Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR - KS141501

EKSTRAKSI INFORMASI MEDIA SOSIAL TWITTER MENGENAI GANGGUAN KEAMANAN MENGGUNAKAN PENDEKATAN ONTOLOGY-BASED INFORMATION EXTRACTION (STUDI KASUS: PT. PERTAMINA (PERSERO))

INFORMATION EXTRACTION OF TWITTER SOCIAL MEDIA ABOUT SECURITY DISRUPTIONS USING ONTOLOGY-BASED INFORMATION EXTRACTION APPROACH (CASE STUDY: PT. PERTAMINA (PERSERO))

BENEDICT TIMOTIUS CHRISTIAN
05211540000055

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 GANGGUAN KEAMANAN MENGGUNAKAN PENDEKATAN ONTOLOGY-BASED INFORMATION EXTRACTION (STUDI KASUS: PT. PERTAMINA (PERSERO))

BENEDICT TIMOTIUS CHRISTIAN
05211540000055

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 SECURITY DISRUPTIONS USING ONTOLOGY-BASED INFORMATION EXTRACTION APPROACH (CASE STUDY: PT. PERTAMINA (PERSERO))

BENEDICT TIMOTIUS CHRISTIAN
05211540000055

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 GANGGUAN KEAMANAN MENGGUNAKAN PENDEKATAN ONTOLOGY-BASED INFORMATION EXTRACTION (STUDI KASUS: PT. PERTAMINA (PERSERO))

TUGAS AKHIR

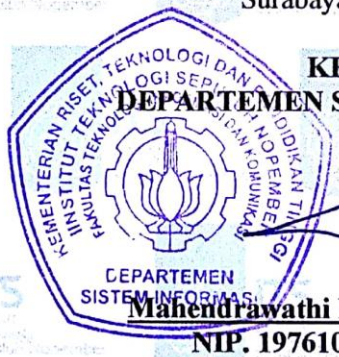
Disusun untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Departemen Sistem Informasi
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh :

BENEDICT TIMOTIUS CHRISTIAN
NRP. 05211540000055

Surabaya, Juli 2019

KEPALA
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 GANGGUAN KEAMANAN MENGGUNAKAN PENDEKATAN ONTOLOGY-BASED INFORMATION EXTRACTION (STUDI KASUS: PT. PERTAMINA (PERSERO))

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 :

BENEDICT TIMOTIUS CHRISTIAN
NRP. 05211540000055

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)**

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

Halaman ini sengaja dikosongkan.

EKSTRAKSI INFORMASI MEDIA SOSIAL TWITTER MENGENAI GANGGUAN KEAMANAN MENGGUNAKAN PENDEKATAN ONTOLOGY-BASED INFORMATION EXTRACTION (STUDI KASUS: PT. PERTAMINA (PERSERO))

Nama Mahasiswa	**: Benedict Timotius Christian**
NRP	**: 05211540000055**
Departemen	**: Sistem Informasi FTIK-ITS**
Pembimbing I	**: Nur Aini Rakhmawati, S.Kom., M.Sc. Eng., Ph.D**

ABSTRAK

Pemberitaan pada masa kini sudah beralih dari media massa konvensional menuju media teknologi informasi seperti media sosial Twitter, di Indonesia jumlah pengguna Twitter mencapai sekitar 49% dari total 130 juta pengguna media sosial di Indonesia. Namun dari pemberitaan yang beredar di media sosial, masih belum pemetaan atau pengolahan data tersebut. Oleh karena itu penelitian ini bertujuan untuk memetakan dan mengolah tweet pemberitaan, khususnya mengenai kasus kejahatan atau gangguan keamanan untuk membuat informasi yang bermanfaat dan membantu memberi masukan kepada PT. Pertamina (Persero). Metodologi yang digunakan dalam penelitian adalah Ontologi untuk ekstraksi informasi pada data yang sudah di crawling berdasarkan kategori yang sudah ditentukan dan Named-Entity Recognition. Metode Named-Entity Recognition digunakan untuk mengkategorikan data per tweet ke dalam kategori seperti aktor, lokasi, keterangan, time. Hasil yang didapatkan adalah metode Named Entity Recognition dalam pembuatan model dapat menghasilkan ekstraksi informasi yang cukup akurat, serta penggunaan Ontologi mampu mengkategorikan tipe kejahatan/gangguan keamanan. Nilai akurasi yang didapatkan oleh model aktor sebesar 90.65% untuk precision, 90.82% untuk recall, 90.74%

untuk f1 score. Nilai akurasi yang didapatkan oleh model lokasi sebesar 99.54% untuk precision, 98.37% untuk recall, 98.95% untuk f1 score. Nilai akurasi yang didapatkan oleh model keterangan sebesar 95.86% untuk precision, 99.75% untuk recall, 97.77% untuk f1 score. Nilai akurasi yang didapatkan oleh model time sebesar 99.99% untuk precision, 100% untuk recall, 100% untuk f1 score.

Kata kunci: Semantic Web, Ontology, NER, Twitter, kejahatan, gangguan keamanan

INFORMATION EXTRACTION OF TWITTER SOCIAL MEDIA ABOUT SECURITY DISRUPTIONS USING ONTOLOGY-BASED INFORMATION EXTRACTION APPROACH (CASE STUDY: PT. PERTAMINA (PERSERO))

Name	**: Benedict Timotius Christian**
NRP	**: 05211540000055**
Department	**: Information System FTIK-ITS**
Supervisor	**: Nur Aini Rakhmawati, S.Kom., M.Sc. Eng., Ph.D**

ABSTRACT

The mass media nowadays has been switched from conventional mass media to technology-based media such as Twitter. The number of Twitter users in Indonesia reaches around 49% of the total 130 million social media users in Indonesia. But from the news circulating on social media, it still hasn't mapped or processed the data. Therefore, this study aims to map and process news tweets, especially regarding crime or security disturbances cases to make useful information and help provide input to PT. Pertamina (Persero). The methodology used in research is Ontology for extracting data on crawled data based on predetermined categories and Named-Entity Recognition. The Named-Entity Recognition method is used to categorize data per tweet into categories such as actor, location, causes, time. The Named-Entity Recognition method is used to categorize data per tweet into categories such as actor, location, description, time. The results obtained are that the Named Entity Recognition method in modeling can produce fairly accurate information extraction, and the use of Ontology is able to categorize the types of crime / security disturbances. The value of accuracy obtained by the actor model is 90.65% for precision, 90.82% for recall, 90.74% for score f1. The accuracy value obtained by the location model is 99.54% for

precision, 98.37% for recall, 98.95% for score f1. The causes value obtained by the answer model is 95.86% for precision, 99.75% for recall, 97.77% for score f1. The value obtained by the time model is 99.99% for precision, 100% for recall, 100% for score f1.

