TUGAS AKHIR - KS141501

EKSTRAKSI INFORMASI MEDIA SOSIAL TWITTER MENGENAI KECELAKAAN LALU LINTAS MENGGUNAKAN PENDEKATAN ONTOLOGY-BASED INFORMATION EXTRACTION

INFORMATION EXTRACTION OF TWITTER SOCIAL MEDIA ABOUT TRAFFIC ACCIDENT USING ONTOLOGY-BASED INFORMATION EXTRACTION APPROACH

YASIN AWWAB
05211540000127

Dosen Pembimbing
Nur Aini Rakhmawati, S.Kom., M.Sc. Eng., Ph.D

DEPARTEMEN SISTEM INFORMASI
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2019

TUGAS AKHIR - KS141501

EKSTRAKSI INFORMASI MEDIA SOSIAL TWITTER MENGENAI KECELAKAAN LALU LINTAS MENGGUNAKAN PENDEKATAN ONTOLOGY-BASED INFORMATION EXTRACTION

YASIN AWWAB
05211540000127

Dosen Pembimbing
Nur Aini Rakhmawati, S.Kom., M.Sc. Eng., Ph.D

DEPARTEMEN SISTEM INFORMASI
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2019

Halaman ini sengaja dikosongkan.

UNDERGRADUATE THESIS - KS141501

INFORMATION EXTRACTION OF TWITTER SOCIAL MEDIA ABOUT TRAFFIC ACCIDENT USING ONTOLOGY-BASED INFORMATION EXTRACTION APPROACH

YASIN AWWAB
05211540000127

Supervisor
Nur Aini Rakhmawati, S.Kom., M.Sc. Eng., Ph.D

INFORMATION SYSTEM DEPARTMENT
Information Technology and Communication Faculty
Sepuluh Nopember Institute of Technology
Surabaya 2019

Halaman ini sengaja dikosongkan.

LEMBAR PENGESAHAN

EKSTRAKSI INFORMASI MEDIA SOSIAL TWITTER MENGENAI KECELAKAAN LALU LINTAS MENGGUNAKAN PENDEKATAN ONTOLOGY-BASED INFORMATION EXTRACTION

TUGAS AKHIR

Disusun untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Departemen Sistem Informasi
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh :

YASIN AWWAB
NRP. 05211540000127

Surabaya, Juli 2019

KEPALA
DEPARTEMEN SISTEM INFORMASI

KEMENTERIAN RISET, TEKNOLOGI DAN PENDIDIKAN TINGGI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
FAKULTAS TEKNOLOGI INFORMASI DAN KOMUNIKASI
DEPARTEMEN SISTEM INFORMASI

Mahendrawathi Er, S.T., M.Sc., Ph.D.
NIP. 19761011 200604 2 001

Halaman ini sengaja dikosongkan.

LEMBAR PERSETUJUAN

EKSTRAKSI INFORMASI MEDIA SOSIAL TWITTER MENGENAI KECELAKAAN LALU LINTAS MENGGUNAKAN ONTOLOGY PENDEKATAN ONTOLOGY-BASED INFORMATION EXTRACTION

TUGAS AKHIR

Disusun untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Departemen Sistem Informasi
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh :

YASIN AWWAB
NRP. 05211540000127

Disetujui Tim Penguji : Tanggal Ujian : Juli 2019
Periode Wisuda : September 2019

Nur Aini Rakhmawati, S.Kom., M.Sc.Eng., Ph.D **(Pembimbing I)**

Faisal Johan Atletiko, S.Kom, M.T **(Penguji I)**

Irmasari Hafidz, S.Kom, M.Sc **(Penguji II)**

Halaman ini sengaja dikosongkan.

EKSTRAKSI INFORMASI MEDIA SOSIAL TWITTER MENGENAI KECELAKAAN LALU LINTAS MENGGUNAKAN PENDEKATAN ONTOLOGY-BASED INFORMATION EXTRACTION

Nama Mahasiswa	**: Yasin Awwab**
NRP	**: 05211540000127**
Departemen	**: Sistem Informasi FTIK-ITS**
Pembimbing I	**: Nur Aini Rakhmawati, S.Kom., M.Sc. Eng., Ph.D**

ABSTRAK

Kecelakaan lalu lintas menjadi salah satu kejadian yang sering terjadi di Indonesia. Dari data yang dihimpun KORLANTAS POLRI diketahui setiap 3 bulan terdapat sekitar 25.000 kecelakaan lalu lintas. Informasi mengenai kecelakaan lalu lintas banyak dibagikan oleh pengguna media sosial termasuk Twitter. Twitter memiliki berbagai informasi mengenai kecelakaan lalu lintas. Namun, informasi yang berada di Twitter tersebut belum terdapat pengolahan dan pemetaan mengenai informasi tersebut. Oleh sebab itu, penelitian ini bertujuan untuk mengolah dan memetakan informasi mengenai kecelakaan lalu lintas yang terdapat di Twitter sehingga dapat memberikan manfaat bagi masyarakat maupun menjadi masukan dalam pengembangan terkait lalu lintas di Indonesia. Metode yang dalam penelitian ini menggunakan domain ontologi dan Named-Entity Recognition untuk proses ekstraksi data. Metode Named-Entity Recognition digunakan untuk mendapatkan kata kunci dari sebuah tweet berdasarkan kategori kelasnya seperti aktor, waktu, lokasi, dan keterangan penyebab kecelakaan tersebut. Penelitian ini menghasilkan model Named Entity Recognition yang dapat menghasilkan tingkat akurasi yang cukup akurat, serta penggunaan Ontologi mampu mengkategorikan penyebab kecelakaan lalu lintas. Nilai akurasi yang didapatkan oleh model Actor memiliki

precision sebesar 99,31%, recall sebesar 98,56%, dan F1 score sebesar 98,93%. Untuk model Location memiliki precision sebesar 99,54%, recall sebesar 98,37%, dan F1 score sebesar 98,95%. Untuk model Keterangan memiliki precision sebesar 99,83%, recall sebesar 90,58%, dan F1 score sebesar 94,98%. Untuk model Time memiliki precision sebesar 100%, recall sebesar 100%, dan F1 score sebesar 100%.

Kata kunci: Semantic Web, Ontologi, Named Entity Recognition, Twitter, Kecelakaan Lalu Lintas

INFORMATION EXTRACTION OF TWITTER SOCIAL MEDIA ABOUT TRAFFIC ACCIDENT USING ONTOLOGY-BASED INFORMATION EXTRACTION APPROACH

Name : **Yasin Awwab**
NRP : **05211540000127**
Department : **Information System FTIK-ITS**
Supervisor : **Nur Aini Rakhmawati, S.Kom., M.Sc. Eng., Ph.D**

ABSTRACT

Traffic accidents are one of the most common occurrences in Indonesia. From the data collected by KORLANTAS POLRI, it's known that every 3 months there are around 25,000 traffic accidents. Information about traffic accidents is shared by many social media users including Twitter. Twitter has various information about traffic accidents. However, the information on Twitter has not yet been processed and mapped about the information. Therefore, this study aims to process and map information about traffic accidents found on Twitter so it can be provide more benefits to the community as well as become the input in the development related to traffic in Indonesia. The method in this study uses the ontology domain and Named-Entity Recognition for data extraction processes. The Named-Entity Recognition method is used to get keywords from a tweet based on class categories such as actors, time, location, and information about the causes of the accident. This study produced a model based on the Named Entity Recognition method that can produce a fairly accurate level of accuracy, and the use of Ontology is able to categorize the causes of traffic accidents. The accuracy value obtained by the Actor model has a precision of 99.31%, a recall of 98.56%, and an F1 score of 98.93%. For Model Location has a precision of 99.54%, a recall of 98.37%, and an F1 score of 98.95%. For the model

information has a precision of 99.83%, a recall of 90.58%, and an F1 score of 94.98%. The Time model has a precision of 100%, a recall of 100%, and an F1 score of 100%.

Keywords: Semantic Web, Ontology, Named Entity Recognition, Twitter, Traffic Accidents

KATA PENGANTAR

Puji syukur penulis haturkan ke hadirat Tuhan YME yang telah memberikan anugerah dan tuntunan kepada penulis sehingga penulis dapat menyelesaikan tugas akhir dengan judul *"EKSTRAKSI INFORMASI MEDIA SOSIAL TWITTER MENGENAI KECELAKAAN LALU LINTAS MENGGUNAKAN PENDEKATAN ONTOLOGY-BASED INFORMATION EXTRACTION"* sebagai salah satu syarat kelulusan pada Departemen Sistem Informasi, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember Surabaya. Penyusunan tugas akhir ini senantiasa mendapatkan dukungan dari berbagai pihak baik dalam bentuk doa, motivasi, semangat, kritik, saran dan berbagai bantuan lainnya. Untuk itu, secara khusus penulis akan menyampaikan ucapan terima kasih yang sedalam-dalamnya kepada:

1. Segenap keluarga besar terutama kedua orang tua penulis, Bapak Purnawirawanto, dan Ibu Sayidah Orbariana yang senantiasa mendoakan, memberikan motivasi dan semangat, sehingga penulis mampu menyelesaikan pendidikan S1 ini dengan baik.
2. Ibu Mahendrawathi Er, S.T., M.Sc., Ph.D. selaku Kepala Departemen Sistem Informasi ITS, Bapak Nisfu Asrul Sani, S.Kom, M.Sc selaku KaProdi S1 Sistem Informasi ITS serta seluruh dosen pengajar beserta staf dan karyawan di Jurusan Sistem Informasi, FTIF ITS Surabaya selama penulis menjalani kuliah
3. Ibu Nur Aini Rakhmawati, S.Kom., M.Sc., Eng. Ph.D dosen pembimbing yang telah banyak meluangkan waktu untuk membimbing, mengarahkan, dan mendukung dengan memberikan ilmu, petunjuk, dan motivasi dalam penyelesaian Tugas Akhir
4. Ibu Renny Pradina K., ST, MT, SCJP sebagai dosen wali penulis selama menempuh pendidikan di Departemen Sistem Informasi.

5. Bapak Faizal Johan Atletiko, S.Kom., M.T dan Ibu Irmasari Hafidz, S.Kom, M.Sc selaku dosen penguji yang telah memberikan kritik, saran, dan masukan yang dapat menyempurnakan Tugas Akhir ini.
6. Teruntuk Gienzka Azzahra Wibowo Putri yang selalu ada untuk memberikan semangat yang tiada henti, memberikan motivasi, dan membatu proses pengerjaan tugas akhir ini.
7. Teman-teman Ex-Dukitty yang telah berbagi cerita kehidupan kampus terutama untuk Benedict Timoni Christianto yang menjadi kolega penulis dalam proses pengerjaan tugas akhir.
8. Teman-teman Sistem Informasi angkatan 2015 (LANN15TER) yang senantiasa menemani dan memberikan motivasi bagi penulis selama perkuliahan hingga dapat menyelesaikan tugas akhir.
9. Segenap dosen dan karyawan Departemen Sistem Informasi.
10. Serta seluruh pihak-pihak lain yang tidak dapat disebutkan-satu per satu yang telah banyak membantu penulis selama perkuliahan hingga dapat menyelesaikan tugas akhir ini.

Terima kasih atas segala bantuan, dukungan, serta doanya. Semoga Tuhan YME senantiasa melimpahkan anugerah serta membalas kebaikan yang telah diberikan kepada penulis.

Penulis menyadari bahwa masih terdapat kekurangan dalam penyusunan tugas akhir ini, oleh karena itu penulis mengharapkan saran dan kritik yang membangun demi kebaikan penulis dan tugas akhir ini. Akhir kata, penulis berharap bahwa tugas akhir ini dapat memberikan kebermanfaatan.

Surabaya, Juni 2019

Penulis

DAFTAR ISI

Halaman ini sengaja dikosongkan.

DAFTAR GAMBAR

DAFTAR TABEL

Halaman ini sengaja dikosongkan.

DAFTAR KODE PROGRAM

BAB I
PENDAHULUAN

Pada bab ini berisi mengenai gambaran umum tugas akhir meliputi latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat tugas akhir, dan relevansi tugas akhir di Laboratorium Akuisisi Data dan Diseminasi Informasi:

1.1 Latar Belakang

Media sosial merupakan sebuah media online yang sedang banyak dinikmati oleh masyarakat dari banyak kalangan. Media sosial menyediakan cara berkomunikasi aktif bagi setiap penggunanya. Tidak hanya organisasi maupun kelompok, media sosial juga bisa digunakan oleh individu tanpa batasan yang jelas. Meningkatnya jumlah pengguna media sosial juga dikarenakan faktor mudahnya akses bagi setiap orang dan semua kalangan dapat dengan mudah bergabung menjadi pengguna. Media sosial *Twitter* menjadi salah satu media yang paling populer [1].

Twitter merupakan media sosial berbasis mobile dan web yang menghadirkan fitur untuk saling berbagi informasi secara mudah. Pengguna *Twitter* dapat memposting sebuah *tweet* untuk berbagi informasi dengan pengguna lain, *tweet* tersebut dapat berupa tulisan, gambar, maupun berbagi link video. Sebagai layanan microblogging terbesar, *Twitter* telah menarik perhatian perusahaan pada perilaku dan layanan yang dimiliki penggunanya. Dengan kekokohannya, *Twitter* semakin banyak digunakan oleh organisasi berita dalam menerima pembaruan selama keadaan darurat [2]. Keadaan darurat tersebut salah satunya merupakan kecelakaan lalu lintas.

Kecelakaan lalu lintas merupakan salah satu kejadian darurat yang sering terjadi di Indonesia. Pada bulan Oktober - Desember 2018, total telah terjadi lebih dari 25.000 kejadian kecelakaan lalu lintas [3]. Dengan banyaknya kejadian tersebut,

sumber informasi mengenai kecelakaan lalu lintas merupakan suatu hal yang penting untuk diolah menjadi sebuah informasi yang dapat dipelajari. Sumber informasi kecelakaan lalu lintas tersebut dapat diambil dari berbagai sumber informasi, salah satunya yaitu tersebar di berbagai media sosial termasuk *Twitter*. Informasi kecelakaan lalu lintas yang terdapat di media sosial *Twitter* merupakan salah satu sumber informasi kecelakaan lalu lintas yang besar dan beragam. Dengan melakukan ekstraksi informasi mengenai kecelakaan lalu lintas dari sumber informasi *Twitter* diharapkan dapat membantu dalam pengembangan dan perbaikan lalu lintas kedepannya. Namun, informasi mengenai kecelakaan lalu lintas yang terdapat di media sosial *Twitter* sulit untuk diekstraksi karena ekspresi linguistik dari insiden dapat bervariasi secara signifikan di antara pengguna *Twitter* yang berbeda [4].

Dari permasalahan terkait ekstraksi informasi tersebut, perlu adanya sebuah *model* yang dapat mengekstraksi sebuah informasi mengenai kecelakaan lalu lintas yang terdapat di media sosial *Twitter* sehingga dapat dilakukan pemetaan berdasarkan aktor, lokasi, waktu dan keterangan penyebab kecelakannya. Oleh sebab itu perlu dibuat sebuah ontologi menggunakan Bahasa Indonesia yang terkait dengan kecelakaan lalu lintas sebagai kerangka *knowledge base* pada proses ekstraksi informasi untuk memetakan komponen kecelakaan lalu lintas yang akan digunakan dalam penelitian ini. Kemudian dengan menggunakan metode *Named Entity Recognition* dapat mendeteksi *named-entity* pada setiap *tweet* yang didapatkan dari *Twitter*. Melalui penelitian ini diharapkan mampu melakukan ekstraksi informasi mengenai kecelakaan lalu lintas di media sosial *Twitter* sehingga dapat memberikan manfaat maupun dapat menjadi masukan dalam pengembangan yang berhubungan dengan lalu lintas di Indonesia.

1.2 Perumusan Masalah

Berdasarkan uraian latar belakang, maka rumusan permasalahan yang menjadi fokus dan akan diselesaikan dalam Tugas Akhir ini antara lain:

1. Bagaimana mendesain domain ontologi yang dapat memilah data dalam pengambilan informasi mengenai kecelakaan lalu lintas dari media sosial *Twitter*?
2. Bagaimana cara membuat model menggunakan metode *Named Entity Recognition* pada ekstraksi informasi kecelakaan lalu lintas?
3. Bagaimana cara merancang platform untuk visualisasi data jenis kecelakaan ke dalam bentuk *dashboard*?

1.3 Batasan Masalah

Dari permasalahan yang disebutkan di atas, batasan masalah dalam tugas akhir ini adalah:

1. Studi kasus yang digunakan pada penelitian ini yaitu wilayah dengan tingkat terkecil kacamatan.
2. Menggunakan data yang diambil dari *Twitter* mengenai topik kecelakaan lalu lintas.
3. *Named entity* menggunakan entitas berupa aktor kejadian *(ACTOR)*, nama tempat *(LOCATION)*, waktu kejadian *(TIME)* dan keterangan penyebab kecelakaan lalu lintas (KETERANGAN).
4. Entitas TIME dibatasi hanya menggunakan format tanggal *dd-mm-yyyy*.
5. Rentang waktu yang digunakan untuk visualisasi hanya tahun 2018 – 2019.

1.4 Tujuan

Berdasarkan hasil perumusan masalah dan batasan masalah yang telah disebutkan sebelumnya, maka tujuan yang dicapai dari tugas akhir ini adalah untuk melakukan ekstraksi informasi mengenai kecelakaan lalu lintas dari media sosial *Twitter* untuk

mengetahui persebaran data kecelakan lalu lintas sesuai dengan entitas yang telah ditentukan. Sehingga dengan adanya penelitian ini dapat memberikan manfaat maupun dapat menjadi masukan dalam pengembangan yang berhubungan dengan lalu lintas di Indonesia.

1.5 Manfaat

Berikut ini adalah manfaat yang diperoleh berdasarkan tiga sudut pandang, yang diantaranya yaitu sudut pandang penulis, perusahaan, dan teoritis. Manfaat yang diharapkan dapat diperoleh dalam pengerjaan tugas akhir ini adalah :

1. Manfaat Sudut Pandang Penulis
   Menambah ilmu pengetahuan terkait akuisi data menggunakan metode *NER (Named Entity Recognition)*.
2. Manfaat Sudut Pandang Pemerintah
   Mengetahui lokasi yang sering terjadi kecelakaan pada kendaraan untuk lebih diperhatikan kedepannya dan mengetahui persebaran kecelakaan yang paling sering terjadi di suatu daerah agar bisa menjadi menjadi masukan dalam pengembangan terkait lalu lintas di Indonesia.
3. Manfaat Sudut Pandang Teoritis
   Penulisan tugas akhir ini dapat dijadikan referensi dari penelitian terkait untuk kedepannya. Penulisan tugas akhir ini dapat dijadikan referensi dari penelitian terkait untuk kedepannya.

1.6 Relevansi

Tugas akhir ini disusun untuk memenuhi salah satu syarat kelulusan sebagai Sarjana Komputer Departemen Sistem Informasi, Fakultas Teknologi Informasi dan Komunikasi, Institut Teknologi Sepuluh Nopember, Surabaya. Laboratorium Akuisisi Data dan Diseminasi Informasi (ADDI) di Departemen Sistem Informasi ITS memiliki tiga topik utama yang

diantaranya yaitu Akuisisi Data, Paradigma Pengolahan, dan Diseminasi Informasi. Penelitian tugas akhir ini mengambil topik Akuisisi Data sebagai topik utamanya. Adapun mata kuliah yang berkaitan dengan topik yang bersangkutan adalah Teknologi Web (TW) dan Pemrograman Bahasa Alami (PBA).

Halaman ini sengaja dikosongkan.

BAB II
TINJAUAN PUSTAKA

Bab ini akan menjelaskan mengenai penelitian sebelumnya dan dasar teori yang dijadikan acuan atau landasan dalam pengerjaan tugas akhir ini. Landasan teori akan memberikan gambaran secara umum dari landasan penjabaran tugas akhir ini.

2.1 Penelitian Terkait

Berikut adalah penelitian yang terkait dengan tugas akhir yang diusulkan :

Tabel 2.1 Literatur 1

Judul	*Spatiotemporal and semantic information extraction from Web news reports about natural hazards*
Penulis	Wei Wang, Kathleen Stewart
Ringkasan	Penelitian ini bertujuan untuk melakukan ekstraksi data melalui website news reports mengenai kejadian alam yang berbahaya dan kemudian ditampilkan dalam bentuk geocoding. Metode yang digunakan pada penelitian ini adalah dengan mengekstrak web text document berita yang kemudian dilakukan preprocessing data menggunakan aplikasi GATE. Penelitian ini menggunakan aplikasi ontologi editor dengan menggunakan. Tahap terakhir dengan geocoding dan geovisualization.

