1].lower()):

160.              x['_directed'] = "smartfren"
161.              mydetection.insert_one(x)
162.            with _producer_trouble.get_sync_producer(
    ) as producer:
163.              producer.produce(x)
164.          else:
165.              print("No trouble detected")
```

Lampiran F: Contoh keluaran ekstraksi fitur

Hasil ekstraksi fitur
[(0, -0.0036493001971393824), (1, -0.0018113732803612947), (2, -8.849225559970364e-05), (3, -0.002436865121126175), (4, 0.0019216787768527865), (5, -0.002038093749433756), (6, 0.0017495574429631233), (7, 0.003964245785027742), (8, 0.0028539460618048906), (9, -0.004903879016637802), (10, -0.0038759352173656225), (11, 0.004988883156329393), (12, 0.002395566087216139), (13, 0.0006833295919932425), (14, 0.0004065705870743841), (15, -0.0026863643433898687), (16, 0.00030177453299984336), (17, 0.002300648484379053), (18, -0.00175756576936692), (19, -0.0017503215931355953), (20, 0.003544115461409092), (21, -0.0038002976216375828), (22, -0.0036648553796112537), (23, -0.0032449031714349985), (24, 0.003973687067627907), (25, -0.00017703967751003802), (26, 0.0012262787204235792), (27, 0.004493934568017721), (28, -0.0034474004060029984), (29, 0.0012683351524174213), (30, 0.004111244808882475), (31, 0.002594278659671545), (32, -0.003989155404269695), (33, 0.0014738824684172869), (34, 0.002430615946650505), (35, 0.0036193770356476307), (36, 0.0037854244001209736), (37, 0.0035679952707141638), (38, 0.0012028919300064445), (39, -0.00011067337618442252), (40, -0.004818256478756666), (41, 0.004309103358536959), (42, -0.002356089884415269), (43, 2.3356216843239963e-05), (44, -0.0018712918972596526), (45, -0.004411255475133657 5), (46, -0.001204981585033238),

(47, -0.0008114629308693111), (48, -0.0020582203287631273), (49, -0.003759663784876466), (50, -0.0033857817761600018), (51, -0.004041024949401617), (52, -0.0027176225557923317), (53, -0.004254217725247145), (54, -0.0007906273240223527), (55, 0.0002179540751967579), (56, -0.0008293489227071404), (57, 0.002346411347389221), (58, 0.0010190088069066405), (59, 0.0009795983787626028), (60, 0.0028177285566926003), (61, 0.0037121272180229425), (62, -0.003308637300506234), (63, 0.001652823411859572), (64, 0.001845494844019413), (65, 0.002797933528199792), (66, -0.003285478102043271), (67, -0.0034764581359922886), (68, -0.0026192734949290752), (69, -0.0021981550380587578), (70, -0.004805809818208218), (71, -0.0003598603652790189), (72, -0.0026018787175416946), (73, 0.0014857680071145296), (74, 0.00020689406665042043), (75, 0.0030684673693031073), (76, -0.002697356976568699), (77, 0.002459984738379717), (78, 0.0005328787374310195), (79, -0.0047742221504449844), (80, 0.0005051820771768689), (81, -0.004173089750111103), (82, -0.002324427478015423), (83, -7.364134944509715e-05), (84, 0.0012350475881248713), (85, -0.002012338489294052), (86, -0.0029129241593182087), (87, 0.0008385445107705891), (88, -0.002415510592982173), (89, 0.004860735032707453), (90, 0.000723708129953593), (91, 0.0014894333435222507), (92, 0.0003879146825056523), (93, -0.003986349795013666), (94, -0.002979378215968609), (95, -0.001674968982115388), (96, 0.004546934738755226), (97, -0.0013264993904158473),

(98, -0.004322385881096125), (99, -0.0011463324772194028)]

Lampiran G: Contoh data yang disimpan pada MongoDB


```
{
  "_id":{
    "$oid":"5eab5ed8a5e7602a8ad45d8a"
  },
  "_tweetId":"1256002173139222528",
  "_screenName":"dediblessvictor",
  "_processedText":"b'@IndosatCare 08561335150 a.n DEDI VICTORIA'",
  "_fullText":"b'@IndosatCare 08561335150 a.n DEDI VICTORIA'",
  "_address":"DESA SITORANG I",
  "_mentions":[
    {
      "screen_name":"IndosatCare",
      "name":"Indosat Ooredoo Care",
      "id":{
        "$numberInt":"548904824"
      },
      "id_str":"548904824",
      "indices":[
        {
          "$numberInt":"0"
        },
        {
          "$numberInt":"12"
        }
      ]
    }
  ],
  "_directed":[
    "indosatcare"
```

```
  ],
  "_time":"Thu Apr 30 23:27:08 +0000 2020",
  "_ner":"b'@IndosatCare 08561335150 a.n <span class='label label-primary'>DEDI VICTORIA'[LOKASI] </span>",
  "_received":{
    "$numberDouble":"1588289231.7483017"
  },
  "_status":{
    "$numberInt":"0"
  },
  "_classification":"produk",
  "_coordinates":[

  ],
  "_elapsedtime":{
    "$numberDouble":"8.646024942398071"
  }
}
```

[Halaman ini sengaja dikosongkan]

BIODATA PENULIS

Akbar Noto Ponco Bimantoro lahir di Kediri, 9 April. Penulis telah menempuh jenjang pendiikan formal di SD Negeru Ngronggo VIII Kediri (2004-2010), SMP Negeri 4 Kediri (2010-2013), SMA Negeri 1 Kediri (2013-2016). Saat ini penulis sedang menempuh pendidikan Sarjana Teknik Informatika di Institut Teknologi Nopember Surabaya (ITS). Selama menempuh pendidikan di ITS, penulis aktif dalam berbagai acara kepanitiaan seperti Schematic, FTIF Journey, dan LKMM. Penulis aktif dan berpengalaman menjadi asisten dosen berbagai mata kuliah pada 2017-2019. Selain itu, penulis juga aktif bekerja sebagai *Web Developer* baik secara *self-employment* maupun bekerja paruh waktu di *softwarehouse*. Dalam menyelesaikan pendidikan sarjana, penulis mengambil rumpun mata kuliah (RMK) Manajemen Informasi (MI). Untuk komunikasi penulis dapat dihubungi melalui surel akbarnotopb@gmail.com.

[Halaman ini sengaja dikosongkan]