Keywords: Semantic Web, Ontology, NER, Twitter, crime, security disturbances

KATA PENGANTAR

Dengan mengucapkan rasa syukur kepada Tuhan Yang Maha Pengasih dan Maha Penyayang atas izin-Nya penulis dapat menyelesaikan buku yang sederhana ini dengan judul Ekstraksi Informasi Media Sosial Twitter Mengenai Gangguan Keamanan Menggunakan Pendekatan *Ontology-Based Information Extraction (Studi Kasus: PT. Pertamina (Persero))*. Dalam penyelesaian Tugas Akhir ini, penulis diiringi oleh pihak-pihak yang selalu memberi dukungan, saran, dan doa sehingga penelitian berlangsung dengan lancar. Secara khusus penulis mengucapkan terima kasih dari lubuk hati terdalam kepada:

1. Ibu Nur Aini Rakhmawati, S.Kom., M.Sc. Eng., Ph.D selaku Dosen Pembimbing, atas waktu dan kesabaran yang telah diberikan dalam membantu Penulis dengan menyediakan berbagai pemikiran dan petunjuk untuk memperbaiki kekurangan dalam penyelesaian Tugas Akhir ini.
2. Orang tua dan keluarga penulis, yang tiada hentinya mendoakan dan memberikan dukungan kepada penulis sehingga penulis mampu menyelesaikan pendidikan S1 dengan baik.
3. Bapak Faisal Johan Atletiko, S.Kom, M.Kom, dan Irmasari Hafidz, S.Kom, M.Sc, selaku Dosen Penguji yang telah meluangkan waktu untuk menguji dan memberikan masukan yang berharga bagi Penulis.
4. Ibu Mahendrawathi Er, S.T., M.Sc., Ph.D. selaku Kepala Departemen Sistem Informasi ITS, Bapak Nisfu Asrul Sani, S.Kom, M.Sc selaku Kepala Prodi S1 Sistem Informasi ITS serta seluruh dosen pengajar beserta staf dan karyawan di Jurusan Sistem Informasi.
5. Teman-teman apartemen Ex-dukitty yang telah memberikan hari-hari yang cerah maupun muram dalam kehidupan kampus selama 4 tahun, terutama Yasin Ayub yang menjadi partner penulis dalam pengerjaan tugas akhir.

6. Persekutan Jokopi yang selalu memberikan kumpul malam yang penuh makna (Faiq, Farchan, Farhan, Ardo, Rahadhi, Ijul, Oky, Azzam, Dito).
7. Teman per-koriyaan yang sudah meracuni penulis sehingga tenggelam ke dalam dunia koriya (Azzam, Dito, Ijul. Farchan).
8. Teman-teman Sistem Informasi 2015 (LANN15TER) yang senantiasa menemani dan memberi motivasi bagi penulis selama perkuliahan.
9. Pihak lainnya yang berkontribusi dalam tugas akhir yang belum dapat penulis sebutkan satu per satu.

Penyusunan tugas akhir ini masih jauh dari kata sempurna, untuk itu penulis menerima segala kritik dan saran yang membangun sebagai upaya menjadi lebih baik lagi ke depannya. Semoga buku tugas akhir ini dapat memberikan manfaat untuk pembaca.

Surabaya, Juli 2019

Penulis

DAFTAR ISI

Halaman ini sengaja dikosongkan.

DAFTAR GAMBAR

DAFTAR TABEL

Halaman ini sengaja dikosongkan.

DAFTAR KODE PROGRAM

BAB I
PENDAHULUAN

Bab ini akan menjelaskan tentang pendahuluan pengerjaan tugas akhir yang meliputi latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, dan manfaat yang akan diperoleh dari penelitian tugas akhir ini.

1.1 Latar Belakang

Saat ini gaya hidup masyarakat membuat media informasi mengalami perubahan dari yang sebelumnya menggunakan media massa *mainstream* seperti berita di TV menuju era media informasi berbasis teknologi informasi. Hal ini terjadi karena media informasi berbasis teknologi menawarkan produktifitas, efisiensi, kecepatan, dan tanpa batas ruang dibandingkan dengan media massa *mainstream* [1]. Dengan berkembang pesatnya internet, pengguna internet di Indonesia sangat berkembang, yaitu sekitar 143.26 juta jiwa persurvey pada tahun 2017 menurut APJII dan jumlah pengguna media sosial di Indonesia pada tahun 2018 adalah sekitar 130 juta, dengan penetrasi sebesar 49% dibanding jumlah penduduk Indonesia menurut surver yang dilakukan oleh *We Are Sosial* [2][3].
Media informasi berbasis teknologi informasi saat ini banyak digunakan sebagai sarana pemberitaan mengenai tindakan atau kasus kejahatan, salah satunya adalah media sosial *Twitter* dengan jumlah pengguna aktif sekitar 40% dari seluruh pengguna media sosial di Indonesia per Januari 2019 menurut stastistik yang dilakukan oleh *StatCounter* [4]. Dengan jumlah pengguna aktif yang banyak, jumlah *tweet* mengenai pemberitaan juga banyak, salah satunya mengenai kasus kejahatan. Pada tahun 2017 jumlah kejadian kasus kejahatan di Indonesia mencapai 336.197 kasus, dengan jumlah kasus kejahatan yang dapat diselesaikan hanya sebesar 212.058 kasus

menurut Statistik Kriminal yang dilakukan oleh Badan Pusat Statistik Indonesia [5].

Pengguna media sosial, khususnya *Twitter* menghasilkan data *tweet* dalam jumlah yang sangat besar setiap harinya. Namun banyaknya data ini tidak berpengaruh apabila dari semua data tersebut tidak diolah menjadi informasi yang bermanfaat, termasuk *tweet* mengenai kasus kejahatan yang terjadi pada aset-aset vital PT. Pertamina (Persero). Sehingga penelitian ini akan membuat sistem yang dapat mengolah data media sosial *Twitter* untuk memfilter *tweet* yang berhubungan dengan kasus kejahatan dengan membuat ontologi untuk memilah data yang berhubungan dengan kasus kejahatan serta menggunakan algoritma *Named Entity Recognition* yang mampu mengelompokan *tweet* berdasarkan entitasnya (*named entity*).

Diharapkan dengan penelitian ini dilaksanakan, akan membantu mengurangi jumlah kejahatan dan mempercepat deteksi kejadian kasus kejahatan agar mudah ditangani, sehingga dapat mengurangi jumlah kejadian kasus kejahatan di aset-aset vital PT. Pertamina (Persero).

1.2 Rumusan Masalah

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

1. Bagaimana desain domain ontologi yang dapat memilah data dalam pengambilan informasi mengenai kejahatan?
2. Bagaimana cara membuat model menggunakan metode *Named Entity Recognition* pada ekstraksi informasi kecelakaan lalu lintas?
3. Bagaimana cara melakukan visualisasi data jenis kejahatan ke dalam suatu dashboard?