Berdasarkan penjelasan pada tabel di atas mengenai penelitian terkait yang berjudul "*Spatiotemporal and semantic information extraction from Web news reports about natural hazards*" dapat diketahui perbedaan dengan tugas akhir yang dikerjakan, yaitu:

1. Penelitian sebelumnya membahas mengenai ontologi berdasarkan laporan berita pada website mengenai kejadian alam yang berbahaya, sedangkan pada tugas akhir yang diusulkan membahas mengenai ontologi berdasarkan sosial media mengenai kecelakaan lalu lintas.
2. Penelitian sebelumnya menggunakan ontologi editor Neon dan Gate sebagai aplikasi penunjang pengolahan data, sedangkan pada penelitian yang diusulkan menggunakan bahasa pemrograman *python* dan aplikasi ontologi editor Protégé.

Tabel 2.2 Literatur 2

Judul	*A methodology for traffic-related Twitter messages interpretation*
Penulis	Fa´bio C. Albuquerque, Marco A. Casanova, He´lio Lopes, Luciana R. Redlich, Jose´ Antonio F. de Macedo, Melissa Lemos, Marcelo Tilio M. de Carvalho, Chiara Renso
Ringkasan	Pada penelitian ini bertujuan untuk melakukan interpretasi dari data yang telah diekstrak dari *Twitter* mengenai berbagai macam hal yang berkaitan dengan lalu lintas. Data-data yang didapatkan akan diolah dengan menggunakan ontologi dengan model TEDO yang merupakan *Traffic Event Domain Ontologi.* Sehingga data yang diinterpretasikan terdiri dari nama kejadian, aktor yang terlibat, lokasi, dan waktu.

Berdasarkan penjelasan pada tabel di atas mengenai penelitian terkait yang berjudul "*A methodology for traffic-related Twitter messages interpretation*" dapat diketahui perbedaan dengan tugas akhir yang dikerjakan, yaitu:

1. Penelitian sebelumnya membahas mengenai sebuah kejadian yang berhubungan dengan lalu lintas seperti kemacetan, kejahatan lalu lintas, jalan rusan dan lain

sebagainya. Sedangkan dalam penelitian ini memiliki fokus kepada kejadian kecelakaan lalu lintas saja.

Tabel 2.3 Literatur 3

Judul	*Fuzzy ontologi-based sentiment analysis of transportation and city feature reviews for safe traveling*
Penulis	Farman Ali, Daehan Kwak, Pervez Khan, S.M. Riazul Islam, Kye Hyun Kim, K.S. Kwak
Ringkasan	Penelitian ini membahas mengenai pengolahan data lalu lintas yang terdapat pada kota-kota besar dengan tujuan untuk meningkatkan kenyamanan dan keamanan selama perjalanan. Penelitian ini berdasarkan data yang diambil pada social networking dan online discussion yang terdapat di forum website. Data yang didapatkan kemudian melalui tahap pre-processing data. Proses ini mempertimbangkan setiap kata adjectaive dan adverbs di tiap kata nya. Kemudian diolah menggunakan Ontologi SWRL dan Ontologi Fuzzy berdasarkan score tiap senti-word. Data yang dihasilkan ditujukan untuk pengendara di jalan raya dan perusahan transportasi.

Berdasarkan penjelasan pada tabel di atas mengenai penelitian terkait yang berjudul "*Fuzzy ontologi-based sentiment analysis of transportation and city feature reviews for safe traveling*" dapat diketahui perbedaan dengan tugas akhir yang dikerjakan, yaitu:

2. Penelitian sebelumnya menggunakan ontologi *Fuzzy* dan ontologi SWRL dimana pada penelitian ini menggunakan ontologi SWRL.
3. Penelitian sebelumnya menganalisis sentimen mengenai lalu lintas dari media sosial dan webiste sedangkan penelitian ini menganalisis sentimen dari media sosial *Twitter* mengenai kecelakaan lalu lintas.

Tabel 2.4 Literatur 4

Judul	*Social big data: Recent achievements and new challenges*
Penulis	Gema Bello-Orgaz, Jason J. Jung, David Camachoa
Ringkasan	Penelitian ini membahas mengenai pemilihan social media sebagai sumber data yang akan digunakan. Sosial media sudah menjadi salah satu sumber data yang paling representative dan relevan. Pengambilan dan pemrosesan data yang ada di dalam social media menggunakan paradigma algoritma MapReduce dan Apache Spark.

Berdasarkan penjelasan pada tabel di atas mengenai penelitian terkait yang berjudul "*Social big data: Recent achievements and new challenges*" dapat diketahui perbedaan dengan tugas akhir yang dikerjakan, yaitu:

1. Penelitian sebelumnya membahas hanya mengenai pengambilan data melalui *social media* secara keseluruhan. Sedangkan pada penelitian kali ini membahas social media *Twitter*.
2. Penelitian sebelumnya menggunakan metode *MapReduce* sedangkan pada penelitian ini menggunakan metode *Named Entity Recognition*.

Tabel 2.5 Literatur

Judul	*Analysis of named entity recognition and linking for tweets*
Penulis	Leon Derczynski, Diana Maynard, Giuseppe Rizzo, Marieke van Erp, Genevieve Gorrell, Raphaël Troncy, Johann Petrak, Kalina Bontcheva
Ringkasan	Penelitian ini bertujuan untuk melakukan analisis kekokohan (robustness) dari metode Named Entity Recognition dan mencaritahu penyebab terjadinya false negative dan false positive. Penelitian ini melakukan ekstraksi data dari media sosial *Twitter*.

	Penelitian lebih membahas mengenai analisis dari proses ekstraksi data dari media sosial *Twitter*. Penelitian ini juga membandingkan berbagai macam metode dalam menggunakan Named Entity Recognition. Dengan hasil berupa akurasi, presisi, dan recall dari setiap percobaan metode yang dilakukan.

Berdasarkan penjelasan pada tabel di atas mengenai penelitian terkait yang berjudul "*Analysis of named entity recognition and linking for tweets*" dapat diketahui perbedaan dengan tugas akhir yang dikerjakan, yaitu:

1. Penelitian sebelumnya membahas mengenai analisis *Named Entity Recognition* dengan metode linking pada ekstraksi data *Twitter*, sedangkan penelitian ini membahas mengenai ekstraksi data dari *Twitter* menggunakan metode *Named Entity Recognition* pada sentimen kecelakaan lalu lintas.

2.2 Dasar Teori

Berikut ini adalah latar belakang dan dasar teori yang digunakan untuk menentukan pembuatan pada penelitian ini.

2.2.1 Tweetscraper

Tweet scraping atau biasa disebut *tweetscraper* memiliki konsep yang sama dengan Web Scraping yang merupakan sebuah proses pengambilan sebuah dokumen semi-terstruktur dari internet, umumnya berupa halaman-halaman web dalam bahasa markup seperti HTML atau XHTML, dan menganalisis dokumen tersebut untuk diambil data tertentu dari halaman tersebut untuk digunakan bagi kepentingan lain [6]. Sedangkan *tweetscraper* merupakan sebuah proses untuk mengekstrak sebuah informasi dari suatu *tweet* yang terdapat pada sosial media *Twitter*. *Tweetscraper* dapat dilakukan menggunakan API yang tersedia maupun tidak menggunakan API. Bahasa

pemrograman yang biasa digunakan dalam proses *tweetscraper* adalah R, *Python, Java*, dan lain sebagainya.

2.2.2 Named Entity Recognition

Name Entity Recognition (NER) atau Name Entity Recognition and Classification (NERC) adalah salah satu dari beberapa komponen utama dari information extraction yang memiliki tujuan untuk mendeteksi dan mengklasifikasikan sebuah named-entity pada sebuah string (teks) [7].

Gambar 2.1 Contoh deteksi entitas dengan *NER*

Tiap ekspresi dari entitas akan dianotasikan berdasarkan beberapa keunikan dari entitas tersebut. Keunikan tersebut diantaranya ENAMEX yang terdiri dari entitas yang menunjukkan pada suatu organisasi, perorangan, dan lokasi. Kemudian TIMEX yang terdiri dari entitas yang menunjukkan tanggal dan waktu. Dan terakhir adalah NUMEX yaitu entitas yang menunjukkan suatu persentase dan nilai moneter [8].
NER memiliki berbagai macam manfaat dalam banyak aplikasi NLP *(Natural Language Processing)* misalnya question-answering, rangkuman dan sistem dialog. NER memiliki sebuah keterkaitan dengan *task information extraction* lain misalnya seperti relation *detection, event detection* dan *temporal analysis.*
Untuk melakukan sebuah deteksi *named entity*, NER melihat pola kata disekitarnya. NER juga diselesaikan dengan pelabelan pada urutan kata statistic *(statisctical sequence-labeling)* yang mendeteksi batas atau segmen dan tipe dari *named entity*.

2.2.3 Ontologi

Ontologi adalah spesifikasi dari sebuah klasifikasi [9]. *Ontologi* terbagi menjadi 4 komponen yang diantaranya yaitu:

2.2.3.1 Concept

Concept (Konsep) adalah sebuah kumpulan dari berbagai objek. Konsep adalah bentuk fundamental elemen dari sebuah domain yang biasanya merepresntasikan kumpulan dari kelompok yang memiliki kesamaan.

2.2.3.2 Instance

Instance atau biadanya dikenal sebagai *individual* adalah sebuah komponen *"ground-level"* yang merepresentasikan objek yang spesifik dari sebuah konsep.

2.2.3.3 Relation

Relation (Relasi) adalah suatu hubungan antara dua konsep dalam sebuah domain.

2.2.3.4 Axioms

Axioms digunakan untuk memaksakan kendala atau masalah pada nilai dari suatu kelas.
Ontologi memiliki struktur yang dapat dideskripsikan seperti :

$$5\text{-}tuple\ O: = (C, H^C, R, H^R, I),$$

Dimana,

- C merepresentasikan kumpulan dari konsep *(instances of "rdf:Class")*. Konsep ini disusun sesuai subsumsi dari hirarki H^C.
- R merepresentasikan kumpulan dari relasi yang berhubungan dengan suatu konsep dengan konsep lainnya *(instances of "rd:Property").* $Ri \in R$ dan $Ri \rightarrow C \times C$.
- H^C merepresentasikan sebuah hirarki konsep dalam sebuah relasi *(a binary relation corresponding to*

"rdfs:subClassOf"). $H^C \subseteq C \times C$, dimana $H^C (C_1, C_2)$ menunjukkan bahwa C_1 adalah subkonsep dari C_2.

- H^R merepresentasikan sebuah relasi hirarki dalam bentuk relasi $H^R \subseteq R \times R$, dimana $H^R (R_1, R_2)$ menunjukkan bahwa R_1 adalah subrelasi dari R_2 *("rdfs:subPropertyOf")*.
- I adalah instansiasi dari sebuah konsep di dalam domain *("rdf:type")*.

Ontologi sudah menjadi pencarian yang populer di dalam berbagai disiplin ilmu dengan berbagai studi kasus yang dilakukan. *Ontologi* pertama kali ada di dalam AI laboratorium, sebelum digunakan ke ranah yang lainnya, contohnya seperti pengaplikasian ontologi dalam bidang:
Semantic Web: Ontologi berperan penting dalam memainkan peran di dalam *semantic web* untuk mendukung pertukaran informasi yang tersebar dari berbagai lingkungan. *Semantic web* merepresentasikan data ke dalam *machine-processable way*, dimana itu menjadi suatu alas an untuk menjadi sebuah ekstensi dari sebuah web yang sudah ada [10].
Penelitian ini membuat *ontology* dengan menggunakan dua *Class* yaitu *Class* KecelakaanLaluLintas dan *Class Actor*. *Class* Kecelakaan lalu lintas dan *Class Actor* menggunakan dasar teori sebagai berikut:

2.2.3.5 Class KecelakaanLaluLintas

Class KecelakaanLaluLintas merupakan sebuah kelas yang akan dibuat pada *domain ontogy* untuk mengkategorikan suatu penyebab Kecelakaan Lalu Lintas. Sedangkan yang dimaksud dengan kecelakaan lalu lintas menurut pasal 93 Peraturan Pemerintah (PP) Nomor 43 tahun 1993 ayat 1 berdasarkan ketentuan yang telah di tetapkan adalah:

"Suatu peristiwa ijalan yang tidak disangka-sangka dan tidak disengaja melibatkan kendaraan dengan atau tanpa pemakai jalan lainnya mengakibatkan korban manusia atau kerugian harta benda"

Setiap *Class* memiliki faktor masing-masing. Untuk Class KecelakaanLaluLintas memiliki faktor penyebab kecelakaan tersebut. Faktor penyebab kecelakaan lalu lintas tersebut dikategorikan berdasarkan teori dari Direktorat Jendral Dinas Perhubungan Darat dapat dikategorikan berdasarkan 4 kategori [5], yang diantaranya yaitu :

a. Pengemudi

Faktor pengemudi mengambil andil yang cukup besar sebagai penyebab kecelakaan lalu lintas. Pengemudi yang mengendarai kendaraan haruslah dalam keadaan yang sehat, penuh konsentrasi dan sudah harus memiliki izin mengemudi. Kecelakaan lalu lintas yang disebabkan oleh pengemudi dapat disebabkan oleh beberapa faktor yang diantaranya yaitu lengah, mengantuk, tidak terampil, mabuk, kecepatan tinggi, tidak menjaga jarak, kesalahan pejalan dan gangguan binatang.

b. Kendaraan

Kecelakaan lalu Lintas tentu tidak dapat terlepas dari faktor kendaraan. Berdasarkan Peraturan Pemerintah (PP) No. 44 Tahun 1993 pasal 1 tentang Kendaraan dan Pengemudi (peraturan pelaksana dari Undang-undang Lalu Lintas dan Angkutan Jalan), kendaraan bermotor adalah kendaraan yang digerakkan oleh peralatan teknik yang berada pada kendaraan itu. Kendaraan bermotor dapat dikategorikan menjadi beberapa golongan, yang diantaranya yaitu: sepeda motor, mobil penumpang, mobil bus, mobil barang, dan kendaraan khusus. Faktor penyebab kecelakaan lalu lintas berdasarkan kendaraan diantaranya disebabkan oleh ban pecah, kerusakan sistem rem, kerusakan sistem kemudi, as/ kopel lepas, sistem lampu tidak berfungsi.

c. Jalan

Jalan merupakan jalur utama yang dilalui oleh pengemudi kendaraan. Jalan memiliki indicator kelayakan untuk dilalui

oleh pengemudi. Untuk membuat indikator kelayakan yang layak, tentu perlu dibuat perencanaan yang matang, baik dalam perencaan indicator maupun perencanaan praktik pembuatan jalannya. Penyebab kecelakaan lalu lintas yang disebabkan oleh jalan diantaranya disebabkan oleh persimpangan, jalan sempit, akses yang tidak dikontrol atau dikendalikan, marka jalan kurang atau tidak jelas, tidak ada rambu batas kecepatan, permukaan jalan licin.

d. Lingkungan

Terjadinya kecelakaan yang disebabkan oleh lingkungan dapat terjadi karena lalu lintas campuran antara kendaraan cepat dengan kendaraan dengan pejalan, pengawasan dan penegakan hhukum belum efektif, pelayanan gawat darurat yang kurang cepat. Selain itu seperti cuaca juga menjadi indicator kecelakaan lalu lintas. Cuaca yang : gelap, hujan. Kabut, asap.

Besarnya persentase masing-masing faktor penyebab kecelakaan lalu lintas di Indonesia tersebut yaitu sebesar 93,52% untuk faktor manusia, 2,76% untuk faktor kendaraan, 3,23% untuk faktor jalan, dan 0,49% untuk faktor lingkungan. Kecelakaan lalu lintas yang sering didapati di Indonsia bisa terjadi karena kombinasi dari keempat faktor penyebab tersebut [5]. Berikut ini adalah tabel ringkasan dari faktor penyebab terjadinya kecelakaan seperti dijelaskan pada poin-poin diatas:

Tabel 2.6 Kategori faktor penyebab kecelakaan lalu lintas

Faktor Penyebab	Keterangan	Presentase (%)
Pengemudi	Lengah, mengantuk, tidak terampil, mabuk, kecepatan tinggi, tidak menjaga jarak, kesalahan pejalan, gangguan binatang	93,52%

Kendaraan	Ban pecah, kerusakan system rem, kerusakan system kemudi, as/kopel lepas, system lampu tidak berfungsi	2,76%
Jalan	Persimpangan, jalan sempit, akses yang tidak dikendalikan, marka jalan kurang, tidak ada rambu, permukaan jalan licin	3.23%
Lingkungan	Lalu lintas campuran, penegakan hokum kurang efektif, cuaca : gelap, hujan, kabut, asap	0,49%

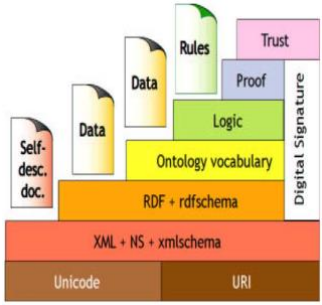
2.2.3.6 Class Actor

Class Actor pada penelitian ini dibuat berdasarkan subjektivitas peneliti dengan mempertimbangkan *keyword* yang paling sering ditemukan setelah melakukan *crawling* data. *Keyword* yang telah ditentukan sebagai *subclass* dari *class Actor* adalah Mobil, Truk, Bis, Becak, Sepeda_motor, dan Sepeda.

2.2.4 Semantic Web

Semantic web merupakan sebuah konsep pemikiran dari bagaimana memiliki data pada web yang dapat didefinisikan dan dihubungkan dengan suatu cara sehingga dapat dimengerti oleh mesin dengan tujuan untuk otomatisasi, integrasi dan penggunaan kembali data diantara berbagai aplikasi. Dengan adanya semantic web maka berbagai perangkat lunak akan

mampu mencari, membagi, dan mengintegrasikan informasi dengan cara yang lebih muda [11].
Pada proses pembuatannya, semantic web didasarkan pada beberapa standar yang telah dikoordinasi oleh *World Wide Web Consortium (W3C)*. Beberapa standar yang sering diterapkan dalam membangun sebuah *semantic web* adalah XML, XML Schema, RDF, OWL, dan SPARQL. W3C [12] merekomendasikan susunan layer semantic web sebagai berikut:



Gambar 2.2 Arsitektur *Semantic Web*

2.2.5 Protégé

Protégé merupakan sebuah perangkat lunak open-source ontologi editor dan sebuah framework yang dikembangkan oleh sebuah organisasi yang bernaung dibawah Standford. *Protégé* bekerja sebagai alat bantu yang digunakan untuk mengembangkan sebuah sistem yang berbasis pada Knowledge-Base System [13]. *Protégé* bekerja dengan cara membuat sebuah domain ontologi, menyesuaikan form untuk entry data, dan memasukan data.
Protégé mendukung berbagai format penyimpanan yang berbasis OWL, RDF, XML, maupun HTML. Protégé menyediakan kemudahan berupa penggunaan yang berbasis *plug-in architecture* sehingga lebih fleksibel dalam membangun sebuah aplikasi berbasis ontologi yang simple maupun ontologi yang kompleks.

2.2.6 OpenNLP

OpenNLP adalah sebuah *library* dari *Java* yang digunakan untuk membantu dalam proses *Natural Language Processing* yang telah dikembangkan oleh *Apache* [14]. OpenNLP memiliki tujuan untuk mempermudah sebuah program computer dalam melakukan ekstraksi makna dari *natural language* [14]. OpenlNLP mampu mendukung beberapa tugas berikut:

- *Tokenization*
- *Sentence Segmentation*
- *Part -of-speech tagging*
- *Named Entity Recognition*
- *Chunking*
- *Parsing*
- *Coreference Resolution.*

OpenNLP melakukan pendekatan dengan cara menginisasi data model yang mendukung tugas tertentu kemudian memberikan *method* kepada model untuk melakukan tugas yang diinginkan. OpenNLP sendiri berjalan diatas *Java*, sehingga dalam pengoperasiannya membutuhkan *JDK (Java Development Kit)* dan *JRE (Java Run-time Environment)* [15].

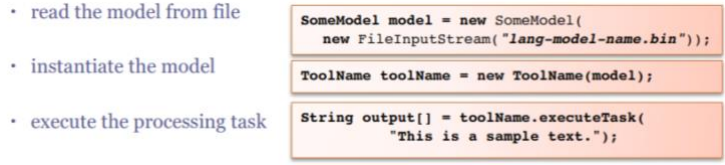
2.2.7 OpenNLP Library Structure

Library OpenNLP menyediakan komponen-komponen untuk menyelesaikan tugas-tugas tertentu yang spesifik. Komponen-komponen tersebut juga dapat dikombinasikan untuk membuat sebuah NLP *Pipeline*. Setiap komponen pada OpenNLP (tokenizer, NER, chunker, POS Tagging, dll) memiliki 3 fungsi yaitu:

- Mengeksekusi task dalam NLP ke dalam *input text stream*
- Melakukan training sebuah model untuk *NLP task*

- Mengevaluasi performa model yang dihasilkan terhadap data test.

Komponen OpenNLP dapat diakses baik melalui Command line ataupun Java API.



Gambar 2.3 Struktur *library OpenNLP*

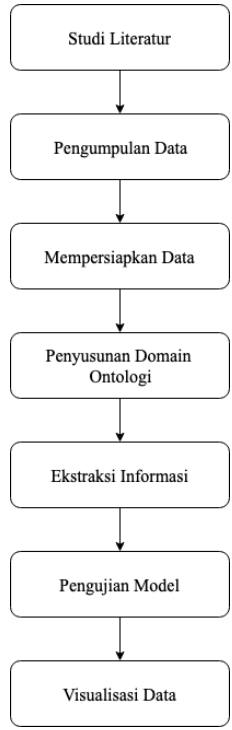
OpenNLP sendiri telah pre-built dengan model-model yang sudah siap digunakan. Namun model tersebut hanya tersedia untuk beberapa saja.

BAB III
METODOLOGI

Pada bab metode penelitian akan dijelaskan mengenai tahapan – tahapan apa saja yang dilakukan dalam pengerjaan tugas akhir ini beserta deskripsi dan penjelasan tiap tahapan tersebut. Lalu disertakan jadwal pengerjaan tiap tahapanan.

3.1 Tahapan Pengerjaan Tugas Akhir

Pada sub bab ini akan menjelaskan mengenai metodologi dalam pelaksanaan tugas akhir. Metodologi ini dapat dilihat pada Gambar 3 :



Gambar 3.1 Metodologi Penelitian

3.2 Studi Literatur

Tahap studi literatur ini dilakukan dengan memiliki tujuan agar dapat memahami konsep, metode, dan teknologi yang akan digunakan sesuai bahasan dan permasalahan yang telah dirumuskan sehingga dapat memberikan solusi yang akan diterapkan pada penyusunan penelitian tugas akhir ini. Adapun beberapa literatur yang digunakan dalam penelitian ini yaitu terkait metode penyusunan ontologi mengenai lalu lintas yang menggunakan data dari *Twitter* oleh Fabio C. Albuquerque [16] dan penggunaan metode Named Entity Recognition mengacu pada penelitian Leon Derczynski [17].

3.3 Pengumpulan Data

Tahap pengumpulan data terdiri dari aktivitas crawling data dari *Twitter* menggunakan modul *tweetscraper* dengan data yang telah ditentukan pada ontologi kemudian aktivitas selanjutnya adalah mengumpulkan data suatu wilayah dengan tingkat terkecil kacamatan dan mengelompokkannya kedalam kumpulan data lokasi berdasarkan kota masing – masing. Rangkaian tahapan ini dilakukan untuk mempersiapkan dokumen yang akan dianalisis sebagai data persebaran kecelakaan berdasarkan wilayah di Indonesia. Berikut adalah gambar bentuk penggolongan data wilayah yang akan dikumpulkan:

Gambar 3.2 Tingkat pengelompokan data wilayah

3.4 Mempersiapkan Data

Tahap mempersiapkan data terdiri dari aktivitas utama yaitu text pre-processing. Data yang didapatkan akan diolah menggunakan text pre-processing yang terdiri dari *Case Folding*, *Delete Duplicate*, dan *Data Cleansing*.

3.4.1 Case Folding

Tahap yang pertama kali harus dilakukan untuk mempersiapkan data adalah melakukan *case folding*. Case folding adalah suatu aktivitas dalam text processing dimana merubah bentuk lowercase atau uppercase untuk diseragamkan. Hal tersebut bertujuan untuk menghindari perbedaan penulisan huruf yang berbeda-beda. Tabel 3.1 berikut ini adalah contoh perbandingan penerapan *case folding*:

Tabel 3.1 Contoh proses *case folding*

Sebelum Case Folding	Sesudah Case Folding
Kecelakaan di jalan Tol	kecelakaan di jalan tol
Membutuhkan P3K	membutuhkan p3k
PRJ jalan tol	prj jalan tol

3.4.2 Delete Duplicate

Sub-tahap ini bertujuan untuk memudahkan mesin dalam melakukan ekstraksi informasi pada tahap selanjutnya, biasanya menghapus *tweet* yang dihasilkan oleh akun-akun robot. Tabel 3.2 berikut adalah contoh pada proses *delete duplication*:

Tabel 3.2 Contoh proses *delete duplicate*

Sebelum *Delete Duplicate*	Sesudah *Delete Duplicate*
• Polisi Tetapkan Tersangka Kasus Dugaan Pembajakan Truk Tangki Pertamina News	• Polisi Tetapkan Tersangka Kasus Dugaan Pembajakan Truk Tangki Pertamina News

• Polisi Sebut Pelaku Pembajakan Truk Pertamina Hanya Ingin Cari Perhatian • Polisi Sebut Pelaku Pembajakan Truk Pertamina Hanya Ingin Cari Perhatian	• Polisi Sebut Pelaku Pembajakan Truk Pertamina Hanya Ingin Cari Perhatian

3.4.3 Data Cleansing

Sub-tahap ini bertujuan untuk memudahkan mesin dalam melakukan ekstraksi informasi pada tahap selanjutnya, biasanya menghapus simbol-simbol, karakter, dan tanda baca yang tidak berguna dalam *tweet*. Berikut adalah contoh proses tersebut pada Tabel 3.3.

Tabel 3.3 Contoh proses *data cleansing*

Sebelum *Data Cleansing*	Sesudah *Data Cleansing*
Apa itu benang mrhnya? RT @TrioMacan2000 : Pertamina , Petral, BBM dan elpiji langka, pembajakan SMYRNI, peledakan ... http:// m.tmi.me/qGGZm	apa itu benang mrhnya pertamina petral bbm dan elpiji langka pembajakan smyrni peledakan

3.5 Penyusunan Domain Ontologi

Tahap penyusunan domain ontologi dari *tweet* mengenai kecelakaan lalu lintas berupa menjabarkan class dan properties yang akan digunakan pada ontologi. Penyusunan ontologi akan menggunakan class KecelakaanLaluLintas. Lingkup kecelakaan lalu lintas disini merupakan kejadian tidak terduga yang mengakibatkan kerugian. Class KecelakaanLaluLintas adalah domain dari properti tipe data, yang memiliki Actor, memiliki objek, memiliki lokasi primer, memiliki lokasi sekunder, dan memiliki waktu kejadian. Berikut adalah detail yang akan digunakan dalam class KecelakaanLaluLintas:

3.5.1 Aktor

Aktor dalam class KecelakaanLaluLintas merupakan pelaku berupa objek kendaraan yang berkaitan dengan kejadian kecelakaan lalu lintas. Aktor yang digunakan dalam ontologi yaitu mobil, sepedamotor, pejalan kaki, sepeda, truk, kontainer, dan kereta api.

3.5.2 Kecelakaan Lalu Lintas

Kecelakaan lalu lintas merupakan sebuah keterangan mengenai class yang menjelaskan mengenai penyebab terjadinya kecelakaan lalu lintas tersebut. Penyebab disini dibagi kedalam 4 faktor penyebab yaitu faktor pengemudi, faktor jalan, faktor lingkungan, dan faktor kendaraan.

3.6 Ekstraksi Informasi

Tahap ini merupakan sebuah tahap yang dilakukan untuk melakukan ekstraksi sebuah informasi yang ada didalam sebuah *tweet* dari pengguna media sosial. Untuk mendapatkan sebuah informasi dari suatu *tweet* diperlukan beberapa proses sebagai berikut:

3.6.1 Tagged Tweet

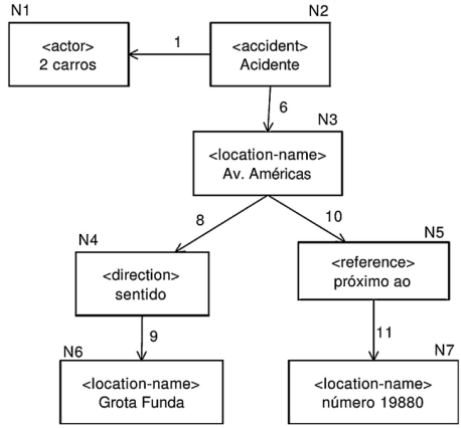
Proses ini merupakan bagian dari ekstraksi data menggunakan Named Entity Recognition (NER) dimana proses tersebut menerima sebuah input berupa *tweet* (W) dan mengembalikan *tweet* yang telah ditandai (T), yaitu *tweet* (W) dengan elemen teks yang diklasifikasikan dengan bantuan tag. Elemen teks yang ditandai dari T adalah pasangan (e, t), di mana e adalah elemen teks dari T dan t adalah tag yang sesuai dalam T [16]. Berikut adalah contoh pemberian tag pada *tweet* yang telah dilakukan oleh Fabio C. Albuquerque dalam penelitiannya yang berjudul "*A methodology for traffic-related Twitter messages interpretation*":

Tabel 3.4 Contoh proses kata yang dimasukkan dalam *tagging tweet*

No	Tag	Contoh
1	<kecelakaan>	“nabrak” (“tabrak”), “jatoh” (“jatuh”)
2	<lokasi>	“prapatan” (“perempatan”), “jalanan” (“jalan”)
3	<kendaraan>	“motor” (“motor”), “trek” (“truk”), “bus” (“bis”)
4	<kondisi-cuaca>	“ujan” (“hujan”), “baday” (“badai”)
5	<lalu-lintas>	“macet” (“macet”), “padet” (“padat”)

3.6.2 Dependency Tree

Proses ini merupakan bagian dari Relation Extraction yang mengacu pada masalah mendeteksi hubungan antara entitas yang diekspresikan dalam *tweet* yang telah ditandai. Proses tersebut menerima sebagai input *tweet* yang ditandai T dan mengembalikan dependancy tree QT = (NT, ET), di mana NT adalah himpunan elemen teks yang ditandai dari T dan ET adalah himpunan busur yang menunjukkan bagaimana elemen-elemen teks yang ditandai dalam NT terkait [16].



Gambar 3.3 Contoh *dependency tree*

3.7 Pengujian Model

Tahapan yang terakhir merupakan tahap pengujian dari ekstraksi yang telah dilakukan. Tahap pengujian ini meliputi beberapa aktivitas yang berutjuan untuk menguji data dan menampilkan hasil pemetaan dari data yang telah didapatkan. Berikut merupakan aktivitas yang akan dilakukan pada tahap pengujian :

3.7.1 Mengukur Performa Data Model

Pada aktivitas mengukur performa data model, dilakukan uji performa data untuk mengetahui *Precision*, *Recall*, *Accuracy*, dan *F-Measure* dari data yang berhasil di ekstraksi. Dari aktivitas ini akan didapatkan kualitas performa dari data yang didapatkan. Dapat diketahui bahwa rumus dari *recall, precision*, dan *F-measure* dapat di klasifikasikan seperti Gambar 3.4 dibawah ini :

$$\text{Precision} = \frac{T_p}{(T_p + F_p)}$$

$$\text{Recall} = \frac{T_p}{(T_p + F_n)}$$

$$F\text{-measure} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Gambar 3.4 Rumus menghitung *Precision, Recall,* dan *F-measure*

Dimana keterangan dari kalsifikasinya dapat dilihat seperti dibawah ini :

Tabel 3.5 Keterangan klasifikasi pengujian

Variable	Klasifikasi
Tp	True positive
Fp	False positive
Tn	True negative
Fn	False negative

3.8 PembuatanDashboard Visualisasi

Aktivitas pembuatan *dashboard* dilakukan untuk menampilkan informasi yang berhasil diekstraksi dari data. Dengan demikian informasi akan ditampilkan pada *platform website* dengan informasi berupa pemetaan kecelakaan lalu lintas berdasarkan penyebabnya disetiap lokasi daerah di Indonesia.

BAB IV
PERENCANAAN

Pada bab ini membahas mengenai perancangan terkait beberapa hal yang akan diperlukan pada proses pengerjaan penelitian sesuai dengan alur yang dijelaskan pada bab metodologi. Perancangan ini akan digunakan sebagai panduan dalam melakukan penelitian tugas akhir, yang dijelaskan sebagai berikut.

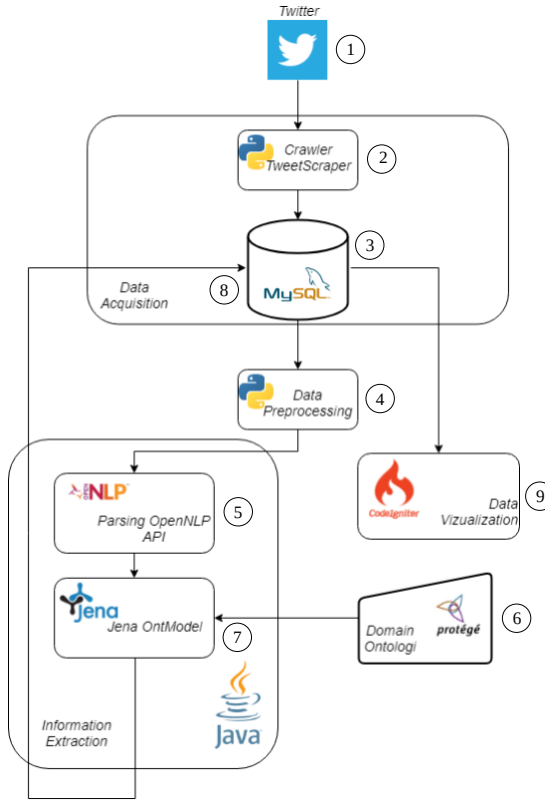
4.1 Arsitektur Sistem

Tahap ini akan melakukan aktivitas perancangan arsitektur sistem secara umum, yang nanti akan diimplementasikan sepanjang pengerjaan penelitian tugas akhir ini. merupakan gambar arsitektur sistem penelitian. Arsitektur sistem yang akan dibuat dapat dilihat pada Gambar 4.1. Adapun penjelasan dari arsitektur sistem adalah sebagai berikut:

1. Data yang digunakan dalam penelitian ini merupakan *tweet* mengenai kecelakaan lalu lintas.
2. *Data* tersebut didapatkan dari proses *crawling Twitter* menggunakan *tweetscraper*.
3. *Data* yang telah didapatkan kemudian akan disimpan di *database mySQL*.
4. *Data* yang tersimpan dalam *database* tersebut kemudian kemudian dilakukan *preprocessing*. Proses tersebut akan membersihkan *data* mencakup penghapusan duplikasi, *case folding*, *date merging*, *data cleansing*.
5. *Data* yang telah dibersihkan tersebut selanjutnya akan melalui proses ekstraksi informasi. Proses ini menggunakan *OpenNLP API Parser* untuk

mendapatkan informasi yang diinginkan dari suatu *tweet.*

6. Kemudian menyusun *Ontology* sebagai panduan dalam melakukan pengelompokan entitas berdasarkan kategori menggunakan *software* protégé.
7. Langkah selanjutnya adalah mengkategorikan *tweet* ke dalam kategori yang sudah dibuat pada *Domain Ontology* dengan bantuan *software Protégé*. Kategori yang terkandung di dalam ontologi berbentuk *class* dan *subclass* yang ditentukan di dalam buku panduan Pertamina. *Library* yang digunakan untuk membantu tahap ini adalah *Apache Jena.*
8. *Data* hasil proses ekstraksi informasi yang telah didapatkan akan disimpan kedalam *database MySQL*.
9. Pada tahap visualisasi menggunakan program berbasis *PHP* dan *javascript* dengan *framework code igniter* yang kemudian dibuat sedemikian rupa sehingga terbuat *dashboard* visualisasi.



Gambar 4.1 Arsitektur Sistem

4.2 Pengambilan Data

4.2.1 Desain Database Kecamatan

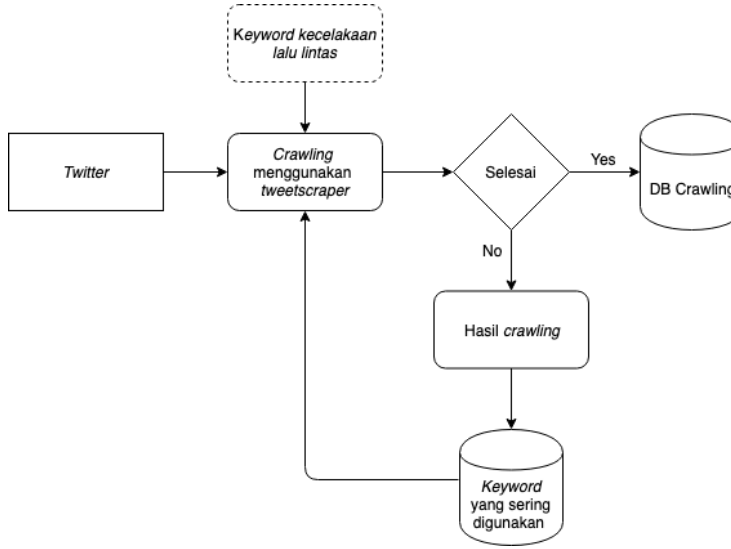
Penelitian ini menggunakan data lokasi berupa kecamatan diseluruh Indonesia. Data yang dibutuhkan berupa data tipe teks berisi nama-nama kecamatan. Dalam melakukan pengumpulan data dibutuhkan nama kecamatan, kota, dan provinsi sebagai atribut dari masing-masing data. Adapun atribut pada tabel kecamatan ditunjukkan pada Tabel 2.1 sebagai berikut.

Tabel 4.1 Atribut *database crawler*

Nama Attribut	Tipe Data	Keterangan
id	int	Merupakan primary key dari tiap kecamatan
kecamatan	varchar(255)	Merupakan nama kecamatan
kota	varchar(255)	Merupakan nama kota dari kecamatan tersebut
provinsi	varchar(255)	Merupakan nama provinsi dari kecamatan tersebut

4.2.2 Desain Crawling Data

Crawling data merupakan sebuah tahap dimana terdapat proses pengambilan data *tweet* dari *Twitter* yang kemudian data tersebut akan digunakan sebagai data dalam proses *pre-proccessing data*. Tahap *crawling data* ini menggunakan *keyword* mengenai kecelakaan lalu lintas pada suatu *tweet* di *Twitter*. Tahap *crawling data* di *Twitter* menggunakan *tweetscraper* yang kemudian dimasukkan dalam bentuk *array* dan selanjutnya disimpan dalam database. *Crawling* pertama kali mengetahui *keyword* yang paling sering muncul. Setelah itu dilakukan *crawling* dengan menggunakan *keyword* yang paling sering digunakan tersebut sehingga menghasilkan *database* berupa *tweet* hasil *crawling data*. Gambaran proses *crawling* dijelaskan pada Gambar 4.2.



Gambar 4.2 Alur *crawling data tweet*

4.2.3 Desain Database Crawler

Tabel 4.2 menunjukkan desain *database* yang digunakan untuk menyimpan hasil *crawling tweet* yang kemudian disimpan pada *database mySQL*.

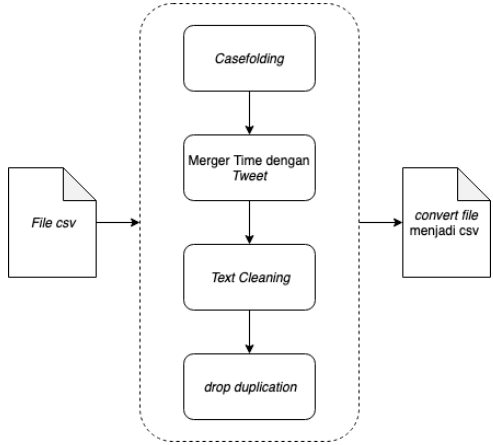
Tabel 4.2 Atribut pada *database crawler*

Nama Attribut	Tipe Data	Keterangan
id	char(20)	Merupakan primary key dari setiap *tweet*
url	varchar(140)	Merupakan url yang dimiliki oleh *tweet* tersebut
datetime	varchar(22)	Merupakan tanggal dan waktu kapan *tweet* tersebut dibuat
tweet	text	Merupakan konten dari suatu *tweet*
user_id	char(30)	Merupakan id user yang membuat *tweet*

UsernameTweet	varchar(20)	Merupakan username dari yang membuat *tweet*

4.3 Data Preprocessing

Alur pada tahap pra-proses data memiliki beberapa langkah pengerjaan utama yaitu melakukan pengubahan data teks menjadi huruf kecil (tahap *case folding*), menghapus data yang memiliki duplikasi (tahap *delete duplication*), dan melakukan pembersihan data dari simbol-simbol *tweeter* dan sebagainya (tahap *data cleansing*). Data yang didapatkan dari hasil *crawling* menunjukkan bahwa keterangan waktu *tweet* di *post* terdapat didalam kolom yang berbeda dengan isi *tweet*. Oleh sebab itu perlu melakukan merger isi tweet dengan keterangan waktu *posting* (*timestamp*) kedalam satu kolom yang sama. Untuk penjelasan secara lebih detail, dapat merujuk pada Gambar 4.3:



Gambar 4.3 Alur proses *preprocessing data tweet*

4.4 Desain Domain Ontologi

Perancangan Ontologi Kecelakaan Lalu Lintas dilakukan menggunakan perangkat lunak Protégé. Pada tahap ini akan ditentukan *Class* apa saja yang akan dibuat dalam ontology dan

memiliki instance, yaitu *Class Actor* dan *Class* Kecelakaan lalu lintas.