1.3 Batasan Permasalahan

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

1. Lingkup tugas akhir yang dikerjakan meliputi seluruh daerah di Indonesia dengan kecamatan sebagai tingkat daerah terkecil yang digunakan.
2. Data yang digunakan berasal dari media sosial, dan berfokus pada *Twitter*.
3. Pembagian jenis kejahatan berdasarkan kelompoknya menurut PT. Pertamina (Persero).
4. Pembuatan ontologi dengan metode *NER* berfungsi untuk memetakan kasus kejahatan di Indonesia dan menggunakan dashboard untuk menampilkan hasil pemetaan.

1.4 Tujuan

Berdasarkan hasil perumusan masalah dan batasan masalah yang telah disebutkan sebelumnya, maka tujuan yang dicapai dari tugas akhir ini adalah untuk membantu pemetaan kategori kejahatan di Indonesia dengan menggunakan data yang didapatkan dari media sosial *Twitter*, dengan harapan dapat membantu PT. Pertamina (Persero) mendapatkan masukan untuk pengembangan aset-aset vital milik Pertamina.

1.5 Manfaat

Manfaat yang diharapkan dapat diperoleh dari tugas akhir ini adalah:

1. Mengetahui lokasi dan jenis kejadian kejahatan, serta frekuensinya di aset-aset vital PT. Pertamina (Persero) untuk menjadi masukan bagi pengembangan ke depannya.
2. Memberikan pengetahuan kepada mahasiswa, khususnya untuk mahasiswa jurusan terkait Sistem Informasi untuk mempelajari metode pemetaan informasi dengan ontologi.

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 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 Pemrograman Berbasis Web, Teknologi Web dan Pemrograman Bahasa Alami.

BAB II
TINJAUAN PUSTAKA

Bab tinjauan pustaka terdiri dari landasan-landasan yang akan digunakan dalam penelitian tugas akhir ini, mencakup penelitian-penelitian sebelumnya, kajian pustaka, dan metode yang digunakan selama pengerjaan.

2.1 Penelitian Sebelumnya

Terdapat beberapa penelitian yang memiliki topik yang hampir serupa dengan penelitian ini, diantaranya akan dijelaskan pada Tabel 2.1-Tabel 2.5.

Tabel 2.1 Literatur 1

Judul	*An Event Ontology Construction Approach to Web Crime Mining*
Penulis	Li Cunhua, Hu Yun, Zhong Zhaoman
Ringkasan	Penulis menggunakan metode *event ontology based cyber crime* untuk mendemonstrasikan bagaimana hal tersebut digunakan untuk menjabarkan *cyber crimes* dalam berbagai *level of event, relation,* dan *event class.* Teknik seperti SVM untuk klasifikasi teks juga digunakan dalam penyusunan *prototype* sistem. Penulis membandingkan hasil *web crime mining* antara metode 1 yaitu *event ontology* dan *SVM-based,* dengan metode 2 yaitu *SVM-based* saja. Hasil yang didapatkan menunjukkan bahwa metode 1 lebih efektif dan memberikan hasil lebih baik dalam *web crime mining.*

Berdasarkan penjelasan pada Tabel 2.1 atas mengenai penelitian terkait yang berjudul "*An Event Ontology Construction*

Approach To Web Crime Mining" dapat diketahui perbedaan dengan tugas akhir yang dikerjakan, yaitu:

1. Penulis menggunakan metode SVM untuk klasifikasi teks, sedangkan dalam penulis dalam pengerjaan tugas akhir ini tidak menggunakan metode SVM.

Tabel 2.2 Literatur 2

Judul	*Spatiotemporal and semantic information extraction from Web news reports about natural hazards*
Penulis	Wei Wang, Kathleen Stewart
Ringkasan	Penulis melakukan ekstraksi informasi dari berbagai web berita mengenai bencana alam untuk membuat visualisasi dari data *spatiotemporal*. Metode yang digunakan penulis adalah ekstraksi website berita, lalu melakukan *preprocessing* data menggunakan aplikasi *General Architecture for Text Engineering* (GATE), kemudian membuat ontologinya menggunakan aplikasi *NeON ontology editor* dan membuat visualisasi dari data *geocode/spatiotemporal* menggunakan peta digital.

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 pengambilan informasi berdasarkan laporan berita pada website mengenai kejadian alam yang berbahaya dan membuat visualisasi pada peta digital, sedangkan pada tugas akhir yang diusulkan membahas mengenai pemetaan kejahatan di wilayah negara Indonesia dengan metode *NER*,
2. Penelitian sebelumnya menggunakan aplikasi *NeON ontology editor* dan *General Architecture for Text Engineering* (GATE) untuk menunjang pengolahan data,

sedangkan pada pengerjaan tugas akhir yang diusulkan menggunakan bahasa pemrograman *python* dan aplikasi *Protégé* sebagai editor ontologi.

Tabel 2.3 Literatur 3

Judul	*Spatial eigenvector filtering for spatiotemporal crime mapping and spatial crime analysis*
Penulis	Marco Helbich, Jamal Jokar Arsanjani
Ringkasan	Penelitian ini menggunakan metode *Eigenvector Spatial Filtering* (ESF) untuk memetakan kejahatan dan analisis kejahatan spasial. Dalam penelitian ini penulis mendemonstrasikan bagaimana metode ESF digunakan dalam kriminologi untuk menujukkan kesalahan model validasi, seperti *residual spatial autocorrelation*, menggunakan dataset kejahatan *non-violent* untuk wilayah metropolitan Houston, Texas, selama periode 2005-2010. Hasilnya menunjukkan bahwa geografi lokal dan regional secara signifikan berkontribusi pada penjelasan pola kejahatan. Selain itu, *common space-time eigenvectors* yang dipilih pada basis tahunan menunjukkan pola *spatiotemporal* yang mencolok bertahan dalam rentang waktu yang lama.

Berdasarkan penjelasan pada tabel di atas mengenai penelitian terkait yang berjudul "*Spatial eigenvector filtering for spatiotemporal crime mapping and spatial crime analysis*" dapat diketahui perbedaan dengan tugas akhir yang dikerjakan, yaitu:

1. Penelitian tersebut menggunakan metode ESF untuk memetakan pola dan analisa kejahatan, sedangkan untuk pengerjaan tugas akhir ini menggunakan metode NER,
2. Cakupan yang dilakukan penulis dilakukan pada wilayah kota Houston, sedangkan untuk pengerjaan tugas akhir ini

mengambil cakupan yang lebih luas, yaitu negara Indonesia.