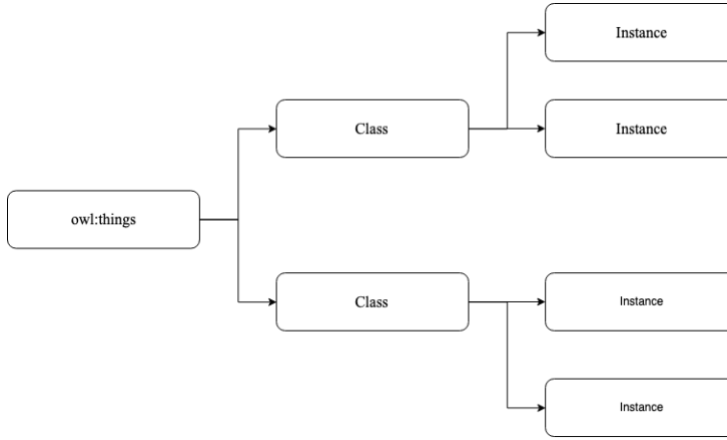
4.4.1 Actor

Class Actor menampung aktor-aktor yang didapatkan dari hasil ekstraksi menggunakan metode *NER*. Aktor-aktor tersebut berbentuk *instances* dalam ontologi.

4.4.2 Kecelakaan lalu lintas

Class Kecelakaan lalu lintas memiliki beberapa kategori kecelakaan lalu lintas dalam bentuk *subclass. Subclass* tersebut akan menampung *instances* keterangan yang sesuai dengan kategorinya.

Rancangan ontology yang akan dibuat seperti pada Gambar 4.4.

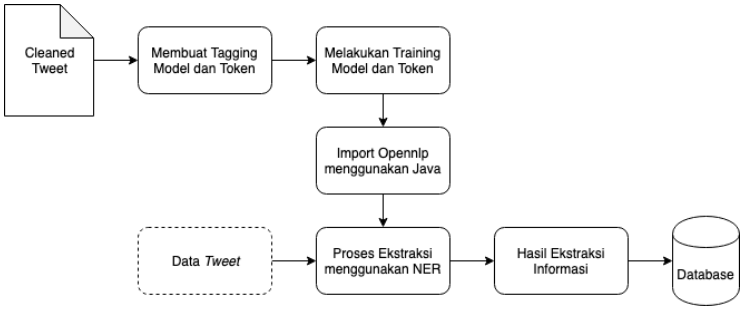


Gambar 4.4 Desain *domain ontology*

4.5 Ekstraksi Data

Pada tahap ekstraksi data ini menjelaskan mengenai tahapan atau alur kegiatan yang dilakukan dalam proses ekstraksi informasi dari setiap *tweet* yang didapatkan menggunakan Opennlp dan NER. Untuk melakukan ekstraksi informasi mengenai kecelakaan lalu lintas dalam sebuah *tweet*, maka

diperlukan beberapa tahap diantaranya adalah *NER Tagging*, *Token Tagging*, *Model Training*, dan *Parser* menggunakan Java. Berikut merupakan Gambar 4.5 merupakan alur pengerjaan pada proses ekstraksi data.



Gambar 4.5 Alur proses ekstraksi data *tweet*

4.6 NER Tagging

Pada proses ini menjelaskan mengenai proses labeling atau nameSpan pada dataset yang digunakan dengan memberikan nameSpan <START: > dan <END> pada entitas yang telah ditentukan. Masing-masing entitas memiliki data dengan nameSpan masing-masing dengan jumlah minimum data yang dibutuhkan untuk setiap entitasnya adalah 15000 kalimat dengan nameSpan. Entitas yang digunakan dalam penelitian ini adalah Actor, Location, Keterangan dan Time. Untuk tagging Actor dan Keterangan dilakukan secara manual. Sedangkan untuk Location dan Time dilakukan secara otomatis menggunakan kode program *python*.

4.6.1 Location Tagging

Sub-tahap ini merupakan bagian dari *NER Tagging*, akan dilakukan pencarian *dataset* daerah Indonesia dari kecamatan, kabupaten/kota hingga provinsi. Selanjutnya adalah proses *replace string* dengan cek per baris *tweet* menggunakan fungsi

str_replace(). *String* lokasi yang ditemukan akan di *replace* dengan *tag* <START:location> *entity_name* <END>.

Berikut pada Gambar 4.6 - Gambar 4.9 merupakan kumpulan lokasi yang akan digunakan pada *tagging* lokasi.

	id	provinsi	p_bsni
0	1	Aceh	ID-AC
1	2	Sumatra Utara	ID-SU
2	3	Sumatra Barat	ID-SB
3	4	Riau	ID-RI
4	5	Jambi	ID-JA
5	6	Sumatra Selatan	ID-SS
6	7	Bengkulu	ID-BE
7	8	Lampung	ID-LA
8	9	Kepulauan Bangka Belitung	ID-BB
9	10	Kepulauan Riau	ID-KR
10	11	Daerah Khusus Ibukota Jakarta	ID-JB
11	12	Jawa Barat	ID-JB
12	13	Jawa Tengah	ID-JT
13	14	Daerah Istimewa Yogyakarta	ID-YO
14	15	Jawa Timur	ID-JI
15	16	Banten	ID-BT

Gambar 4.6 Daftar Provinsi di Indonesia

	id	provinsi_id	kabupaten_kota	ibukota	k_bsni
0	1	1	Kabupaten Aceh Barat	Meulaboh	MBO
1	2	1	Kabupaten Aceh Barat Daya	Blangpidie	BPD
2	3	1	Kabupaten Aceh Besar	Jantho	JTH
3	4	1	Kabupaten Aceh Jaya	Calang	CAG
4	5	1	Kabupaten Aceh Selatan	Tapak Tuan	TTN
5	6	1	Kabupaten Aceh Singkil	Singkil	SKL
6	7	1	Kabupaten Aceh Tamiang	Karang Baru	KRB
7	8	1	Kabupaten Aceh Tengah	Takengon	TKN
8	9	1	Kabupaten Aceh Tenggara	Kutacane	KTN
9	10	1	Kabupaten Aceh Timur	Langsa	LGS
10	11	1	Kabupaten Aceh Utara	Lhoksukon	LSK
11	12	1	Kabupaten Bener Meriah	Simpang Tiga Redelong	STR
12	13	1	Kabupaten Bireuen	Bireuen	BIR
13	14	1	Kabupaten Gayo Lues	Blangkejeren	BKJ
14	15	1	Kabupaten Nagan Raya	Suka Makmue	SKM
15	16	1	Kabupaten Pidie	Sigli	SGI

Gambar 4.8 Daftar Kabupaten dan Kota di Indonesia

	id	kabkot_id	kecamatan
0	1	1	Arongan Lambalek
1	2	1	Bubon
2	3	1	Johan Pahlawan
3	4	1	Kaway XVI
4	5	1	Meureubo
5	6	1	Pante Ceureumen (Pantai Ceuremen)
6	7	1	Panton Reu
7	8	1	Samatiga
8	9	1	Sungai Mas
9	10	1	Woyla
10	11	1	Woyla Barat
11	12	1	Woyla Timur
12	13	2	Babah Rot
13	14	2	Blang Pidie
14	15	2	Jeumpa
15	16	2	Kuala Batee

Gambar 4.7 Daftar Kecamatan di Indonesia

	id	singkatan
0	1	jateng
1	2	jabar
2	3	jatim
3	4	ntb
4	5	sulsel
5	6	sulteng
6	7	sulut
7	8	sulbar
8	9	kaltim
9	10	kalbar
10	11	kalsel
11	12	kalteng
12	13	jakut
13	14	jaktim
14	15	jakbar
15	16	jaksel

Gambar 4.9 Daftar lokasi dengan nama singkatan

4.6.2 Token Tagging

Pada proses ini menjelaskan mengenai proses labeling yang akan digunakan sebagai acuan dalam melakukan proses tokenization. Dataset yang digunakan, pada setiap token, word, phrase yang ingin di token dipisahkan dengan memberikan white space ataupun special syntax seperti <SPLIT> . Dengan ketentuan, *syntax* tersebut digunakan disetiap akhir kalimat ditandai dengan titik atau koma.

4.6.3 Model Training

Pada proses ini menjelaskan mengenai proses training pada dataset yang digunakan guna mendapatkan output file data model (dengan ekstensi .bin / .train). Proses training dilakukan

dengan menggunakan library OpenNLP, TokenizerTrainer pada Command Line.

4.6.4 Parser dengan Opennlp API

Pada proses ini menjelaskan mengenai proses pengembangan *executor* untuk menjalankan proses *tokenizer* dan *NER* secara berurutan. Dengan begitu proses *NER* bisa lebih baik karena input dilakukan *tokenization* terlebih dahulu.

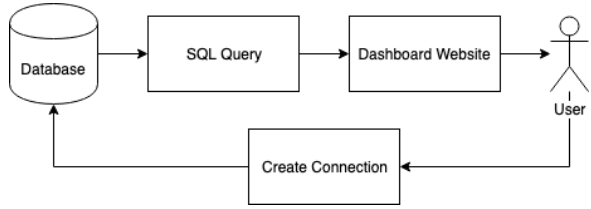
4.7 Pengujian Data

Model yang telah dibuat akan melalui tahap pengujian data untuk mengetahui kualitas data tersebut. Pengujian data akan menggunakan *Precision*, *Recall*, *Accuracy*, dan *F-Measure*. Setelah mendapatkan hasil dari masing-masing model pengujian, maka data akan diuji lagi menggunakan *MAP(mean Average Precision)*. Prosedur pengujian akan membagi data menjadi 2 proporsi bagian, yaitu 80% bagian untuk pelatihan, dan 20% bagian untuk melakukan pengujian. Dalam setiap tes yang telah dilakukan, 20% bagian dipisahkan untuk evaluasi kinerja dan 80% bagian sisanya digunakan untuk pelatihan. Alur pengujian dapat dilihat pada Gambar 4.10.

Gambar 4.10 Alur pengujian model

4.8 Desain Visualisasi Dashboard

Melalui hasil yang telah didapatkan dari hasil ekstraksi informasi yang telah dilakukan maka data siap divisualkan kedalam sebuah model dashboard. Pada penelitian ini menggunakan visualisasi dalam bahasa pemrograman *PHP* dan *Javascript*. Rancangan dashboard yang akan ditampilkan dari hasil ekstraksi informasi mengenai kecelakaan lalu lintas berdasarkan wilayah, periode waktu, dan jumlah kecelakaan yang terjadi. Alur rancangan integrasi antar muka (*user*) dengan model dashboard visualisasi dapat dilihat pada Gambar 4.11.



Gambar 4.11 Alur informasi yang berjalan dalam proses visualisasi

Halaman ini sengaja dikosongkan.

BAB V
IMPLEMENTASI

Bab ini menjelaskan hasil dari implementasi perancangan studi kasus atau hasil dari proses pelaksanaan penelitian. Hasil yang akan dijabarkan adalah hasil eksperimen terhadap data yang digunakan sebagai acuan penelitian. Selain itu, akan dijelaskan juga mengenai tantangan dan kesulitan dalam proses pelaksanaan penelitian.

5.1 Lingkungan Implementasi

Dalam pelaksanaan ekstraksi informasi mengenai kecelakaan lalu lintas di media sosial *Twitter* ini membutuhkan beberapa perangkat yang menunjang jalanannya penelitian ini. Adapun perangkat-perangkat tersebut dapat berupa perangkat keras yang spesifikasinya ditunjukkan dengan Tabel 5.1. Sedangkan untuk perangkat lunak yang digunakan pada proses implementasi model ditunjukkan dalam Tabel 5.2. Penelitian ini juga menggunakan beberapa *library* untuk mendukung proses pengolahan data dan ekstraksi informasi menggunakan *python* dan *java*. *Library* yang digunakan tersebut ditunjukkan dalam Tabel 5.3.

Tabel 5.1 Perangkat keras yang digunakan

Hardware	MacBook Pro
Jenis	Laptop
Spesifikasi	
Processor	2,8 GHz Intel Core i7
Memory	16 GB 2133 MHz LPDDR3
Hardisk	256 GB
Graphics	Intel HD Graphics 630 1536 MB

Tabel 5.2 Perangkat lunak yang digunakan

Software	**Penggunaan**	**Versi**
MacOS	Sistem Operasi	Mojave 10.14.5
Anaconda	Sebagai environment Python 3.7 untuk pemrosesan data	5.7.4
Jupyter Notebook IDE	Menjalankan *pre-proccessing*	5.7.4
Eclipse IDE	Menjalankan *NER*	4.11.0
Ms. Excel 365	Mengolah file CSV	1906
Java	Pemrosesan *NER*	1.8.0_211
Protege	Pembuatan domain ontology	5.2.0
DBMS MySQL	Database hasil *crawling*	10.1.40-MariaDB
Tweetscraper	*Crawler Twitter*	0.1

Tabel 5.3 *Library* yang digunakan

Library	**Penggunaan**	**Versi**
Python		
pandas	Manipulasi dan analisa data	0.24.2
preprocessor	Membersihkan *tweet*	0.1.2
scrapy	*Crawling twitter*	1.6.0
Java		
ArrayList	Membuat *array*	1.8.0_211
OpenNLP	Pemrosesan *NER*	1.9.1
File	Konversi hasil NER ke CSV	1.8.0_211
Maven	Tools untuk mengotomasi build project	4.0.0
Jena	Untuk mengolah ontologi	3.12.0

5.2 Pengambilan Data

Pengambilan data adalah tahap awal dimana data didapatkan dari hasil crawling media sosial *Twitter* yang membahas mengenai kecelakaan lalu lintas. *Crawling* data dilakukan dengan menggunakan *tools TweetScraper* yang menggunakan *library scrapy* tanpa menggunakan *Twitter's APIs*. *TweetScraper* dapat di download di https://github.com/jonbakerfish/TweetScraper. Sebelum menjalankan *crawling* data ada beberapa hal yang perlu diperhatikan. Diantaranya adalah agar data yang didapatkan tersimpan di database mySQL. Untuk melakukan hal tersebut, pada file *setting.py*, aktifkan kode penyimpanan mySQL dengan menghapus tanda pagar (#) pada *SavetoMySQLPipeline* seperti yang terlihat pada Kode Program 5.1.

```
# -*- coding: utf-8 -*-

# !!! # Crawl responsibly by identifying yourself (and
your website/e-mail) on the user-agent
USER_AGENT = 'yasin.awb@gmail.com'

# settings for spiders
BOT NAME = 'TweetScraper'
LOG_LEVEL = 'INFO'
DOWNLOAD_HANDLERS    =    {'s3':    None,}    #    from
http://stackoverflow.com/a/31233576/2297751, TODO

SPIDER_MODULES = ['TweetScraper.spiders']
NEWSPIDER_MODULE = 'TweetScraper.spiders'
ITEM_PIPELINES = {
    #'TweetScraper.pipelines.SaveToFilePipeline':100,
    #'TweetScraper.pipelines.SaveToMongoPipeline':100,
# replace `SaveToFilePipeline` with this to use MongoDB
    'TweetScraper.pipelines.SavetoMySQLPipeline':100, #
replace `SaveToFilePipeline` with this to use MySQL
}
# settings for where to save data on disk
SAVE TWEET PATH = './Data/tweet/Ringsek'
SAVE USER PATH = './Data/user/'
# settings for mongodb
MONGODB_SERVER = "127.0.0.1"
MONGODB PORT = 27017
MONGODB DB = "TweetScraper"          # database name to
save the crawled data
MONGODB_TWEET_COLLECTION = "tweet" # collection name to
save tweets
MONGODB_USER_COLLECTION = "user"   # collection name to
save users
```

Kode Program 5.1 *Setting* penyimpanan pada *tweetscraper*

Untuk menjalankan *TweetScraper* menggunakan terminal dan memasukkan kode program yang dapat dilihat pada Kode Program 5.2. Kode tersebut menjelaskan mengenai cara melakukan *crawling* data di media twitter dengan *keyword* mengenai kecelakaan lalu lintas yang telah ditentukan sebelumnya.

```
scrapy crawl TweetScraper -a query="Kecelakaan
Lalu Lintas"
scrapy crawl TweetScraper -a query="Lakalantas"
scrapy crawl TweetScraper -a query="Kecelakaan
Beruntun"
scrapy crawl TweetScraper -a query="Kecelakaan
Tunggal"
```

Kode Program 5.2 Menjalankan *tweetscraper* pada terminal

Hasil yang didapatkan dari proses ini adalah *database* kumpulan *tweet* yang didapatkan. *Data* yang didapatkan meliputi *ID tweet*, *url*, *datetime*, isi *tweet*, *ID User*, dan *Username*. Proses ini secara urut mendapatkan *tweet* yang baru di *posting* terlebih dahulu.