Tabel 2.4 Literatur 4

Judul	*Toward a Cybercrime Classification Ontology: A Knowledge-based Approach*
Penulis	Charlette Donalds, Kweku-Muata Osei-Bryson
Ringkasan	Penelitian ini membahas mengenai pembuatan ontologi untuk kejahatan siber dengan beberapa perspektif untuk menyediakan sudut pandang yang lebih luas. Tujuan penelitian ini adalah pencegahan kejahatan siber dengan membuat aplikasi untuk klasifikasi kejahatan siber dengan pendekatan *knowledge-based*, luaran dari aplikasi tersebut adalah solusi yang dapat diterapkan dari setiap evaluasi *CCO (Cybercrime Classfication Ontology).*

Berdasarkan penjelasan pada tabel di atas mengenai penelitian terkait yang berjudul *"Toward a Cybercrime Classification Ontology: A Knowledge-based Approach"* dapat diketahui perbedaan dengan tugas akhir yang dikerjakan, yaitu:

1. Penelitian tersebut membahas pembuatan ontologi untuk kejahatan siber, sedangkan pada pengerjaan tugas akhir ini difokuskan untuk kejahatan secara umum menurut kelompok jenis kejahatan di negara Indonesia,
2. Penelitian tersebut menghasilkan luaran berbentuk solusi dari aplikasi, sedangkan luaran pada tugas akhir ini berbentuk data *spatiotemporal/geocode(longitude langitude).*

Tabel 2.5 Literatur 5

Judul	*A cyber forensics ontology: Creating a new approach to studying cyber forensics*
Penulis	Ashley Brinson, Abigail Robinson, Marcus Rogers

Ringkasan	Penelitian ini bertujuan untuk menemukan lapisan yang benar untuk spesialisasi, sertifikasi, dan pendidikan dalam domain forensik siber dengan cara membuat ontologi forensik siber. Ontologi yang dibuat tersusun atas 5 lapisan struktur hierarki dengan layer final di spesifikkan untuk area sertifikasi dan spesialisasi. Domain forensik siber yang diamati ada 2 yaitu *technology* dan *profession*, untuk *technology* ada *hardware* dan *software*, untuk *profession* ada *law*, *academia*, *military*, dan *private sector*.

Berdasarkan penjelasan pada tabel di atas mengenai penelitian terkait yang berjudul *"A cyber forensics ontology: Creating a new approach to studying cyber forensics"* dapat diketahui perbedaan dengan tugas akhir yang dikerjakan, yaitu:

1. Penelitian sebelumnya berfokus untuk membuat ontologi pada domain forensik siber yang berguna untuk penelitian lebih lanjut pada masa mendatang, sedangkan pada pengerjaan tugas akhir ini berfokus pada pembuatan ontologi pada domain kejahatan untuk membuat peta persebaran kejahatan di negara Indonesia.

2.2 Dasar Teori

2.2.1 *Class* Kejahatan

Kejahatan atau kriminalitas merupakan suatu tindakan atau perbuatan tingkah laku yang bertentangan dengan undang-undang menurut R. Soesilo. Definisi "Kejahatan" menurut R.Soesilo dalam bukunya berjudul "Kitab Undang-Undang Hukum. Pidana serta Komentar-Komentar Lengkap Pasal Demi Pasal" (1985, Penerbit Politeia) membedakan pengertian kejahatan menjadi dua sudut pandang yakni sudut pandang secara yuridis sudut pandang sosiologis. Dalam Kitab Undang-Undang Hukum Pidana (KUHP) tidak dijelaskan secara eksplisit mengenai definisi kejahatan, namun KUHP sudah mengatur berbagai delik kejahatan dalam pasal 104 hingga pasal

488 KUHP.Menurut Laporan Evaluasi Data Polri, SUSENAS(Survei Sosial Ekonomi Nasional), dan PODES (Potensi Desa), berikut beberapa konsep kriminalitas yang berhubungan dengan tindak kejahatan [5]:

2.2.1.1 Tindak kejahatan/kriminalitas

adalah perbuatan seseorang yang melanggar hukum menurut Kitab Undang-Undang Hukum Pidana (KUHP), Undang-Undang, atau peraturan lainnya yang berlaku di Indonesia dan dapat dijerat hukum.

2.2.1.2 Pelaku kejahatan

adalah orang yang melakukan kejahatan, turut melakukan kejahatan, menyuruh melakukan kejahatan, me mbujuk orang lain untuk melakukan kejahatan, dan membantu untuk melakukan kejahatan.

2.2.1.3 Tahanan

adalah tersangka pelaku tindak kejahatan yang ditahan oleh pihak kepolisian sebelum diteruskan kepada Kejaksaan atau masih dalam proses pengusutan lebih lanjut. Lama penahanan kurang dari 20 hari.

2.2.1.4 Korban kejahatan

adalah seseorang/harta bendanya mengalami kerugian akibat tindak kejahatan atau usaha tindak kejahatan. Korban kejahatan dalam Susenas dikelompokkan menjadi dua klasifikasi, yaitu rumah tangga dan individu. Penentuan kriteria korban kejahatan ini hanya berdasarkan pada pengakuan responden tanpa melihat lagi aspek hukumnya. Klasifikasi korban kejahatan menurut umur adalah:

- Anak-anak: orang yang berumur kurang dari 18 tahun,
- Dewasa: orang yang berumur 18 tahun dan lebih.

2.2.1.5 Kerugian

adalah hilang, rusak, atau musnahnya harta benda yang ditimbulkan dari tindak kejahatan atau usaha tindak kejahatan, dan tidak termasuk korban jiwa/badan.

2.2.2 Bentuk Gangguan Keamanan

Adalah tindakan yang sudah nyata, menimbulkan situasi tidak aman terhadap properti, personil/ manusia, data/informasi dan lingkungan pengamanan dan menimbulkan dampak bagi perusahaan. Jenis gangguan keamanan menurut buku panduan PT. Pertamina ada pada Tabel 2.6. Buku Panduan PT. Pertamina bersifat rahasia sehingga tidak dapat digunakan sebagai sitasi.

Tabel 2.6 Bentuk dan Area Gangguan Keamanan

Bentuk Gangguan	Area
Sabotase	• Sabotase terhadap Instalasi dan peralatan operasional migas termasuk diantaranya proses kerja, pipa, tangki penyimpanan • Sabotase terhadap kapal laut, mobil tangki, dan sarana pengangkutan
Terorisme	Perbuatan teror terhadap instalasi dan peralatan operasional migas termasuk diantaranya proses kerja, pipa, tangki penyimpanan, kapal laut, mobil tangki, dan sarana pengangkutan.
Pencurian Properti	• Pencurian terhadap Hidrokarbon yang disimpan, diproses, diproduksi dan diangkut • Pencurian terhadap peralatan di stasiun metering (titik kirim, titik serah), dan barang yang disimpan dalam pergudangan • Pencurian terhadap Instalasi dan peralatan operasional migas termasuk diantaranya peralatan proses kerja, pipa, tangki penyimpanan • Pencurian terhadap peralatan dan/atau kapal laut, mobil tangki, dan sarana pengangkutan

	• Pencurian terhadap properti penunjang seperti instalasi penunjang (air, listrik, gas) hingga fasilitas penunjang (perumahan, area olahraga, dll)
Pencurian Data	Pencurian terhadap informasi bisnis penting, data, sistem manajemen yang dapat diakses baik melalui softcopy maupun hardcopy, dikirimkan melalui pengiriman manual hingga mobile.
Demonstrasi	Demonstrasi di area Obvitnas perusahaan.
Blokade	Blokade terhadap akses operasional perusahaan, juga terhadap jalur yang dilalui Kapal laut, Mobil tangki dan sarana pengangkutan lainnya.
Pembunuhan	Pembunuhan terhadap Pekerja dan Mitra kerja/Kontraktor Mode 1 & 2 (Beroperasi di bawah supervise Perusahaan).
Penganiayaan/ Perkelahian	Penganiayaan terhadap Pekerja dan Mitra kerja/Kontraktor Mode 1 & 2 (Beroperasi di bawah supervise Perusahaan).
Kegagalan Akses Kontrol	Kegagalan mengelola akses control (Masuk dan keluar tanpa izin, tidak terdatanya personil yang masuk ke dalam lokasi, Pendokumentasian tanpa izin dapat membahayakan keamanan di lokasi kerja dan merugikan perusahaan).
Penculikan	Penculikan terhadap Pekerja dan Mitra kerja/Kontraktor Mode 1 & 2 (Beroperasi di bawah supervise Perusahaan).
Vulnerability	Kerawanan di Instalasi dan peralatan operasional migas termasuk diantaranya proses kerja, pipa, tangki penyimpanan, juga terhadap Pekerja dan Mitra kerja/Kontraktor Mode 1 & 2.

2.2.3 *Twitter*

Adalah platform *microblogging* yang memiliki pengguna aktif sekitar 554.7 juta dan setiap hari terdapat sekitar 58 juta "*tweets*" yang dibuat oleh pengguna. *"Tweets"* adalah pesan

yang dikirim oleh pengguna dan dapat dibaca oleh pengguna lainnya[6].

Gambar 2.1 Contoh *tweet*

Gambar 2.1 merupakan contoh dari suatu *tweet,* dapat dilihat beberapa elemen yang membentuk *tweet* tersebut. Terdapat elemen teks yang merupakan '*body*', ada *hashtag (#),* ada waktu *tweet* tersebut dikirim *(metadata timestamp),* dan gambar profil serta *username* dari pengguna tersebut [7].

2.2.4 *Named Entity Recognition*

Adalah sub tugas dari ekstraksi informasi dari data tidak terstruktur, seperti *tweet* menjadi beberapa kategori dalam data terstruktur. *NER* berfungsi untuk mengidentifikasi informasi yang dibutuhkan seperti nama, kejadian, waktu, dll [10]. Untuk melakukan sebuah deteksi pola/kategorisasi, *NER* melihat pola kata disekitarnya.

NER juga diselesaikan dengan pelabelan pada urutan kata statistik *(statisctical sequence-labeling)* yang mendeteksi batas atau segmen dan tipe dari *named-entity*. Berikut merupakan contoh bagaimana metode *NER* mendeteksi *entity* pada suatu *tweet* pada Gambar 2.2.

Gambar 2.2 Contoh deteksi *entity* dengan *NER*

2.2.5 *Ontology*

Ontology merupakan bagian dari machine learning, berfungsi untuk memudahkan pemahaman informasi antar orang dan sistem aplikasi, dengan memberikan spesifikasi tertentu seperti skema metadata yang menyediakan kosakata konsep yang terkontrol, sehingga dapat dipahami oleh mesin dan batasan-batasan penggunaannya didefinisikan secara eksplisit [11]. Dalam ontologi untuk *semantic web*, ada beberapa istilah yang sering digunakan [12], yaitu:

1. *Class*, atau yang biasa disebut concept adalah cara untuk mengelompokkan suatu objek yang memiliki *traits* dan *properties* yang sama ke dalam suatu kelompok yang disebut *class*. Contoh: *class* "Kejahatan terhadap fisik" adalah untuk seluruh kasus kejahatan yang mengakibatkan rasa sakit, luka, atau merugikan kesehatan korban yang dilakukan oleh pelaku kejahatan tersebut.
2. *Subclass*, merupakan bagian dari class namun mengelompokkan objek yang memiliki traits dan properties yang tidak umum untuk class secara keseluruhan. Contoh: *subclass* "Kekerasan dalam rumah tangga" adalah bagian dari class "Kejahatan terhadap fisik", tetapi hanya mengelompokkan kasus kejahatan yang terjadi dalam rumah tangga.
3. *Individual*, adalah objek atau entitas tunggal yang dimiliki oleh suatu *class*. Contoh: "Pemukulan bu Suminah oleh suaminya" adalah *individual* oleh *class* "Kejahatan dalam rumah tangga". *Individual* dapat berbentuk aktivitas maupun objek.
4. *Property*, digunakan untuk menggambarkan hubungan dalam ontologi. Ketika suatu property dihubungkan ke suatu class, class tersebut menjadi *domain* dari *property* tersebut. Objek dari suatu *class* yang dirujuk oleh property, disebut sebagai *Range*.
5. *Property Restriction*, digunakan untuk membentuk/membatasi suatu *property*. *Property*

Restriction biasanya berbentuk batasan nilai dan kardinalitas.

2.2.6 *TweetScraper*

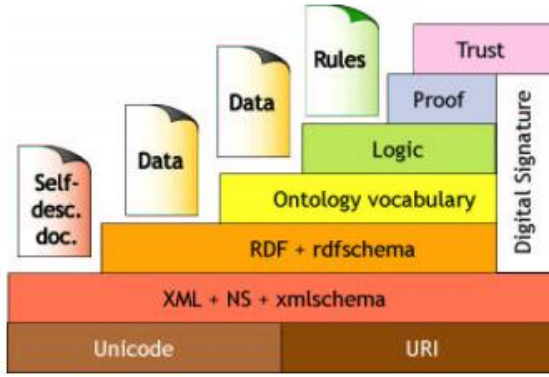
TweetScraper merupakan sebuah proses untuk mendapatkan informasi konten atau keseluruhan isi halaman yang terdapat pada suatu halaman web dan menyimpannya secara *offline* [13]. Dalam penelitian ini crawling dilakukan pada laman media sosial *twitter*. Penggalian data *Twitter ini* bertujuan untuk medapatkan data berupa *tweet* yang bersumber dari media sosial *Twitter*. *Data-data* yang diambil adalah *data* publik berupa *tweet*, waktu, pengguna, dan keterangan lain yang dibutuhkan. Bahasa pemrograman yang biasa digunakan adalah *Phyton*, serta modul yang digunakan untuk mengakses *Twitter* adalah *scrapy*.