5.3 Pre-processing Data

Pada tahap ini, pemberishan data yang dilakukan meliputi penghapusan simbol, tagar, mention, *retweet*, *link*, *tweet* yang duplikat, dan *tweet* yang tidak memenuhi kaedah SPOK. Kode program *python* yang dibuat dan dijalankan dengan menggunakan *Jupyter Notebook IDE*. Langkah pertama yang dilakukan adalah melakukan *import* seluruh *library* yang akan digunakan. Pada tahap kali ini hanya menggunakan *library pandas* dan *pre-processor*. Untuk melakukan *import library* tersebut ditunjukkan seperti pada Kode Program 5.3.

```
1.  import pandas as pd
2.  import preprocessor as p
```

Kode Program 5.3 Import *library*

Data *tweet* yang telah didapatkan dari hasil *crawling* tersimpan di dalam database kemudian akan dimuat ke dalam bentuk *csv* untuk diolah ke tahap selanjutnya. Langkah pertama pembuatan kode program yang dilakukan ditunjukkan pada baris ke-4 Kode Program 5.5 yaitu memuat data dengan menggunakan *library pandas*. Selanjutnya Kode Program 5.5 baris ke-6 menunjukkan

review data yang akan digunakan dan baris ke-8 menghitung jumlah *tweet* menggunakan *library pandas.*

```
3.  #load file ascii (menghilangkan kanji, mandar
    in dll.)
4.  with open("D:/S1 Sistem Informasi ITS/Semeste
    r 8/TUGAS AKHIR/kebakaran.csv", encoding='asc
    ii', errors='ignore') as infile:
5.      df = pd.read_csv(infile)
6.  df.head(5)
7.  #cek jumlah tweet
8.  total_rows = len(test)
9.  print(total_rows)
```

Kode Program 5.5 *Meload* data dan menghitung jumlahnya

Selanjutnya, beberapa *Tweet* yang berhasil didapatkan memiliki *link (http)* yang perlu dibersihkan. *Link* yang terdapat disetiap *tweet* memiliki pola yang sama yaitu selalu terletak dibagian paling belakang dari sebuah isi *tweet*. Untuk mempercepat proses pemberishan *link*, kata *http* sampai seterusnya akan dihapus. Untuk melakukan hal tersebut, menggunakan kode program yang terdapat pada Kode Program 5.4 baris ke-11 sampai baris ke-18.

```
10. #menghilangkan http dst
11. without_http = []
12. x = 0
13. while x in range(0, total_rows):
14.     text = str(test.loc[x])
15.     head, sep, tail = text.partition('http')

16.     without_http.append(head)
17.     print(x , 'DEL : ', tail)
18.     x += 1
```

Kode Program 5.4 Menghilangkan *link* pada suatu *tweet*

Proses selanjutnya adalah menggabungkan tanggal *tweet* tersebut diposting dengan isi dari *tweet* tersebut. Tujuan dari penggabungan ini adalah untuk mempermudah proses NER pada tahap selanjutnya karena tanggal dan isi *tweet* menjadi satu kalimat. Kode program yang digunakan pada proses ini ditunjukkan pada Kode Program 5.6 baris ke-20 sampai baris ke-28.

```
19. #menambahkan date
20. with_date = []
21. x = 0
22. for x in range(0, total_rows):
23.     dates =  ' !'+  str(date.loc[x])[:10]
24.     textmain = str(without_http[x])
25.     text = textmain + dates
26.     with_date.append(text)
27.     print(x, text)
28.     x += 1
```

Kode Program 5.6 Menggabungkan tanggal posting pada isi *tweet*

Tahap berikutnya setelah menggabungkan tanggal, maka dilakukan pembersihan tweet dari simbol-simbol dan tanda baca. Untuk setiap tanda baca dan symbol dilakukan mapping terlebih dahulu dengan menyebutkan setiap tanda baca dan simbol yang ingin dibersihkan dapat dilihat pada Kode Program 5.7 baris ke-31. Setelah proses pembersihan tersebut, dilakukan proses menghitung banyak kata dalam suatu kalimat dengan tujuan untuk menghilangkan *tweet* yang tidak memenuhi kaidah SPOK (*tweet* kurang dari 4 kata pada Kode Program 5.7 baris ke-43, 44). Setelah itu, *tweet* dibagi menjadi dua *array* yaitu *array* untuk *short tweet* dan *long tweet*. *Array* dengan short *tweet* ini berisi array yang telah diperpendek menjadi 30 karakter saja dengan tujuan untuk menghapus *tweet* yang duplikat berdasarkan 30 karakter pertama. Pembuatan kode program tersebut ditunjukkan oleh Kode Program 5.7 pada baris ke-47 dan 48.

```
29. #membersihkan text
30. def clear(text):
31.     mapping = [ (' rt ',''),(',',''), ('~','')
    ,('{',''),('}',''),('|',''), ('.',''), (':',''
    ), ('(',''), (')',''), ("'",''), ('?',''), ('m
    arker',''), (';',''), ('#',''),('\\n',''), ('[
    ',''),(']',''),('/',''),('=',''),('$',''),('_'
    ,''),('+',''),('"',''),('*',''),('!',''),('%',
    ''), ('<',''),('>',''),('^','')]
32.     for m, n in mapping:
33.         text = text.replace(m, n)
34.     return text
35.
36. cleaned_text_short = []
37. cleaned_text = []
38.
39. #memasukkan text ke array
40. x = 0
41. while x in range(0, total_rows):
42.     hasil = clear(p.clean(str(with_date[x])))

43.     res = len(hasil.split())
44.     if(res < 5):  #membuang tweet yang tidak m
    emiliki kaidah SPOK
45.         print('BUANG', hasil)
46.     else: #memasukkan tweet ke array
47.         cleaned_text_short.append(hasil[:30])

48.         cleaned_text.append(hasil)
49.         print(x ,hasil[:30])
50.     x += 1
```

Kode Program 5.7 Pembersihan *tweet* dari simbol-simbol

Selanjutnya adalah mengubah *array short tweet* menjadi kedalam bentuk *dataframe* (Kode Program 5.8 baris ke-52). Kemudian menggunakan fungsi *drop_duplicate* untuk menghapus duplikasi *tweet* (Kode Program 5.8 baris ke-57). Setelah dilakukan pembersihan *tweet* yang duplikat, maka dilakukan pengecekan jumlah *tweet*-nya.

```
#mengubah array jadi dataframe
CTdf_short = pd.DataFrame(data=cleaned_text_short, columns=['cleaned_text'])
CTdf_short.head(5)

data = CTdf_short
# dropping ALL duplicte values
data.drop_duplicates(subset ="cleaned_text", keep = 'first', inplace=True)

# displaying data
data['cleaned_text']

#jumlah tweet setelah drop duplicate
len_cleaned_short = len(data['cleaned_text'])

print(len_cleaned_short)
data.head(5)
```

Kode Program 5.8 Menghapus *tweet* yang duplikat

Selanjutnya megubah *dataframe* dari *short tweet* menjadi ke dalam bentuk *array*. Setelah itu dilakukan pencocokan data *long tweet* dengan *short tweet*. Tujuan dari proses ini adalah menghilangkan *tweet* yang duplikat pada *long tweet*. Kode program untuk melakukan proses tersebut dijelaskan pada Kode Program 5.9 baris ke-79 sampai baris ke-90.

```
66. #mengubah dataframe short tweet menjadi array

67. d = data['cleaned_text']
68. d_array= d.values
69. d_array[10]
70. print(len(d_array))
71. #mengubah dataframe long tweet menajadi array

72. c = CTdf['cleaned_text']
73. c_array= c.values
74. c_array[10]
75. #mencocokkan short tweet dengan long tweet
76. final_text = []
77. x = 0
78. y = 0
79. while x in range(0, len_cleaned):
80.     strUtama = str(c_array[x])
81.     for y in range(0, len(d_array)):
82.         strPembanding = str(d_array[y])
83.         if(strUtama[:30] == strPembanding):
84.             final_text.append(strUtama)
85.             d_array[y] = ['tes']
86.             print(y ,strUtama)
87.             y += 1
88.         else:
89.             y += 1
90.     x += 1
```

Kode Program 5.9 Menghilangkan *tweet* yang duplikat pada *array long tweet*

Array final_text kemudian dirubah kedalam bentuk dataframe. Setelah itu melakukan eksport dataframe ke dalam bentuk CSV seperti pada kode program Kode Program 5.10 baris ke-96.

```
#mengubah array menjadi dataframe
final_df = pd.DataFrame(data=final_text, columns=['text'])
final_df.head(5)

#mengubah dataframe menjadi CSV
final_df.to_csv('path_file/nama_file', sep='\t', encoding='utf-8', header=False, index=False)
```

Kode Program 5.10 Mengubah hasil *preprocessing* kedalam format csv

Seacara keseluruhan *preprocessing data* akan mengolah hasil *crawl tweet* yang telah didapatkan kemudian diproses untuk membersihkan *tweet* tersebut dari simbol, tanda baca, *link*, *rt*, dan *tweet* yang duplikat. Untuk mempermudah pemahaman tersebut, Tabel 5.4 merupakan contoh dari *preprocessing tweets* mulai dari *tweet* sebelum diolah dan hasil dari *tweet* yang didapatkan setelah diolah.

Tabel 5.4 Contoh hasil preprocessing

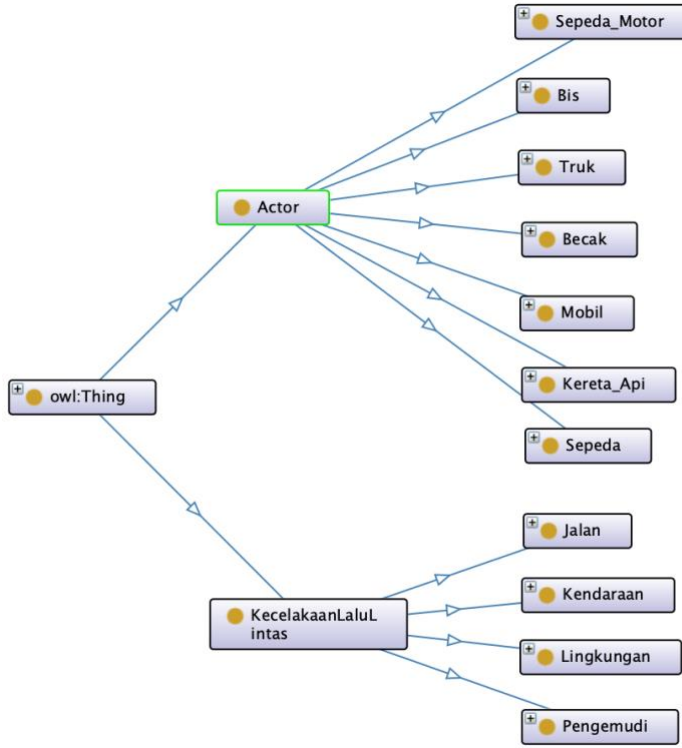
Sebelum *preprocessing*	**Setelah *preprocessing***
Bus menabrak pertengahan trotroar jalan fly over dan terguling ditengah jalan. Kecelakaan tunggal pas mau masuk fly over amplas dari arah asrama haji menuju tj.morawa. menurut supir sendiri dia dalam keadaan mengantuk banting stir,Ä¶ http://medanku.com/bus-menabrak-pertengah	bus menabrak pertengahan trotroar jalan fly over dan terguling ditengah jalan kecelakaan tunggal pas mau masuk fly over amplas dari arah asrama haji menuju tjmorawa menurut supir sendiri dia dalam keadaan mengantuk banting stir 2019-03-03

5.4 Pembuatan Domain Ontology

Pada tahap pembuatan domain ontology akan menggunakan *software Protégé*. Aplikasi ini digunakan untuk membuat rancangan *ontology* yang kemudian digunakan pada tahap selanjutnya. Penyusunan domain *ontology* menggunakan protégé dapat menghasilkan dua *class* yaitu Actor dan KecelakaanLaluLintas. *Class Actor* akan terbagi menjadi beberapa *subclass. Subclass* ini berdasarkan jenis kendaraan seperti Mobil, Sepeda Motor, dan lain-lain. Sedangkan untuk KecelakaanLaluLintas dibagi menjadi 4 *subclass* berdasarkan dari kategori faktor penyebab terjadinya kecelakaan lalu lintas. Lebih detailnya dapat melihat di Gambar 5.1 berikut.

Gambar 5.1 *Model ontology* pada *protégé*

Untuk mempermudah dalam mengetahui pengelompokan data yang dimiliki *ontology* dapat melihat pada Gambar 5.2 yang merupakan gambar *ontograf* dari domain *ontology* yang telah di buat. Sedangkan untuk *instance* di setiap *subclass* sesuai dengan Tabel 5.5 dan Tabel 5.6.



Gambar 5.2 *Model ontology* dalam bentuk *ontograf*

5.4.1 Class Actor

Class actor memiliki 7 *subclass* kategori yang dibuat berdasarkan jenis-jenis *instances* yang ditentukan. Jenis *instance* ini merupakan kata kunci yang terdapat pada sebuah *tweet*.

Berikut adalah daftar *subclass* dan *instances* pada *class actor* pada Tabel 5.5.

Tabel 5.5 *Subclass* dan *instance* pada *class Actor*

Subclass	***Instances***
Mobil	• bajaj • mbl • mbl_elf • mobil • mobil_box • mobil_elf • mobil_travel • pick_up • pikap • sedan • taksi • taxi • travel
Sepeda_Motor	• motor • Pemotor • sepeda_motor • spd_motor • ojek • ojol • ojek_online
Sepeda	• sepeda • sepedah • spd
Bis	• bis • bus • kopaja
Kereta_api	• kereta • kereta_api
Truk	• trek • truck • truk

Becak	• becak • becak_motor

5.4.2 Class Kecelakaan Lalu Lintas

Class Kecelakaan_lalu_lintas memiliki 4 *subclass* kategori yang dibuat berdasarkan jenis-jenis penyebab kecelakaan lalu lintas. Jenis *instance* ini merupakan kata kunci yang terdapat pada sebuah *tweet*.

Berikut adalah daftar *subclass* dan *instances* pada *class* KecelakaanLaluLintas pada Tabel 5.6.

Tabel 5.6 *Subclass* dan *instance* dari *class* KecelakaanLaluLintas

Subclass	***Instances***
Jalan	• jalan_bergelombang • jalan_berlubang • jalan_licin • jalan_rusak • jalan_sempit • persimpangan • tergelincir
Kendaraan	• ban_pecah • lampu_mati • mogok • pecah_ban • rem_blong
Lingkungan	• banjir • berkabut • genangan_air • hujan_deras • penyempitan_lajur
Pengemudi	• kecepatan_tinggi • kehilangan_kendali • kelelahan

	• lalai • mabuk • melamun • melanggar_lampu • melawan_arus • menerobos_lampu • mengantuk • mengebut • ngantuk • ngebut • rem_mendadak • tertidur • tidak_terampil

5.5 Ekstraksi Informasi

Data yang telah dilakukan proses pembersihan sebelumnya kemudian akan diambil informasi-informasi yang terdapat pada data tersebut. Untuk melakukan ekstraksi informasi dari sebuah *tweet,* yang pertama dilakukan adalah membuat model-model yang akan digunakan sesuai dengna kebutuhan data yang ingin di ekstraksi. Model ini terdiri dari model *tagging* dan model token. Setelah membuat model, selanjutnya melakukan *training* model tersebut dan yang terakhir melakukan ekstraksi informasi menggunakan opennlp NER dengan menggunakan maven pada project java.

5.5.1 Model Tagging

Pada tahap pembuatan model, data yang digunakan untuk melakukan proses model tagging adalah data hasil proses preprocessing. Dari data tersebut, setiap kata yang berkaitan dengan kelas yang telah ditentukan dan diberi tag. Model yang akan dibuat ini berfungsi untuk mendeteksi entitas dari suatu data dengan mengecek kata entitas tersebut dan kata di sekitar entitas tersebut. Model-model untuk kelas seperti person, name, etc. sebenarnya sudah disediakan oleh *Apache Opennlp*, namun

model yang tersedia tersebut untuk bahasa selain Bahasa Indonesia. Oleh sebab itu, dibutuhkan pembuatan *custom model* dengan menggunakan modul *NameFinderTrainer* dari *Opennlp*. Untuk membuat custom model ini membutuhkan data minimal 15000 rows. Pembuatan custom model ini menggunakan tag sesuai dengan kelas yang telah ditentukan sebelumnya yaitu Actor, Location, Time, dan Keterangan. Berikut merupakan daftar tag yang akan dilakukan berdasarkan kelasnya pada Tabel 5.7.

Tabel 5.7 *Tagging* entitas *NER* dan penggunaanya

Class	***Tag***	**Penggunaan**
Actor	<START:actor> *entity_name* <END>	Data yang dibutuhkan berupa mobil, motor, sepeda, truk, bus, becak, dan kereta.
Location	<START:location> *entity_name* <END>	Daya yang dibutuhkan adalah data nama provinsi, kabupaten, dan kota.
Time	<START:time> *entity_name* <END>	Data yang dibutuhkan berupa tanggal *tweet* tersebut di *post*.
Keterangan	<START:keterangan> *entity_name* <END>	Data yang dibutuhkan berupa penyebab kecelakaan tersebut merujuk padaTabel 2.6.

Sub-tahap ini merupakan proses pemberian *tag* kepada kata yang termasuk *entitas* yang telah ditentukan dan yang akan dilakukan ekstraksi dari sebuah *tweet*. Berikut pada Tabel 5.8 merupakan contoh dari tahap ini.

Tabel 5.8 Contoh pembuatan *model tagging*

Sebelum *NER Tagging*	**Setelah *NER Tagging***
Tagging Actor	
seorang remaja pengendara sepeda motor meregang nyawa dalam kecelakaan tunggal di kawasan pos ngancar jalan blorangawen turut desa adirejo kecamatan tunjungan blora sepeda motor milik korban mengalami kerusakan pada bagian depan dan diamankan di kant	seorang remaja pengendara <START:actor> sepeda motor <END> meregang nyawa dalam kecelakaan tunggal di kawasan pos ngancar jalan blorangawen turut desa adirejo kecamatan tunjungan blora <START:actor> sepeda motor <END> milik korban mengalami kerusakan pada bagian depan dan diamankan di kant
Tagging Time	
kondisi terbaru jessica korban kecelakaan tunggal di depan griya agung palembang 2019-03-24	kondisi terbaru jessica korban kecelakaan tunggal di depan griya agung palembang <START:time> 2019-03-24 <END>
Tagging Location	
info lalin ruas tol wiyoto wiyono hati hati melintas di ancol ada kecelakaan tunggal dilajur saat ini masih dalam penanganan lalin di grand watu dodol banyuwangi ramai lancar	info lalin ruas tol wiyoto wiyono hati hati melintas di <START:location>ancol<END> ada kecelakaan tunggal dilajur saat ini masih dalam penanganan lalin di grand watu dodol banyuwangi ramai lancar
Tagging Keterangan	
0 wib menjelang cikunir arah cikampek padat kepadatan pertemuan lalin dan kepadatan penyempitan lajur pekerjaan jalan karawang barat km - karawang timur km 600 padat dampak penanganan sisa muatan kendaraan yang mengalami kecelakaan di bahu luarkiri 2019-03-23	0 wib menjelang cikunir arah cikampek padat kepadatan pertemuan lalin dan kepadatan <START:keterangan> penyempitan lajur <END> pekerjaan jalan karawang barat km - karawang timur km 600 padat dampak penanganan sisa muatan kendaraan yang

	mengalami kecelakaan di bahu luarkiri 2019-03-23

5.5.2 Location Tagging

Setelah *data* lokasi/daerah di Indonesia sudah didapatkan, akan dilakukan proses *replace string* lokasi yang terdapat pada *tweet* per baris. *Data* lokasi yang didapatkan berbentuk 4 tabel, yaitu tabel provinsi, tabel kabupaten/kota, tabel kecamatan, dan tabel lokasi singkatan. Berikut merupakan potongan kode program untuk melakukan proses *location tagging* pada Kode Program 5.11.

```
import pandas as pd

#load file ascii (menghilangkan kanji, mandarin dll.)
with open("D:/S1 Sistem Informasi ITS/Semester 8/TUGAS AKHIR/data location/db-wilayah-indonesia-master/csv/tbl_provinsi.csv", encoding='ascii', errors='ignore') as infile:
    prpdf = pd.read_csv(infile, error_bad_lines=False)

prpdf.head(5)

#load file ascii (menghilangkan kanji, mandarin dll.)
with open("D:/S1 Sistem Informasi ITS/Semester 8/TUGAS AKHIR/data location/db-wilayah-indonesia-master/csv/tbl_kabkot.csv", encoding='ascii', errors='ignore') as infile:
    kbktdf = pd.read_csv(infile, error_bad_lines=False)

kbktdf.head(5)

kbktdf['kabupaten_kota'] = kbktdf['kabupaten_kota'].str.replace('Kabupaten ', '')
kbktdf['kabupaten_kota'] = kbktdf['kabupaten_kota'].str.replace('Kota ', '')
kbktdf.head(5)

#load file ascii (menghilangkan kanji, mandarin dll.)
with open("D:/S1 Sistem Informasi ITS/Semester 8/TUGAS AKHIR/data location/db-wilayah-indonesia-master/csv/tbl_kecamatan.csv", encoding='ascii', errors='ignore') as infile:
    kcmtdf = pd.read_csv(infile, error_bad_lines=False)

kcmtdf.head(5)

#load file ascii (menghilangkan kanji, mandarin dll.)
with D:/S1 Sistem Informasi ITS/Semester 8/TUGAS AKHIR/Cleaned/cleaned_location_gabung_banyak.csv", encoding='ascii', errors='ignore') as infile:
    df = pd.read_csv(infile, names = ["text"], error_bad_lines=False)

df.head(5)
```

```
test = df['text']
print(test.loc[1300])
len(test)

p = prpdf['provinsi']
kk = kbktdf['kabupaten_kota']
kc = kcmtdf['kecamatan']
print(p.loc[3])
print(kk.loc[3])
print(kc.loc[3])

#cek jumlah tweet
total_rows_p = len(p)
print(total_rows_p)
total_rows_kk = len(kk)
print(total_rows_kk)
total_rows_kc = len(kc)
print(total_rows_kc)

x = 0
for x in range(0, total_rows_p):
    word = p.loc[x].lower()
    awal = ' {} '.format(word)
    akhir = ' <START:location>{}<END> '.format(word)
    df['text'] = [x.strip().replace(awal, akhir ) for x in df['text']]
    x += 1

print(df['text'].loc[10442])

x = 0
for x in range(0, total_rows_kk):
    word = kk.loc[x].lower()
    awal = ' {} '.format(word)
    akhir = ' <START:location>{}<END> '.format(word)
    df['text'] = [x.strip().replace(awal, akhir ) for x in df['text']]
    x += 1

print(df['text'].loc[13])
```

```
x = 0
for x in range(0, total_rows_kc):
    word = kc.loc[x].lower()
    awal = 'di {} '.format(word)
    awal1 = 'kec {} '.format(word)
    awal2 = 'kecamatan {} '.format(word)
    awal3 = 'daerah {} '.format(word)
    awal4 = 'arah {} '.format(word)
    awal5 = 'wilayah {} '.format(word)
    akhir = 'di <START:location>{}<END> '.forma
t(word)
    akhir1 = 'kec <START:location>{}<END> '.for
mat(word)
    akhir2 = 'kecamatan <START:location>{}<END>
 '.format(word)
    akhir3 = 'daerah <START:location>{}<END> '.
format(word)
    akhir4 = 'arah <START:location>{}<END> '.fo
rmat(word)
    akhir5 = 'wilayah <START:location>{}<END> '
.format(word)
    df['text'] = [x.strip().replace(awal, akhir
 ) for x in df['text']]
    df['text'] = [x.strip().replace(awal1, akhi
r1 ) for x in df['text']]
    df['text'] = [x.strip().replace(awal2, akhi
r2 ) for x in df['text']]
    df['text'] = [x.strip().replace(awal3, akhi
r3 ) for x in df['text']]
    df['text'] = [x.strip().replace(awal4, akhi
r4 ) for x in df['text']]
    df['text'] = [x.strip().replace(awal5, akhi
r5 ) for x in df['text']]
    x += 1

print(df['text'].loc[10769])

#load file ascii (menghilangkan kanji, mandarin
 dll.)
with open("D:/S1 Sistem Informasi ITS/Semester
8/TUGAS AKHIR/data location/db-wilayah-
indonesia-
master/csv/daftar_singkatan.csv", encoding='asc
ii', errors='ignore') as infile:
    sktdf = pd.read_csv(infile,  error_bad_line
s=False)
sktdf
```

```
total_rows_skt = len(sktdf)
print(total_rows_skt)

x = 0
for x in range(0, total_rows_skt):
    word = sktdf['singkatan'].loc[x]
    awal = ' {} '.format(word)
    akhir = ' <START:location>{}<END> '.format(word)
    df['text'] = [x.strip().replace(awal, akhir ) for x in df['text']]
    x += 1

df['text'] = [x.strip().replace('<START:location> <START:location> <START:location> <START:location> ', '<START:location>' ) for x in df['text']]
df['text'] = [x.strip().replace('<END> <END> <END> <END>', '<END>' ) for x in df['text']]
df['text'] = [x.strip().replace('<START:location> <START:location> <START:location>', '<START:location>' ) for x in df['text']]
df['text'] = [x.strip().replace('<END> <END> <END> ', '<END>' ) for x in df['text']]

df['text'] = [x.strip().replace(' <START:location> ', ' <START:location> ' ) for x in df['text']]
df['text'] = [x.strip().replace('<END>', ' <END>' ) for x in df['text']]

#mengubah dataframe menjadi CSV untuk NER <SPLIT>

df['text'].to_csv("D:/S1 Sistem Informasi ITS/Semester 8/TUGAS AKHIR/Cleaned/location/tag_lokasi_v2.csv", sWep='\t', encoding='utf-8', header=False, index=False)
```