2.2.7 *Semantic Web*

Adalah bagian dari *web* sudah ada saat ini, namun perbedaannya dengan *web* biasa adalah dimana informasi yang diberikan mempunyai makna yang jelas dan didefinisikan dengan baik, sehingga dapat membantu mesin memahami dan mengolah isi informasi dari *web* tersebut menjadi hasil yang diinginkan [14].

Menurut *World Wide Web Consortium (W3C*) [15], *Semantic Web* menyediakan banyak spesifikasi model data (*RDF, RDFS, OWL, SKOS),* beberapa spesifikasi aturan *(SWRL, RIF),* dan satu spesifikasi bahasa penelusuran *(SPARQL).*

Berikut adalah standar susunan *Semantic Web* pada Gambar 2.3 [16]:



Gambar 2.3 Standar *Semantic Web* oleh *W3C*

2.2.8 *OpenNLP API*

OpenNLP merupakan sebuah *library Java* untuk membantu dalam proses *Natural Language Processing* yang dikembangkan oleh Apache[17]. Tujuannya sendiri adalah untuk mempermudah program computer dalam melakukan ekstraksi makna dari *natural language*[17]. *OpenNLP* mendukung tugas-tugas berikut:

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

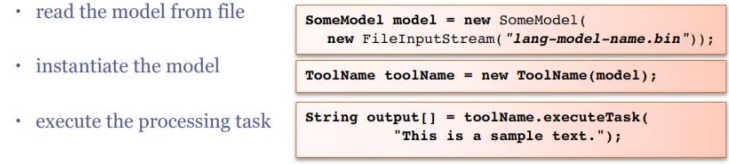
Pendekatan yang dilakukan *OpenNLP* sendiri adalah menginisasi data model yang mendukung tugas tertentu kemudian memberikan *method* kepada model untuk melakukan tugas yang diinginkan. *OpenNLP* sendiri berjalan di atas Java, sehingga dalam pengoperasiannya membutuhkan *JDK (Java Development Kit)* dan *JRE (Java Run-time Environment)*[18].

2.2.8.1 *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 2.4 Struktur *Library OpenNLP*

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

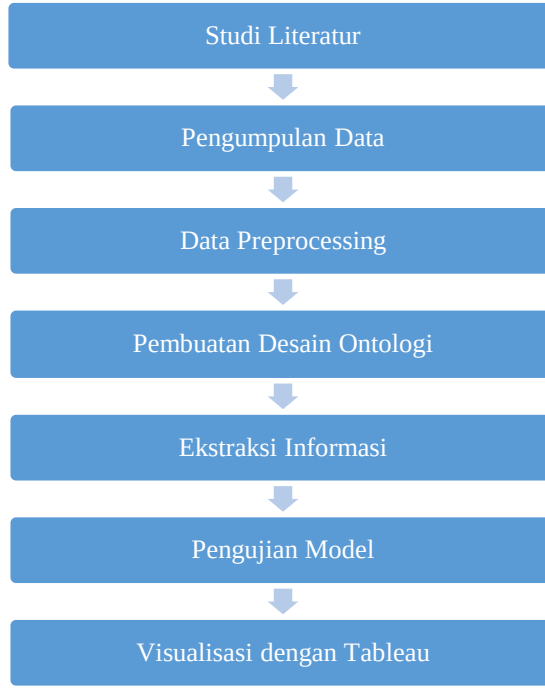
Halaman ini sengaja dikosongkan

BAB III
METODOLOGI

Pada bab metodologi akan dijelaskan mengenai tahapan-tahapan apa saja yang dilakukan dalam pengerjaan tugas akhir ini beserta deskripsi dan penjelasan tiap tahapan.

3.1 Tahapan Pelaksanaan Tugas Akhir

Pada sub bab ini akan menjelaskan mengenai metodologi dalam pengerjaan tugas akhir. Metodologi dapat dilihat pada Gambar 3.1



Gambar 3.1 Diagram metodologi

3.1.1 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 "*Techniques for Named Entity Recognition: A Survey"* yang ditulis oleh Girish Keshav Palshikar terkait *Named Entity Recognition,* dan penelitian *"A methodology for traffic-related Twitter messages interpretation"* yang ditulis oleh Fàbio C. Albuqueque.

3.1.2 Pengumpulan Data

Tahap pengambilan data ini dilakukan aktivitas *crawling* media sosial *Twitter* menggunakan *TweetScraper*, pengambilan data *twitter* dilakukan berdasarkan kategori/*keywords* yang diprediksi dari buku panduan Pertamina untuk setiap kelas kejahatan (ex: bacok, perampokan, jambret, pencurian, maling, begal). Kemudian dari *keywords* tersebut diambil data atau sampel, lalu diambil *top key* yang paling mewakili tiap kelas-kelas kejahatan tersebut. Dari *top key* tersebut, dilakukan kembali crawling untuk mengambil data yang sesuai dengan kelas kejahatan.

Pengumpulan data wilayah di negara Indonesia dilakukan juga untuk mengidentifikasi lokasi kejadian kasus kejahatan, dari tingkat terkecil yaitu kecamatan hingga yang besar yaitu provinsi. Tahap ini bertujuan untuk mengelompokkan dan melihat persebaran lokasi kejahatan yang terjadi di wilayah negara Indonesia.

3.1.3 *Data Preprocessing*

Tahap ini bertujuan untuk mengolah data sosial media yang telah didapatkan untuk memudahkan mesin memahami informasi yang terdapat dalam data. Data yang didapatkan akan

diolah menggunakan text pre-processing yang terdiri dari *Case Folding*, *Delete Duplicate*, dan *Data Cleansing.*

3.1.3.1 *Case Folding*

Sub-tahap ini bertujuan untuk memudahkan mesin memahami teks dengan cara merubah semua huruf menjadi kecil/*lowercase* untuk penyeragaman teks. Berikut adalah contoh pada Tabel 3.1.

Tabel 3.1 Contoh *Case Folding*

Sebelum *Case Folding*	Sesudah *Case Folding*
"Pembunuhan berantai terjadi di kota Manado, mengakibatkan satu keluarga tewas"	"pembunuhan berantai terjadi di kota manado, mengakibatkan satu keluarga tewas"

3.1.3.2 *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. Berikut adalah contoh pada Tabel 3.2.

Tabel 3.2 Contoh *Delete Duplicate*

Sebelum *Delete Duplicate*	Sesudah *Delete Duplicate*
• 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 Tetapkan Tersangka Kasus Dugaan Pembajakan Truk Tangki Pertamina News • Polisi Sebut Pelaku Pembajakan Truk Pertamina Hanya Ingin Cari Perhatian

3.1.3.3 *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 pada Tabel 3.3.