Kode Program 5.11 Pembuatan *model location tagging*

5.4.1 Token Tagging

Untuk melakukan token tagging, dapat menggunakan model yang telah disediakan oleh *Apache Opennlp*, namun model token yang tersedia belum ada yang menggubanak Bahasa Indonesia. Oleh sebab itu, perlu membuat model baru yang disebut dengan *custom model*. Untuk melakukan proses *token*

tagging tersebut menggunakan seluruh *tweet* yang telah didapatkan dan *tweet* tersebut telah dibersihkan sebelumnya. *Tweet* yang sudah didapatkan tersebut disetiap akhir kalimatnya ditambahkan *tagging* berupa <SPLIT> dan kemudian diakhiri dengan tanda titik (.). Syarat yang dibutuhkan untuk membuat token tagging ini adalah minimal memiliki 15000 *rows*. *Token tagging* yang telah dibuat akan dirubah menjadi file dengan ekstensi ***.train***. Untuk mempercepat proses *token tagging*, maka proses tagging dilakukan secara otomatis menggunakan kode program *python* dengan menggunakan *library pandas* seperti pada Kode Program 5.12.

```
1.  import pandas as pd
2.
3.  #load file ascii (menghilangkan kanji, mandarin dll.)
4.  with open("/users/yasinawwab/Documents/TA/hasil/cleaned_Kec
    elakaan.csv", encoding='ascii', errors='ignore',) as infile
    :
5.      df = pd.read_csv(infile,  names = ["text"])
6.
7.  df.head(5)
8.
9.  #cek jumlah tweet
10. total_rows = len(test)
11. print(total_rows)
12.
13. #menambahkan <SPLIT>
14. with_split = []
15. x = 0
16. for x in range(0, total_rows):
17.     split =  ' <SPLIT>.'
18.     textmain = str(test.loc[x])
19.     text = textmain + split
20.     with_split.append(text)
21.     print(x, text)
22.     x += 1
23.
24. #cek jumlah tweet
25. total_rows = len(with_split)
26. print(total_rows)
27.
28. #mengubah array menjadi dataframe
29. final_df = pd.DataFrame(data=final_text, columns=['text'])

30. final_df.head(5)
31.
32. #mengubah dataframe menjadi CSV
33. final_df.to_csv('/users/yasinawwab/Documents/TA/hasil/clean
    ed_Tergelincir.csv', sep='\t', encoding='utf-
    8', header=False, index=False)
```

Kode Program 5.12 Pembuatan *token taggng*

5.5.3 Training Model

Setelah melakukan proses *tagging*, selanjutnya yang dilakukan adalah melakukan train untuk dijadikan sebuah model. Model yang dibutuhkan sesuai dengan jumlah kelas yang telah ditentukan sebelumnya. *Training* model ini memiliki dua jenis modulnyaitu *NamFinderTrainer* yang merupakan *training* dari model-model sesuai dengan kelas yang telah ditentukan. Kemudian *tokenizer trainer* yang merupakan *training model* token.

5.5.3.1 NameFinderTrainer

NameFinderTrainer adalah sebuah modul yang dapat digunakan untuk proses *training* model dari *tagging* yang sesuai dengan kelas yang telah ditentukan sebelumnya. Hasil dari modul *NameFinderTrainer* akan menghasilkan *file* model yang nantinya digunakan pada proses NER. Modul tersebut memiliki beberapa argumen-argumen diantaranya seperti pada Kode Program 5.13.

```
$ opennlp TokenNameFinderTrainer
Usage:                                                    opennlp
TokenNameFinderTrainer[.evalita|.ad|.conll03|.bionlp2004|.conl
l02|.muc6|.ontonotes|.brat] \
[-featuregen featuregenFile] [-nameTypes types] [-sequenceCodec
codec] [-factory factoryName] \
[-resources   resourcesDir]   [-type   typeOverride]   [-params
paramsFile] -lang language \
-model modelFile -data sampleData [-encoding charsetName]

Arguments description:
        -featuregen featuregenFile
                The feature generator descriptor file
        -nameTypes types
                name types to use for training
        -sequenceCodec codec
                sequence codec used to code name spans
        -factory factoryName
                A sub-class of TokenNameFinderFactory
        -resources resourcesDir
                The resources directory
        -type typeOverride
                Overrides the type parameter in the provided
samples
        -params paramsFile
                training parameters file.
        -lang language
                language which is being processed.
        -model modelFile
                output model file.
        -data sampleData
                data to be used, usually a file name.
        -encoding charsetName
                encoding for reading and writing text, if absent
the system default is used.
```

Kode Program 5.13 Argumen pada *training model*

Untuk membuat model menggunakan *script* yang dijalankan menggunakan *command prompt.* Seperti yang ditunjukkan pada Kode Program 5.14, model yang akan dihasilkan memiliki ekstensi *.bin* sedangkan *tagging* yang telah dilakukan harus dimasukkan dengan ekstensi *.train*.

```
opennlp          TokenNameFinderTrainer          -model
[nama_model_yang_akan_dihasilkan].bin  -lang  en  -data
[file_tag_manual].train -encoding UTF-8
```

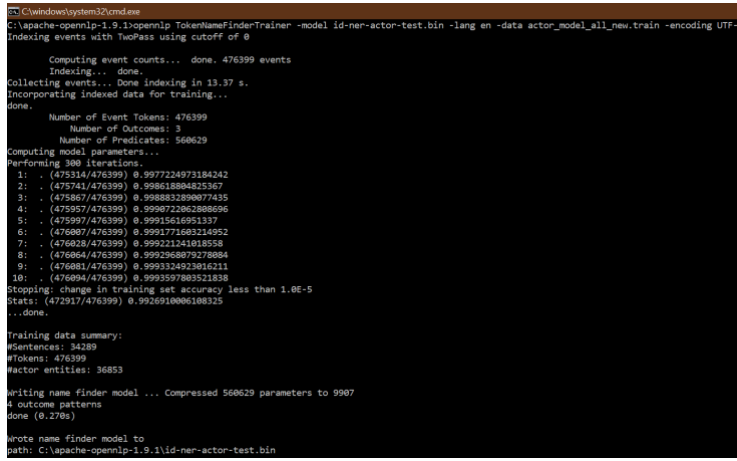
Kode Program 5.14 *Script* untuk melakukan *training model*

Berikut adalah contoh *script* yang digunakan pada *command prompt untuk* membuat *model* pada Gambar 5.3.

```
C:\apache-opennlp-1.9.1>opennlp TokenNameFinderTrainer -model id-ner-actor-test
.bin -lang en -data actor_model_all_new.train -encoding UTF-8
```

Gambar 5.3 Contoh script untuk melakukan *training model*

Setelah script tersebut dijalankan, maka tampilan *console* hasil *training NameFinderTrainer* akan seperti pada Gambar 5.4.



Gambar 5.4 Hasil setelah melakukan *training model*

Hasil tersebut menunjukkan bahwa *output file* dari proses *training*, yaitu model dari *NameFinderTrainer* dengan nama ***id-ner-actor.bin.***

5.5.3.2 Tokenizer Trainer

Tokenizer Trainer adalah sebuah modul yang dapat digunakan untuk proses *training* model dari *tagging* yang digunakan untuk membuat model token. Hasil dari modul *Tokenizer Trainer* akan menghasilkan *file* model yang nantinya digunakan pada proses NER sebagai model dengan token Bahasa Indonesia. Modul tersebut memiliki beberapa argument-argumen diantaranya seperti pada Kode X.

```
$ opennlp TokenizerTrainer
Usage: opennlp TokenizerTrainer[.namefinder|.conllx|.pos]
[-abbDict path] \
                [-alphaNumOpt isAlphaNumOpt] [-params
paramsFile] [-iterations num] \
                [-cutoff num] -model modelFile -lang
language -data sampleData \
                [-encoding charsetName]

Arguments description:
        -abbDict path
                abbreviation dictionary in XML format.
        -alphaNumOpt isAlphaNumOpt
                Optimization flag to skip alpha numeric
tokens for further tokenization
        -params paramsFile
                training parameters file.
        -iterations num
                number of training iterations, ignored if -
params is used.
        -cutoff num
                minimal number of times a feature must be
seen, ignored if -params is used.
        -model modelFile
                output model file.
        -lang language
                language which is being processed.
        -data sampleData
                data to be used, usually a file name.
        -encoding charsetName
                encoding for reading and writing text, if
absent the system default is used.
```

Kode Program 5.15 Argumen pada *tokenizer training*

Untuk membuat model menggunakan *script* yang dijalankan menggunakan *command prompt.* Seperti yang ditunjukkan pada Kode Program 5.16, model yang akan dihasilkan memiliki ekstensi *.bin* sedangkan *tagging* yang telah dilakukan harus dimasukkan dengan ekstensi *.train.*

```
opennlp              TokenizerTrainer              -model
[nama_model_yang_akan_dihasilkan].bin   -alphaNumOpt   -
lang en -data [file_model_token].train -encoding UTF-8
```

Kode Program 5.16 *Script* untuk melakukan *tokenizer training*

Berikut adalah contoh script yang digunakan pada *command prompt* untuk membuat model pada Gambar X.

```
C:\apache-opennlp-1.9.1>opennlp TokenizerTrainer -model id-token-ner-test.bin
-alphaNumOpt 2 -lang en -data token_all.train -encoding UTF-8
```

Gambar 5.5 Contoh *script* yang digunakan untuk melakukan *tokenizer token*

Setelah script tersebut dijalankan, maka tampilan *console* hasil *training NameFinderTrainer* akan seperti pada Gambar 4.6.

```
        Computing event counts...  done. 5290784 events
        Indexing...  done.
Sorting and merging events... done. Reduced 5290784 events to 269270.
Done indexing in 58.81 s.
Incorporating indexed data for training...
done.
        Number of Event Tokens: 269270
            Number of Outcomes: 2
          Number of Predicates: 65240
...done.
Computing model parameters ...
Performing 100 iterations.
  1:  ... loglikelihood=-3667292.0125583564     0.9857816913334583
  2:  ... loglikelihood=-2157243.609481879      0.9868422146887872
  3:  ... loglikelihood=-1540960.2327349496     0.9887453352849029
  4:  ... loglikelihood=-1201077.4887442435     0.9900398504267043
  5:  ... loglikelihood=-984708.0675221032      0.9925096167222098
  6:  ... loglikelihood=-834688.5893590362      0.99459569697043
  7:  ... loglikelihood=-724506.238619718       0.9955422485590038
  8:  ... loglikelihood=-640128.0664170069      0.997319111874535
  9:  ... loglikelihood=-573422.7683536265      0.9973603155978396
```

```
 88:  ... loglikelihood=-62442.932319388325     0.999965978577088
 89:  ... loglikelihood=-61748.56788241982      0.9999661675849931
 90:  ... loglikelihood=-61069.52267056468      0.9999661675849931
 91:  ... loglikelihood=-60405.295024834486     0.9999663565928981
 92:  ... loglikelihood=-59755.404953793906     0.9999663565928981
 93:  ... loglikelihood=-59119.392976310926     0.9999663565928981
 94:  ... loglikelihood=-58496.81903769043      0.9999663565928981
 95:  ... loglikelihood=-57887.26149383594      0.9999665456008032
 96:  ... loglikelihood=-57290.31615844024      0.9999665456008032
 97:  ... loglikelihood=-56705.5954087926       0.9999665456008032
 98:  ... loglikelihood=-56132.72734595378      0.9999665456008032
 99:  ... loglikelihood=-55571.355005523044     0.9999665456008032
100:  ... loglikelihood=-55021.135615458144     0.9999667346087083
Writing tokenizer model ... done (0.424s)

Wrote tokenizer model to
path: C:\apache-opennlp-1.9.1\id-token-ner-test.bin

Execution time: 71.861 seconds
```

Gambar 5.6 Hasil proses *tokenizer training*

Hasil tersebut menunjukkan bahwa *output file* dari proses *training*, yaitu model dari *NameFinderTrainer* dengan nama ***id-token-ner-test.bin.***

5.6 Named Entity Recognition

Model-model yang dihasilkan pada tahap sebelumnya akan menghasilkan 4 *file model named entity* dan 1 *file model token*. Kemudian dilakukan proses *parsing* menggunakan kode program berbasis *Java.* Pada kode *program* tersebut, membuat fungsi yang digunakan untuk memanggil *model* dengan menggunakan *NameFinder API* dan *Tokenizer API* untuk setiap *model* tersebut dibagi kedalam 4 *class* sesuai dengan *model Actor, Time, Location* dan Keterangan. Untuk membuat *class* yang digunakan untuk memanggil *model* menggunakan *dependencies* seperti Kode Program 5.18 baris ke-1 sampai baris ke-8.

```
1.  import opennlp.tools.namefind.NameFinderME;
2.  import opennlp.tools.namefind.TokenNameFinderModel;

3.  import opennlp.tools.tokenize.TokenizerME;
4.  import opennlp.tools.tokenize.TokenizerModel;
5.  import opennlp.tools.util.Span;
6.  import java.io.IOException;
7.  import java.io.InputStream;
8.  import java.util.ArrayList;
```

Kode Program 5.18 Memanggil *dependencies* yang diperlukan

Untuk menyimpan hasil *parsing* yang telah dilakukan dengan menyimpannya didalam *array* yang dapat diakses pada kelas lain (Kode Program 5.17 baris ke-1 sampai baris ke-8). Kemudian membuat fungsi *count word* yang berfungsi untuk

```
1.  public static ArrayList<String> actor = new ArrayLi
    st();
2.
3.      public ArrayList<String> getTweets() {
4.          return actor;
5.      }
6.      public void setTweets(ArrayList<String> actor)
    {
7.          this.actor = actor;
8.      }
9.          static int wordcount(String string)
10.     {
11.       int count=0;
12.
13.         char ch[]= new char[string.length()];

14.         for(int i=0;i<string.length();i++)
15.         {
16.             ch[i]= string.charAt(i);
17.             if( ((i>0)&&(ch[i]!=' ')&&(ch[i-
    1]==' ')) || ((ch[0]!=' ')&&(i==0)) )
18.                 count++;
19.         }
20.         return count;
21.     }
```

Kode Program 5.17 Menyimpan hasil *parsing*

menghitung jumlah *parsing* yang lebih dari 1 kata seperti Kode Program 5.17 baris ke-9 sampai baris ke-21.

Selanjutnya Kode Program 5.19 menjelaskan mengenai cara melakukan *parsing* pada *model actor*.

```
1.  public void findActor(String paragraph) throws IOExcept
    ion {
2.          InputStream inputStream = getClass().getResourc
    eAsStream("/id-ner-actor.bin");
3.          TokenNameFinderModel model = new TokenNameFinde
    rModel(inputStream);
4.          NameFinderME nameFinder = new NameFinderME(mode
    l);
5.          StringBuilder sb = new StringBuilder();
6.
7.          String[] tokens = tokenize(paragraph);
8.
9.          Span nameSpans[] = nameFinder.find(tokens);
10.         int v = 0;
11.         for(Span s: nameSpans) {
12.             for(int i = s.getStart(); i < s.getEnd(); i
    ++ ) {
13.                 sb.append(tokens[i] + " ");
14.             }
15.             int cwords = wordcount(sb.toString());
16.             sb.setLength(0);
17.             for(int i = s.getStart(); i < s.getEnd(); i
    ++ ) {
18.                 if(cwords > 1) {
19.                     if(v == 0) {
20.                         sb.append(tokens[i] + "_");
21.                         v++;
22.                     } else {
23.                         sb.append(tokens[i]);
24.                     }
25.                 } else {
26.                     sb.append(tokens[i]);
27.                 }
28.             }
29.             actor.add(sb.toString());
30.             sb.setLength(0);
31.         }
32.     }
33. public void clearArray() {
34.         actor.clear();}
```

Kode Program 5.19 Cara melakukan *parsing* menggunakan *OpenNLP NER*

Dalam melakukan parsing perlu untuk menggunakan *model token* yang telah dibuat sebelumnya. Untuk menggunakan *model token* tersebut menggunakan fungsi pada Kode Program 5.20.

```
1.  public String[] tokenize(String sentence) throws IO
    Exception{
2.          InputStream inputStreamTokenizer = getClass
    ().getResourceAsStream("/id-token-ner.bin");
3.          TokenizerModel tokenModel = new TokenizerMo
    del(inputStreamTokenizer);
4.          TokenizerME tokenizer = new TokenizerME(tok
    enModel);
5.          return tokenizer.tokenize(sentence);
6.      }
```

Kode Program 5.20 Memanggil *model token*

Kemudian untuk memanggil *ontology* yang telah dibuat sebelumnya. *Ontology* tersebut akan disimpan pada *array* 2D yang berisi kolom pertama *class* dan kolom kedua *instance* dari *ontology* tersebut. Untuk melakukan load *ontology* pada *Java* menggunakan *library Jena*. Untuk menggunakan *library* tersebut tambahkan *dependencies* seperti pada Kode Program 5.21.

```
1.  import org.apache.jena.ontology.Individual;
2.  import org.apache.jena.ontology.OntClass;
3.  import org.apache.jena.ontology.OntDocumentManager;

4.  import org.apache.jena.ontology.OntModel;
5.  import org.apache.jena.rdf.model.Model;
6.  import org.apache.jena.rdf.model.ModelFactory;
7.  import org.apache.jena.util.FileManager;
8.  import org.apache.jena.util.iterator.ExtendedIterat
    or;
```

Kode Program 5.21 Memanggil *dependency jena*

Setelah itu, untuk menjalankan hal tersebut dengan membuat fungsi *load ontology* seperti pada Kode Program 5.22 berikut.

```
1.  private int x;
2.      public static String[][] ontoDomain = new String[10
    0][2];
3.
4.      public static String getOntoDomain(int x, int y) {

5.          return ontoDomain[x][y];
6.      }
7.
8.      static String defaultNameSpace = "http://semanticwe
    b.org/ontologies";
9.      Model schema = null;
10.     OntDocumentManager mgr = new OntDocumentManager();

11.
12.     private void loadontology() throws IOException{
13.         schema = ModelFactory.createOntologyModel();
14.         java.io.InputStream inschema = FileManager.get(
    ).open("/users/yasinawwab/Documents/TA/hasil/fix/lakala
    ntas.owl");
15.         schema.read(inschema, defaultNameSpace);
16.         setX(0);
17.
18.         ExtendedIterator classes = ((OntModel) schema).
    listClasses();
19.         while (classes.hasNext())
20.         {
21.           OntClass thisClass = (OntClass)classes.next()
    ;
22.           String ontoclass = thisClass.toString();
23.           String value = ontoclass.substring(30);
24.           if(!value.equals("KecelakaanLaluLintas")) {
25.               ExtendedIterator instances = thisClass.li
    stInstances();
26.               while (instances.hasNext())
27.               {
28.                 Individual thisInstance = (Individual)
    instances.next();
29.                 String ontoinst = thisInstance.toString
    ();
30.                 ontoDomain[getX()][0] = ontoclass.subst
    ring(30);
31.                 ontoDomain[getX()][1] = ontoinst.substr
    ing(30);
32.                 setX(getX() + 1); }}}}
```