Tabel 3.3 Contoh *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

3.1.4 Pembuatan Desain Ontologi

Tahap ini dimulai dengan membuat *class* utama yaitu *class* Kejahatan dengan lingkup kelompok/jenis kejahatan atau *Class* Kejahatan yang memiliki *Actor*,dan Kejahatan.

3.1.4.1 Actor

Adalah aktor dalam class Kejahatan yang berupa korban atau pelaku yang berhubungan dengan kejahatan yang terjadi.

3.1.4.2 Kejahatan

Adalah jenis-jenis kejahatan/gangguan keamanan, berupa kategori yang berhubungan dengan kejadian yang terjadi.

3.1.5 Ekstraksi Informasi

Tahap ini melakukan ekstraksi informasi di dalam sebuah data *tweet.* Untuk melakukan ektraksi informasi tersebut, berikut proses yang perlu dilakukan.

3.1.5.1 *Tagged Tweet*

Proses ini melakukan aktivitas ektraksi informasi dari data yang diberikan menggunakan metode *Named Entity Recognition (NER),* contoh alur aktivitas ektraksi tersebut seperti berikut: 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.

3.1.6 *Dependencies Tree*

Tugas ekstraksi relasi mengacu pada masalah mendeteksi hubungan antara entitas yang diekspresikan dalam tweet yang ditandai. Ia menerima sebagai input tweet yang ditandai T dan mengembalikan pohon ketergantungan 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. . Kami mencatat bahwa tugas ekstraksi relasi sebenarnya menghitung QT pohon, dan bukan grafik generik.

3.1.7 Pengujian

Pengujian dilakukan dalam beberapa bagian, yaitu pada bagian akurasi ekstraksi informasi menggunakan ontology akan diuiji dengan *precision, recall,* dan *F-measure.*

$$Precision = \frac{Tp}{(Tp + Fp)}$$

$$Recall = \frac{Tp}{(Tp + Fn)}$$

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

Dimana "*Tp*", "*Fp*", "*Fn*", dan "*Tn*" adalah "*true positive*", *"false positive", "false negative",* dan *"true negative"*.

Pengujian dilakukan untuk evaluasi akurasi yang didapat untuk mengkategorikan informasi ke dalam suatu kategori, seperti *actor, location,* keterangan, dan *time*.

3.1.8 Visualisasi dengan Tableau

Aktivitas pembuatan *dashboard* dilakukan untuk menampilkan informasi yang berhasil diekstraksi dari data. Dengan demikian informasi akan ditampilkan pada platform dengan informasi berupa pemetaan kejahatan/gangguan keamanan PT. Pertamina berdasarkan penyebabnya di setiap lokasi daerah di Indonesia.

BAB IV
PERENCANAAN

Pada bab perencanaan akan dijelaskan mengenai detail dari proses pengerjaan penelitian yang dijelaskan pada bab metodologi. Perancangan ini akan digunakan sebagai panduan dalam melakukan penelitian tugas akhir, yang dijelaskan sebagai berikut.

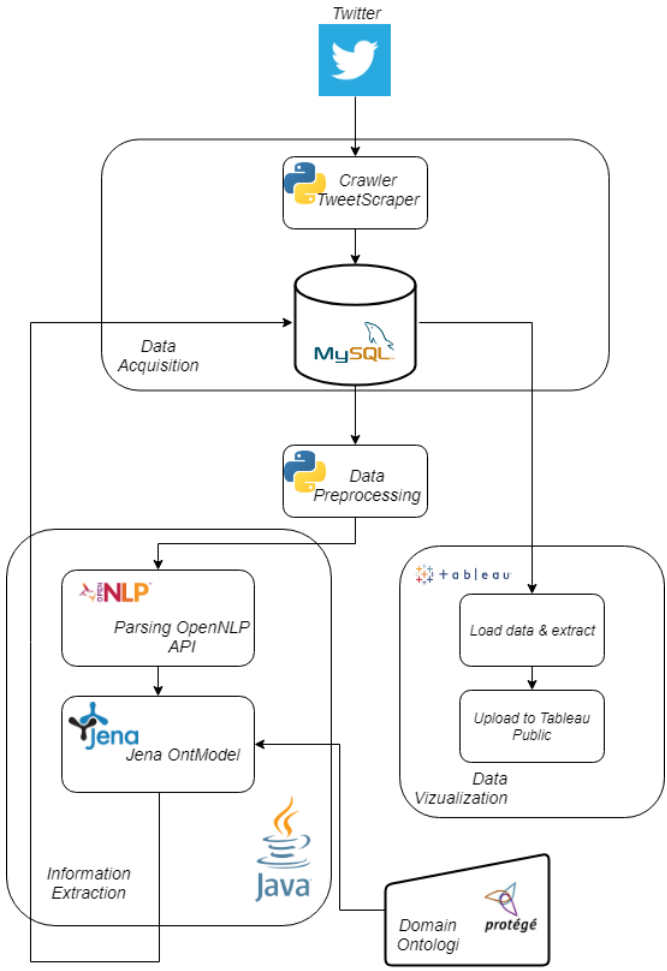
4.1 Arsitektur Sistem

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. Sumber *data* pada tugas akhir ini adalah *data* pada *Twitter*.
2. *Data* pada *Twitter* diakuisisi melalui proses *crawling* menggunakan TweetScraper. Hasil *crawling* data disimpan pada database *MySQL*.
3. *Data* yang telah berhasil diakuisisi kemudian dilakukan *preprocessing*, dimana data dibersihkan (penghapusan duplikasi, *case folding*, *date merging*, *data cleansing*).
4. Setelah itu, *data* dilanjutkan untuk dimasukkan ke dalam proses ekstraksi informasi dengan *OpenNLP API Parser*. Langkah ini berguna untuk mengekstrak entitas yang terkandung di dalam suatu *tweet.*
5. 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*.

6. *Data* kembali dimasukkan ke dalam *database MySQL* setelah proses ekstraksi informasi.
7. Untuk memvisualisasikan hasil yang didapat, penulis menggunakan *Tableau* untuk ditampilkan ke dalam suatu *dashboard*.