Kode Program 5.22 Melakukan *load ontology*

Untuk menentukan *class* dari *instance* Keterangan yang telah diparsing, maka membuat fungsi untuk menentukan *class* dari sebuah *instance* yang didapatkan seperti pada Kode Program 5.23.

```
1.  public String compareTo(String input) {
2.          String output = null;
3.          for(int i=0; i < getX(); i++) {
4.              if(input.equals(getOntoDomain(i,1))) {

5.                  output = getOntoDomain(i,0);}}
6.          return output;}
```

Kode Program 5.23 Menentukan *class* dari seuatu *instance*

Kode Program 5.24 berikut merupakan cara untuk melakukan load file tweet yang telah dibersihkan.

```
1.  CSVReader bacafile = new CSVReader();
2.          bacafile.baca("/users/yasinawwab/Docu
    ments/TA/hasil/fix/all.csv");
3.          ArrayList<String> tweets = bacafile.g
    etTweets();
```

Kode Program 5.24 Melakukan *load document*

Untuk menjalankan *NameFinder API* dengan memanggil fungsi dari *class NameFinder*. Setiap *tweet* secara satu persatu akan di *parsing* berdasarkan tiap modelnya. Salah satu contoh dalam melakukan *parsing tweet* dengan *model Actor* dapat dilihat pada Kode Program 5.25.

```
1.  int x = 1;
2.          for(String token : tweets)
3.          { actorFinder.findActor(token);
4.  String id = "tweet"+x;
5.  for(int i=0; i < actor.size(); i++ ) {
6.                  dataActor = actor.get(i);
7.                  if(dataActor.length() > 2) {

8.                      System.out.println(":Twee
    t" + x + " :hasActor :" + dataActor + "." + c
    ompareTo(dataActor.toString()));
9.                      insertSQL("actor", "actor
    ", id, dataActor, compareTo(dataActor.toStrin
    g()));
10.         }}
11. dataActor = "";
12. actorFinder.clearArray();
13. x++; }
14. }
```

Kode Program 5.25 Melakukan *parsing* pada salah satu *model*

Tahap terakhir adalah dengan melakukan *insert* ekstraksi data yang telah didapatkan ke dalam database *mySQL*. Untuk melakukan insert tersebut dapat dilihat pada Kode Program 5.26.

```
1.  public void insertSQL(String table, String column, Stri
    ng id, String data, String category) {
2.          try
3.          {
4.            // create a mysql database connection
5.            String myDriver = "com.mysql.jdbc.Driver";
6.            String myUrl = "jdbc:mysql://localhost:3306/l
    akalantas20?useSSL=false";
7.            Class.forName(myDriver);
8.            Connection conn = DriverManager.getConnection
    (myUrl, "root", "toortoor");
9.
10.           // the mysql insert statement
11.           String query;
12.
13.           if(category != null) {
14.                query = " insert into " + table + " (id,
     "+ column +", category)"
15.                     + " values (?, ?, ?)";
16.           } else {
17.                query = " insert into " + table + " (id,
     "+ column +")"
18.                 + " values (?, ?)";
19.           }
20.
21.           // create the mysql insert preparedstatement

22.           PreparedStatement preparedStmt = conn.prepare
    Statement(query);
23.           preparedStmt.setString (1, id);
24.           preparedStmt.setString (2, data);
25.           if(category != null) {
26.               preparedStmt.setString (3, category);
27.           }
28.
29.           // execute the preparedstatement
30.           preparedStmt.execute();
31.
32.           conn.close();
33.         }
34.         catch (Exception e)
35.         {
36.           System.err.println("Got an exception!");
37.           System.err.println(e.getMessage());
38.         }}
```

Kode Program 5.26 Melakukan *insert data* ke *database mySQL*

5.7 Pengujian Model

Dalam melakukan pengujian model, terdapat tahap *split training* dan *test* data. Tujuan pembagian ini adalah untuk memberikan proporsi data *test* dan data *training* secara *random* yang selanjutnya akan digunakan sebagai data pengujian model tersebut.

5.7.1 Split Training data dan Test data

Pada tahap ini, untuk melakukan pengujian *model* maka perlu menggunakan model *NER* yang telah dibuat sebelumnya (*Model Actor*, *Time, Location* dan Keterangan) seperti pada Tabel 5.8. Dari keempat *model* yang telah dibuat tersebut, langkah pertama adalah setiap *model* akan dibagi menjadi dua *data* yaitu *data testing* dan *data training.* Jumlah baris yang dibutuhkan untuk membuat *data testing* adalah 20% dari total baris yang terdapat pada *model*. Sedangkan untuk *data training,* membutuhkan jumlah 80% dari total baris yang terdapat pada *model. Data* dibagi secara acak menggunakan *library scipy* menggunakan kode program *python.*

Untuk melakukan pembagian proporsi data *training* dan *testing*¸akan dilakukan dengan kode program *python* dengan *library pandas* dan *numpy*. Berikut adalah potongan kode program pada Kode Program 5.27.

```
import pandas as pd
import numpy as np

#load file ascii (menghilangkan kanji, mandarin dll
.)
with open("C:/apache-opennlp-
1.9.1/TESTMODEL/train_new_lokasi_carabaru.train", e
ncoding='ascii', errors='ignore') as infile:
    df = pd.read_csv(infile, header=None)

df.head(5)
len(df)

dfTes = pd.DataFrame(np.random.randn(100, 2))
msk = np.random.rand(len(df)) < 0.8
train = df[msk]
test = df[~msk]

len(train)
len(test)
#Save training data
df_train = pd.DataFrame(data=train, columns=None)
df_train.head(5)
df_train.to_csv('C:/apache-opennlp-
1.9.1/TESTMODEL/train_new_lokasi_carabaru_model80.t
rain', sep='\t', encoding='utf-
8', header=False, index=False)
#Save testing data
df_test = pd.DataFrame(data=test, columns=None)
df_test.head(5)
df_test.to_csv('C:/apache-opennlp-
1.9.1/TESTMODEL/train_new_lokasi_carabaru_test20.te
st', sep='\t', encoding='utf-
8', header=False, index=False)
```

Kode Program 5.27 Membagi proporsi data *train* dan *test*

File training dengan proporsi 80% akan diolah menjadi *file model* dengan menggunakan modul *NameFinderTrainer* yang ada pada sub-bab Model Tagging.

5.7.2 Evaluation API OpenNLP TokenNameFinderTrainer

Pada tahap ini bertujuan melakukan pengujian kualitas model dengan yang telah dibuat sebelumnya dengan menggunakan modul *Evaluation* dari *Opennlp TokenNameFinderTrainer*.

Berikut adalah deskripsi argumen yang terdapat dalam modul pada Kode Program 5.28.

```
$ opennlp TokenNameFinderEvaluator
Usage: opennlp
TokenNameFinderEvaluator[.evalita|.ad|.conll03|.bionlp2004|.co
nll02|.muc6|.ontonotes|.brat] [-nameTypes types] -model model
[-misclassified true|false] [-detailedF true|false] [-
reportOutputFile outputFile] -data sampleData [-encoding
charsetName]

Arguments description:
        -nameTypes types
                name types to use for evaluation
        -model model
                the model file to be evaluated.
        -misclassified true|false
                if true will print false negatives and false
positives.
        -detailedF true|false
                if true (default) will print detailed FMeasure
results.
        -reportOutputFile outputFile
                the path of the fine-grained report file.
        -data sampleData
                data to be used, usually a file name.
        -encoding charsetName
                encoding for reading and writing text, if
absent the system default is used.
```

Kode Program 5.28 Argumen pengujian model

Proporsi data model yang akan digunakan sebagai data *training* sebesar 80% sedangkan data yang akan digunakan sebagai data *testing* dengan proporsi 20%. Secara urut ketikkan script berikut pada *command prompt*, seperti pada Kode Program 5.29.

```
$ opennlp TokenNameFinderEvaluator -model
[data-model-80%].bin -data [data-testing-
20%].test -encoding UTF-8
```

Kode Program 5.29 *Script evaluation API TokenNameFinderTrainer*

Berikut adalah potongan *script* pada *command prompt Opennlp* padaGambar 5.7.

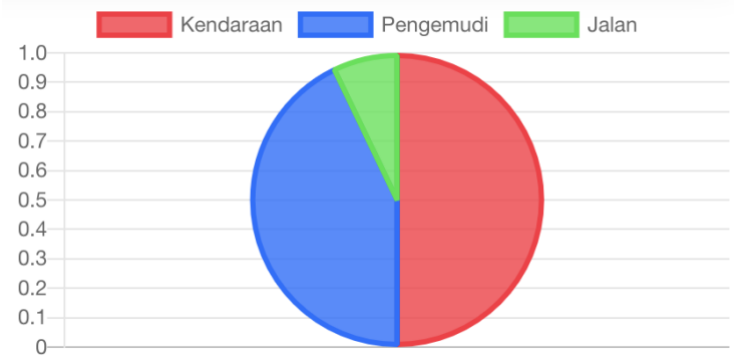
```
C:\apache-opennlp-1.9.1\TESTMODEL>opennlp TokenNameFinderEvaluator -model
 id-ner-location-baru-model80.bin -data train_new_lokasi_carabaru_test20.
test -encoding UTF-8
```

Gambar 5.7 Contoh tampilan *script TokenNameFinderEvaluator* pada *console*

5.8 Desain Visualisasi Dashboard

Visualisasi data dalam penelitian ini merupakan pembuatan dashboard yang dihasilkan dari data *tweet* mengenai kecelakaan lalu lintas yang didapatkan melalui proses *crawling*. *Dashboard* dibuat untuk menunjukkan persebaran dari data yang didapatkan dari kategori masing-masing. Visualisasi data ke dalam *dashboard* dibuat dengan menggunakan bahasa pemrograman *PHP* dengan menggunakan *framework code igniter*. Adapun bentuk visualisasi data menggunakan grafik dan tabel. Grafik ditampilkan dengan menggunakan *bar chart*, *line chart*, dan *pie chart* dengan menggunakan *library chart js* dan visualisasi berupa tabel menggunakan *template bootstrap*. Visualisasi menggunakan *line chart* digunakan untuk menunjukkan jumlah suatu kejadian dibandingkan dengan waktu terjadinya kejadian tersebut. Visualisasi secara spesifik berdasarkan *location* atau *actor* yang dipilih akan menyediakan beberapa tampilan seperti pada Gambar 5.9 yang menjelaskan mengenai proporsi suatu kategori penyebab kecelakaan berdasarkan keseluruhan kategori tersebut.

Gambar 5.8 Statistik menggunakan *line chart*



Gambar 5.9 Proporsi kategori pada kecelakaan lalu lintas

Kemudian menampilkan jumlah kejadian kecelakaan perbulan namun lebih spesifik yaitu berdasarkan pilihan lokasi atau aktor yang terlibat. Contohnya terdapat pada Gambar 5.10 berikut.

Gambar 5.10 Jumlah kejadian kecelakaan dengan parameter

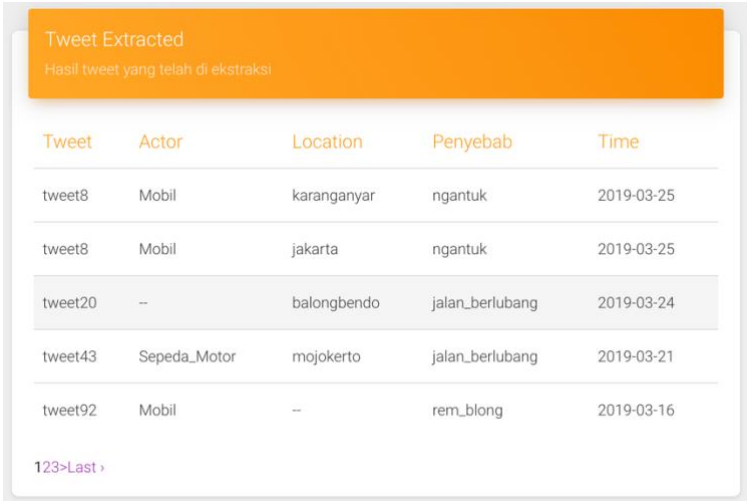
Selanjutnya pada tampilan spesifik terhadap suatu aktor akan menampilkan penyebab kecelakaan yang paling sering terjadi berdasarkan aktor tersebut sesuai dengan Gambar 5.11. Sedangkan Gambar 5.12 menunjukkan tampilan spesifik

terhadap suatu lokasi yang akan menunjukkan aktor apa yang paling sering terlibat kecelakaan lalu lintas di lokasi tersebut.

Gambar 5.11 *Bar chart* mengenai penyebab kecelakaan

Gambar 5.12 *Bar chart* mengenai jumlah *actor* yang terlibat

Visualisasi menggunakan *table* digunakan untuk menampilkan data tweet yang berhasil didapatkan.



Gambar 5.13 Menampilkan keseluruhan hasil ekstraksi informasi

BAB VI
HASIL DAN PEMBAHASAN

Pada bab ini akan menjelaskan terkait analisa dan pengujian yang meliputi tiga hal, yaitu analisa hasil pemodelan, pengujian fungsional dan non fungsional.

6.1 Hasil Pengambilan Data

Dari proses pengambilan data menggunakan *TweetScraper* dengan menuliskan *query* yang ditampilkan pada Kode Program 5.2. Dari proses tersebut secara keseluruhan mendapatkan total 846,311 tweet mengenai kecelakaan lalu lintas dari 4 *keywords* tersebut. Rincian data berupa *keyword* dan jumlah *tweet* yang didapatkan dapat dilihat pada Tabel 6.1.

Tabel 6.1 Jumlah *tweet* yang didapatkan

Keyword	**Jumlah *Tweet***
Kecelakaan Lalu Lintas	274,925
Lakalantas	80,856
Kecelakaan tunggal	162,814
Kecelakaan beruntun	296,274
Total	846,311

6.2 Hasil *Data Preprocessing*

Hasil yang didapatkan pada *preprocessing* data adalah *tweet* yang telah dibersihkan dari simbol-simbol seperti tanda koma, tanda *retweet, link*, dan lain-lain. Setelah penghapusan *tweet* yang tidak memenuhi kaidah SPOK dan menghilangkan *tweet* yang memiliki duplikat. Setelah proses tersebut jumlah *tweet* mengalami pengurangan yang cukup signifikan. Jumlah *tweet* setelah *preprocessing* data dapat dilihat pada Tabel 6.2 berikut.

Tabel 6.2 Jumlah *tweet* setelah *preprocessing*

Keyword	**Jumlah *Tweet***	**Jumlah setelah *preprocessing***
Kecelakaan Lalu Lintas	274,925	63,159
Lakalantas	112,298	44,687
Kecelakaan tunggal	162,814	38,806
Kecelakaan beruntun	296,274	66,719
Total	846,311	213,371

Keseluruhan data tersebut di pisah berdasarkan tahun *posting* pada setiap *tweet*. Data yang didapatkan dari proses yang telah dilakukan sebelumnya paling lama mencapai tahun 2009. Namun karena kebutuhan data yang dibutuhkan pada penelitian kali ini hanya mencakup tahun 2019 dan 2018 saja. Sehingga total data yang didapatkan pada rentang tahun 2018-2019 adalah 25,451 data. Jumlah keseluruhan data pertahun dapat dilihat pada Tabel 6.3 berikut.

Tabel 6.3 Jumlah *tweet* yang didapatkan berdasarkan tahun *posting*

Tahun	**Jumlah *Tweet***
2019	**8000**
2018	**17451**
2017	19762
2016	22377
2015	22874
2014	22697
2013	44357
2012	34605
2011	17695
2010	6612
2009	553
Total	216983

6.3 Hasil Pengujian Model

Pengujian model yang dilakukan terhadap keseluruhan model *named entity*. Hasil pengujian model yang dilakukan pada

model *Actor*, *Location*, *Time*, dan Keterangan. Pengujian menggunakan *query* yang dijelaskan Kode Program X. Hasil keseluruhan pengujian model yang didapatkan dapat dilihat pada Tabel 6.4.

Tabel 6.4 Hasil pengujian model

Model	**Precision**	**Recall**	**F1**
Actor	99,31%	98,56%	98,93%
Location	99,54%	98,37%	98,95%
Time	100,00%	100,00%	100,00%
Keterangan	99,83%	90,58%	94,98%

Model *Actor* mendapatkan nilai-nilai tersebut berdasarkan hasi evaluasi pada 3,397 sampel yang memiliki 3,952 entitas. Dari data entitas tersebut, model berhasil menemukan 3,922. Dari data yang ditemukan, 3,895 berhasil diprediksi dengan benar dan sisanya 27 salah.

Selanjutnya, model *Location* mendapatkan nilai-nilai tersebut berdasarkan hasi evaluasi pada 15,752 sampel yang memiliki 20,022 entitas. Dari data entitas tersebut, model berhasil menemukan 19,787. Dari data yang ditemukan, 19,695 berhasil diprediksi dengan benar dan sisanya 92 salah.

Kemudian, model Time mendapatkan nilai-nilai tersebut berdasarkan hasi evaluasi pada 15,511 sampel yang memiliki 15,511 entitas. Dari data entitas tersebut, model berhasil menemukan 15,511. Dari data yang ditemukan, 15,511 berhasil diprediksi dengan benar dan sisanya 0 salah. Hasil ini dapat mencapai 100% karena keterangan waktu *posting* yang tercantum pada sebuah *tweet* merupakan hasil dari tahap *preprocessing*. Sehingga pola data keterangan waktu yang dimiliki setiap *tweet* sama persis dengan lainnya.

Terakhir model Keterangan mendapatkan nilai-nilai tersebut berdasarkan hasi evaluasi pada 14,073 sampel yang memiliki 14,680 entitas. Dari data entitas tersebut, model berhasil

menemukan 13,319. Dari data yang ditemukan, 13,297 berhasil diprediksi dengan benar dan sisanya 22 salah.

6.4 Hasil Ekstraksi Informasi

Hasil ekstraksi informasi yang telah dilakukan menggunakan *parser OpenNLP API* berupa sekumpulan data dari suatu *tweet* yang menunjukkan *Actor*, *Location*, *Time* dan Keterangan penyebab terjadinya kecelakaan lalu lintas. Proses ekstraksi informasi menyaring *tweet* yang tidak memiliki informasi memadai. Yang dimaksud dengan tidak memiliki informasi memadai yaitu *tweet* hanya memiliki informasi mengenai *Time* saja. Sehingga *tweet* dengan kategori tersebut akan dibuang. Contoh suatu *tweet* yang akan melalui proses ekstraksi informasi dapat dilihat dari Tabel 6.5 berikut.

Table 6.5 Contoh *tweet* yang akan diekstrak informasinya

Tweet
sebuah mobil toyota avanza nopol h fh warna putih terjun ke jurang di kawasan kecamatan pacet kabupaten mojokerto minggu 246 2018 dini hari diduga karena rem blong empat korban selamat dalam kecelakaan tunggal tersebut 2018-06-24
hindari jalan berlubang pemotor tewas terlindas truk gandeng 2019-05-09

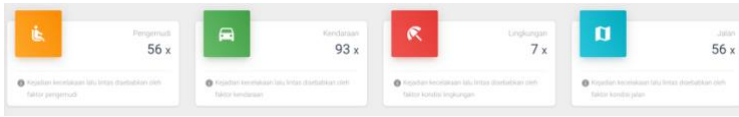
Dari *tweet* diatas, akan dilakukan proses ekstraksi informasi. Hasil ekstraksi akan menampilkan *location, actor*, *time,* dan Keterangan penyebab yang terdapat didalam *tweet* tersebut dan menyimpannya kedalam *database*. Berikut adalah contoh hasil informasi yang didapatkan dari *tweet* tersebut pada .

Table 6.6 Contoh hasil ekstraksi informasi dari suatu *tweet*

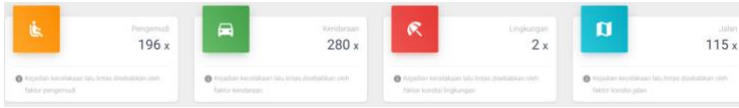
ID	**Actor**	**Location**	**Keterangan**	**Time**
Tweet1	Mobil	Pacet Mojokerto	Kendaraan	2018-06-24
Tweet2	Sepeda_motor Truk	*null*	Jalan	2019-05-09

6.5 Hasil Dashboard Visualisasi

Hasil visualisasi yang dibuat memiliki rentang waktu antara tahun 2018 – 2019. Namun untuk tahun 2019 terhitung hanya sampai dengan bulan ke-6 yaitu bulan juni. Sehingga memiliki dua tampilan yang berbeda tergantung pemilihan tahun yang diinginkan. Hasil visualisasi meliputi statistik keseluruhan mengenai kecelakaan lalu lintas di Indonesia berdasarkan informasi yang berasal dari media sosial *Twitter*. Selanjutnya lebih spesifik berdasarkan *location* yang dipilih atau *actor* yang dipilih.

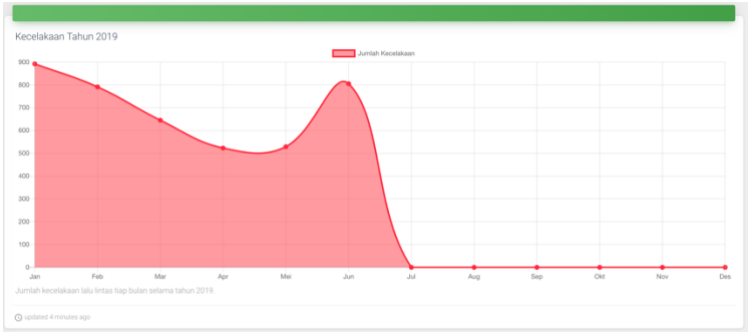


Gambar 6.1 Jumlah kategori kecelakaan lalu lintas pada tahun 2019



Gambar 6.2 Jumlah kategori kecelakaan lalu lintas pada tahun 2018

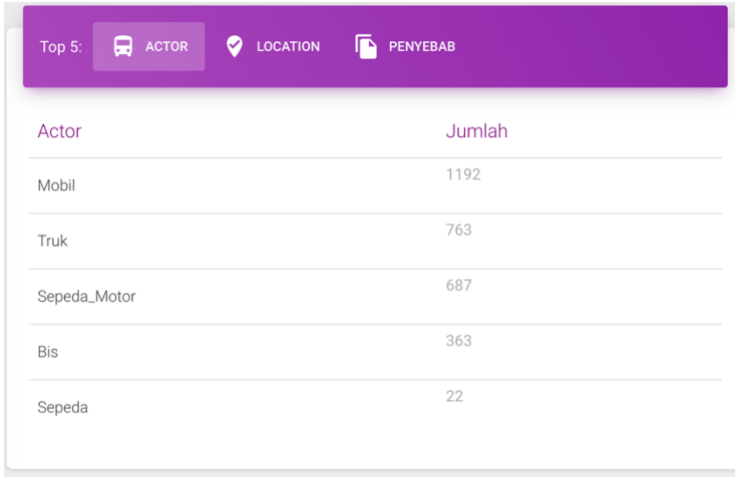
Gambar 6.1 merupakan hasil visualisasi berupa jumlah kategori penyebab kecelakaan lalu lintas yang terjadi selama tahun 2019 terhitung sampai bulan juni. Sedangkan Gambar 6.2 merupakan hasil visualisasi berupa jumlah pebyebab kecelakaan lalu lintas selama tahun 2018. Pada perbandingan tersebut, data tersebut. Menunjukkan bahwa kecenderungan penyebab utama kecelakaan lalu lintas berasal dari kondisi kendaraan yang kurang memadai. Sedangkan faktor lingkungan merupakan faktor yang jarang menjadi penyebab kecelakaan lalu lintas. Faktor tertinggi selanjutnya di tahun 2018 merupakan faktor pengemudi kemudian faktor jalan sebagai penyebab selanjutnya.



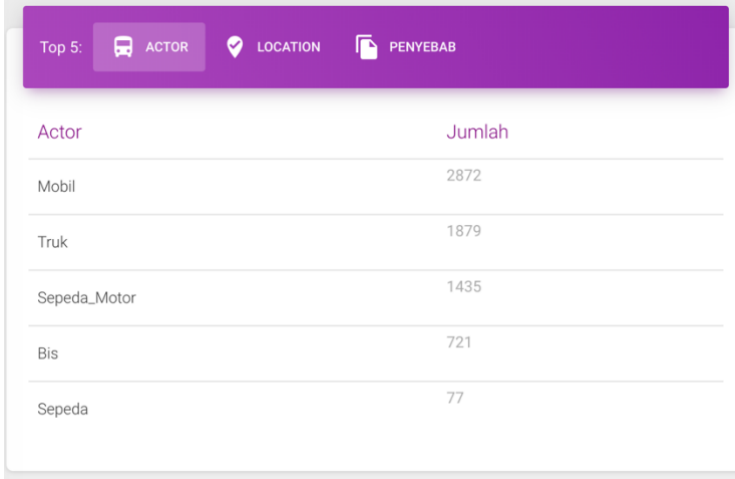
Gambar 6.3 Jumlah kecelakaan lalu lintas pada tahun 2019

Gambar 6.4 Jumlah kecelakaan lalu lintas pada tahun 2018

Gambar 6.3 menunjukkan statistik kejadian kecelakaan lalu lintas disetiap bulan selama tahun 2019 dan Gambar 6.4 di tahun 2018. Dari statistik tersebut tidak terlihat pola tertentu sehingga menunjukkan bahwa waktu tidak mempengaruhi tingkat kejadian kecelakaan lalu lintas. Kemudian pada tahun 2018 menunjukkan bahwa jumlah kejadian kecelakaan lalu lintas tidak menentu dan cenderung tidak stabil.



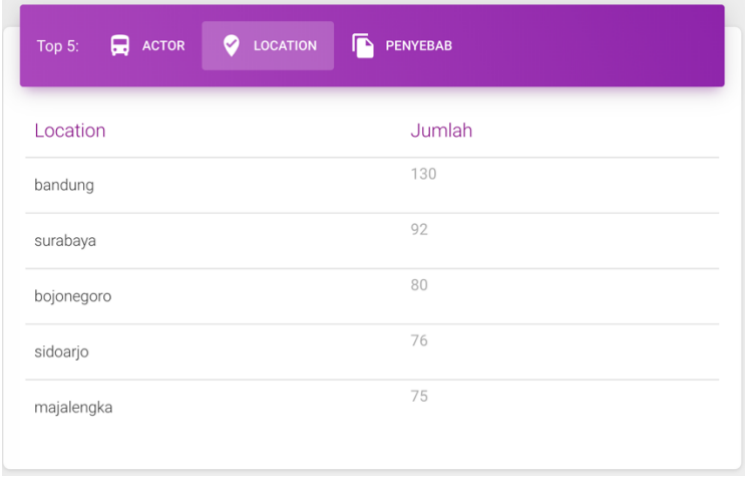
Gambar 6.6 Jumlah *Actor* dalam kecelakaan lalu lintas pada tahun 2019



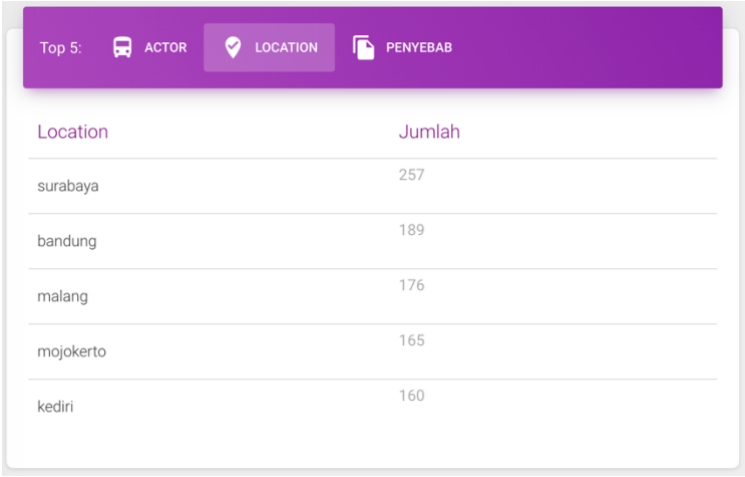
Gambar 6.5 Jumlah *Actor* dalam kecelakaan lalu lintas pada tahun 2018

Gambar 6.6 menunjukkan *Actor* yang sering terlibat dalam sebuah kejadian kecelakaan lalu lintas di tahun 2018 dan Gambar 6.5 di tahun 2019 berdasarkan informasi dari media sosial *Twitter*. Dari statistik tersebut menunjukkan bahwa kejadian kecelakaan lalu lintas yang melibatkan aktor-aktor

besar seperti mobil dan truk lebih sering di*posting* dalam media sosial *Twitter*.



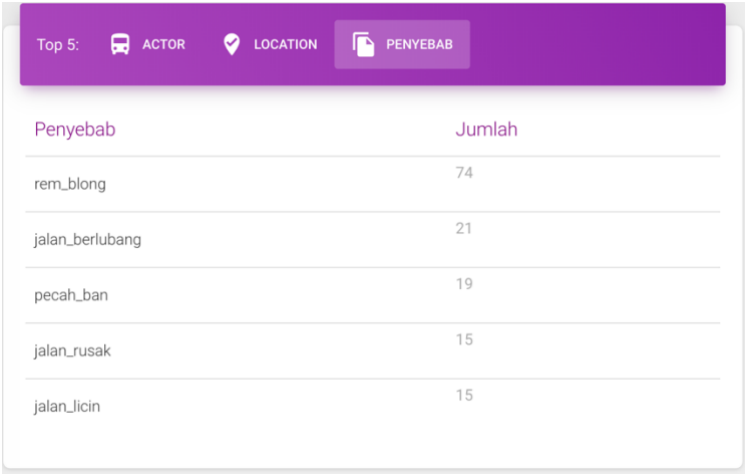
Gambar 6.7 Jumlah *Location* dalam kecelakaan lalu lintas pada tahun 2019



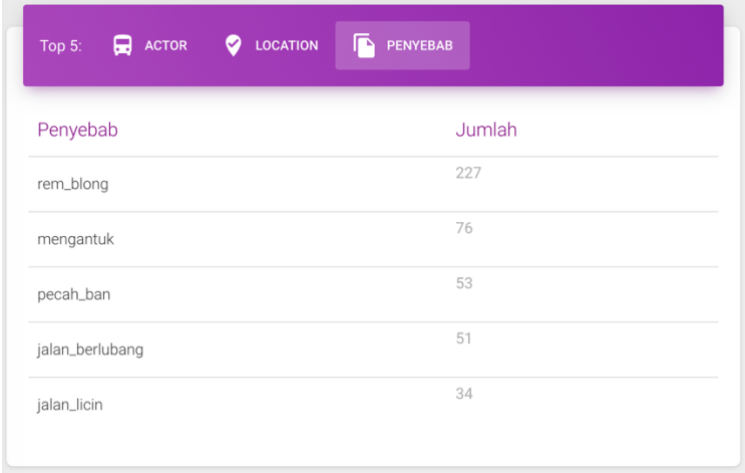
Gambar 6.8 Jumlah *Location* dalam kecelakaan lalu lintas pada tahun 2018

Gambar 6.7 menunjukkan informasi *Location* yang paling banyak tersebar di media sosial Twitter ketika membahas mengenai kecelakaan lalu lintas dalam sebuah kejadian

kecelakaan lalu lintas di tahun 2018 dan Gambar 6.8 di tahun 2019. Dari data 2018-2019 Surabaya dan Bandung menjadi kota yang paling banyak melakukan *update* mengenai kecelakaan lalu lintas di media sosial Twitter.



Gambar 6.10 Jumlah *Penyebab* dalam kecelakaan lalu lintas di tahun 2019



Gambar 6.9 Jumlah *Penyebab* dalam kecelakaan lalu lintas di tahun 2019

Gambar 6.9 menunjukkan informasi penyebab yang paling sering terjadi pada kejadian kecelakaan lalu lintas di tahun 2018 dan Gambar 6.10 di tahun 2019. Rem blong merupakan faktor yang selama tahun 2018 dan 2019 menjadi penyebab utama kecelakaan lalu lintas berdasarkan informasi yang didapatkan dari media sosial *Twitter*. Fakta ini berhubungan dengan kategori kendaraan sebagai penyebab utama kecelakaan lalu lintas.

BAB VII
KESIMPULAN DAN SARAN

Bab kesimpulan dan saran membahas mengenai kesimpulan proses penelitian yang telah dilakukan dan saran yang diusulkan baik untuk perusahaan maupun untuk penelitian serupa di masa mendatang.

7.1 Kesimpulan

Berdasarkan proses-proses yang telah dilakukan dalam pengerjaan tugas akhir yang telah dilakukan, dapat disimpulkan sebagai berikut:

1. Penggunaan *Named Entity Recognition* dengan *OpenNLP API* sebagai metode pembuatan *model* untuk ekstraksi informasi pada *tweet* mampu memberikan akurasi yang cukup tinggi. Untuk model *Actor* memiliki akurasi untuk *precision* sebesar 99,31%, *recall* sebesar 98,56%, dan *F1 score* sebesar 98,93%. Untuk model *Location* memiliki akurasi untuk *precision* sebesar 99,54%, *recall* sebesar 98,37%, dan *F1 score* sebesar 98,95%. Untuk model Keterangan memiliki akurasi untuk *precision* sebesar 99,83%, *recall* sebesar 90,58%, dan *F1 score* sebesar 94,98%. Untuk model *Time* memiliki akurasi untuk *precision* sebesar 100%, *recall* sebesar 100%, dan *F1 score* sebesar 100%.
2. Penelitian ini membuktikan bahwa metode *Named Entity Recognition* dengan memanfaatkan *OpenNLP API* mampu membatu proses ekstraksi informasi suatu *tweet* dengan akurasi model yang cukup tinggi.
3. Penggunaan *ontology* sebagai kerangka klasifikasi keyword yang didapatkan dari proses NER membantu menentukan kategori dari ekstraksi informasi yang didapatkan dengan menggunakan *library jena*.

4. Penggunaan ontology sebagai knowledge model memungkinkan untuk membagikan pengetahuan yang sudah dibentuk pada penelitian ini untuk digunakan dan dikembangkan pada penelitian selanjutnya.
5. Dari pegolahan data yang dihasilkan menunjukkan bahwa faktor penyebab kecelakaan lalu lintas yang paling besar adalah faktor kendaraan dengan masalah utama pada kendaraan disebabkan oleh rem blong.

7.2 Saran

Saran penulis untuk penelitian selanjutnya sebagai berikut:

1. Pembuatan *model training* dan *testing NER modelling* sebaiknya menggunakan *OpenNLP Evaluation API* sehingga memiliki dokumentasi yang lebih baik dan fleksibilitas yang lebih baik.
2. Pembersihan data lebih dioptimalkan pada penyaringan *tweet* yang tidak secara spesifik membahas kejadian kecelakaan lalu lintas sehingga data *tweet* kecelakaan lalu lintas yang didapatkan lebih bagus dan mengurangi *noise* data.
3. Pada penelitian selanjutnya dapat diterapkan pengambilan dan pengolahan data secara *real time* sehingga data dapat mengikuti perkembangan informasi di Twitter. Dengan demikian dapat menjadi salah satu sumber informasi yang lebih bermanfaat.

DAFTAR PUSTAKA

[1] A. Hamzah, "Facebook atau Twitter ? Studi Komparasi Tingkat Kebergunaan Situs Jejaring Sosial," *Semin. Nas. Apl. Teknol. Inf.*, pp. 9–12, 2015.

[2] T. Mihovsky and G. Naydenova, "Comparative study on czech cultivars of red clover (Trifolium Pratense L.) in the conditions of the central northern Bulgaria," *Bulg. J. Agric. Sci.*, vol. 23, no. 5, pp. 739–742, 2017.

[3] "KORLANTAS POLRI - Accident Count." [Online]. Available: http://www.korlantas-irsms.info/graph/accidentData. [Accessed: 14-Feb-2019].

[4] Y. Gu, Z. Qian, and F. Chen, "From Twitter to detector: Real-time traffic incident detection using social media data," *Transp. Res. Part C Emerg. Technol.*, vol. 67, pp. 321–342, 2016.

[5] S. Warpani, *Pengelolaan Lalu Lintas dan Angkutan Jalan*. Bandung, 2002.

[6] A. Josi, L. A. Abdillah, and Suryayusra, "Penerapan teknik web scraping pada mesin pencari artikel ilmiah," no. October, 2014.

[7] A. S. Moghaddam, J. Hosseinkhani, S. Chuprat, H. Taherdoost, and H. B. Baravati, "Proposing a framework for exploration of crime data using web structure and content mining," *Res. J. Appl. Sci. Eng. Technol.*, vol. 6, no. 19, pp. 3617–3624, 2013.

[8] A. Mansouri, L. S. Affendey, and A. Mamat, "Named Entity Recognition Approaches," *Int. J. Comput. Sci. Netw. Secur.*, vol. 8, no. 2, pp. 339–344, 2008.

[9] E. O. Leary, "Developing a Theory-Based Ontology for

" Best Practices " Knowledge Bases," no. 1985, pp. 161–168, 2000.

[10] Mohammad Mustafa Taye, "Understanding Semantic Web and Ontologies: Theory and Applications," *J. Comput.*, vol. 2, no. 6, pp. 182–192, 2010.

[11] T. BERNERS-LEE, J. HENDLER, and L. ORA, *The Semantic Web*, vol. 284, no. 5. 2001.

[12] "World Wide Web Consortium (W3C)." [Online]. Available: https://www.w3.org/. [Accessed: 14-Feb-2019].

[13] "protégé." [Online]. Available: https://protege.stanford.edu/. [Accessed: 14-Feb-2019].

[14] "Getting started with Apache OpenNLP | technobium." [Online]. Available: http://technobium.com/getting-started-with-apache-opennlp/. [Accessed: 28-Mar-2019].

[15] N. Ibrahim, "Pengembangan Aplikasi Semantic Web Untuk Membangun Web yang Lebih Cerdas," *J. Inform.*, vol. 3, pp. 27–40, 2007.

[16] F. C. Albuquerque *et al.*, "A methodology for traffic-related Twitter messages interpretation," *Comput. Ind.*, vol. 78, pp. 57–69, 2016.

[17] L. Derczynski *et al.*, "Analysis of named entity recognition and linking for tweets," *Inf. Process. Manag.*, vol. 51, no. 2, pp. 32–49, 2015.

[18] Y. Munarko, Y. Azhar, M. Balqis, and S. Ekawati, "POS Tagger Tweet Bahasa Indonesia," *KINETIK*, vol. 2, no. 1, pp. 9–16, 2017.

[19] F. Da Costa Albuquerque, M. A. Casanova, J. A. F. De Macedo, M. T. M. De Carvalho, and C. Renso, "A proactive application to monitor truck fleets," *Proc.* -

IEEE Int. Conf. Mob. Data Manag., vol. 1, no. December 2015, pp. 301–304, 2013.

[20] C. D. Manning, P. Raghavan, and S. Hinrich, *Introduction to Information Retrieval*, no. c. 2009.

BIODATA PENULIS

Penulis lahir di Mojokerto pada tanggal 07 Agustus 1996. Penulis merupakan anak keempat dari 6 bersaudara. Pendidikan formal yang ditempuh oleh penulis yaitu SD MI NU Tropodo, SMP Al-Hikmah Surabaya, dan SMA Al-Hikmah Surabaya.

Pada tahun 2015, penulis melanjutkan pendidikan ke jenjang S1 melalui jalur SNMPTN di Institut Teknologi Sepuluh Nopember (ITS) Surabaya yaitu di Departemen Sistem Informasi, Fakultas Teknologi Informasi dan Komunikasi (FTIK), terdaftar dengan NRP 05211540000127. Selama masa perkuliahan, penulis aktif mengikuti beberapa kegiatan kemahasiswaan, antara lain sebagai Pengurus Himpunan Mahasiswa Sistem Informasi (HMSI ITS). Selain itu, dalam hal akademik Penulis juga menjadi juara 1 lomba Pagelaran Mahasiswa TIK 11 (GEMASTIK 11) pada tahun 2018.

Pada tahun ke-4 perkuliahan, penulis tertarik pada bidang *Information Extraction* dan mengambil bidang minat laboratorium Akuisisi Data dan Diseminasi Informasi (ADDI) dan menyelesaikan masa studi selama 8 semester. Penulis dapat dihubungi melalui email: yasinaw27@gmail.com.