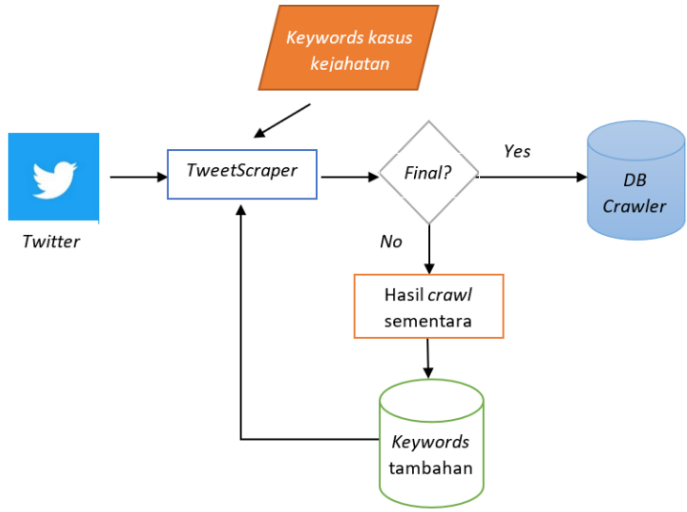


Gambar 4.1 Arsitektur Sistem

4.2 Pengumpulan Data

4.2.1 Desain *Crawler*

Tahap pengumpulan data ini dilakukan aktivitas *crawling* media sosial *Twitter* menggunakan *TweetScraper*, pengumpulan data twitter dilakukan berdasarkan kategori/*keywords* yang diprediksi dari buku panduan Pertamina untuk setiap kelas kejahatan. Kemudian dari *keywords* tersebut diambil *data* atau sampel untuk dihitung *frequency*. Dari hasil perhitungan tersebut akan dihasilkan *keywords* tambahan yang mempertimbangkan subjektifitas penulis, yang akan dilakukan *crawling* lagi untuk mengambil *data* yang akan digunakan sebagai *tweet* tambahan untuk pembuatan *model* pada sub-bab 5.5.1. Hal ini dilakukan agar variasi model untuk menangkap *entities* lebih banyak. Kemudian akan disimpan ke dalam *database MySQL*. Berikut pada Gambar 4.2 dijelaskan alur *crawler Twitter*.



Gambar 4.2 Alur *crawler*

4.2.2 Desain *Database*

Tahap ini bertujuan untuk membuat tempat penyimpanan hasil crawling dari *twitter*. Berikut adalah atribut database yang akan dibuat untuk menyimpan data hasil crawl pada Tabel 4.1.

Tabel 4.1 Atribut *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*

Data hasil *crawling* yang didapatkan akan diolah menggunakan *pre-processing* yang terdiri dari *Case Folding, Delete Duplicate, Data Cleansing*, dan *Date Merging.* Alur dapat dilihat pada Gambar 4.3.

4.3.1 *Case Folding*

Sub-tahap ini akan melakukan pengubahan struktur kata menjadi huruf kecil. Hal ini bertujuan untuk normalisasi *tweet.*

4.3.2 *Delete Duplicate*

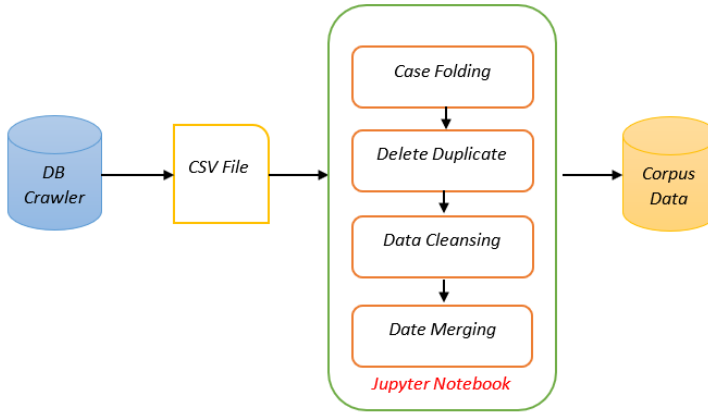
Sub-tahap ini melakukan proses menghapus *tweet* yang duplikat agar tidak membuat *noise* yang besar pada data. Akan dilakukan dengan modul *drop_duplicate()* dari *Pandas.*

4.3.3 *Data Cleansing*

Sub-tahap ini melakukan proses menghapus karakter-karakter, simbol, *link* pada *tweet*. Akan dilakukan dengan modul *preprocessor*.

4.3.4 *Date Merging*

Sub-tahap ini melakukan proses menggabungkan *tweet* dengan tanggal *tweet* tersebut dikirimkan. Tanggal didapatkan dari kolom *datetime* pada Tabel 4.1.



Gambar 4.3 Alur *Data Preprocessing*

4.4 Pembuatan Desain Ontologi

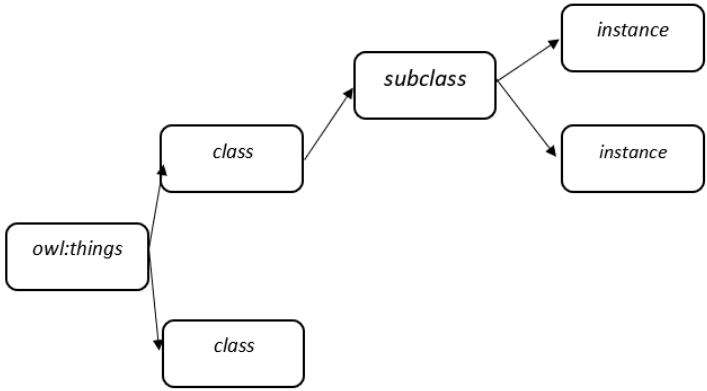
Perancangan Ontologi Kejahatan dilakukan menggunakan perangkat lunak *Protégé*. Pada tahap ini akan ditentukan *Class* apa saja yang akan dibuat dalam ontologi, yaitu *Class* Kejahatan. Rancangan desain ontologi dapat dlihat pada Gambar 4.4.

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 Kejahatan

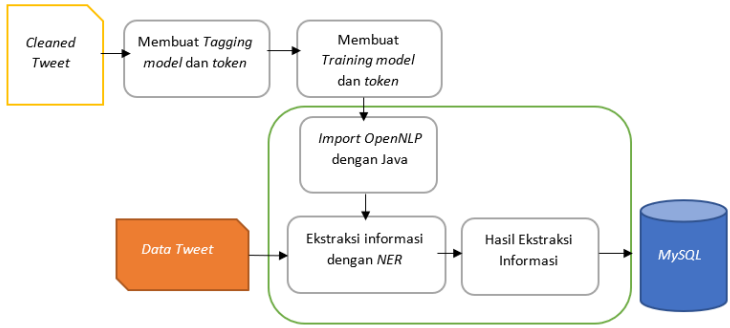
Class Kejahatan memiliki beberapa kategori kejahatan dalam bentuk *subclass. Subclass* tersebut akan menampung *instances* keterangan yang sesuai dengan kategorinya. Kategori kejahatan mengacu pada buku panduan PT. Pertamina.



Gambar 4.4 Rancangan desain ontologi

4.5 Ekstraksi Informasi

Pada tahap ekstraksi informasi 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 alur pengerjaan pada proses ekstraksi informasi pada Gambar 4.5.



Gambar 4.5 Alur ekstraksi informasi

4.5.1 *NER Tagging*

Tahap ini akan melakukan proses *labeling* atau *nameSpans* pada dataset yang digunakan dengan memberikan *nameSpans* <START:*tag*> dan <END> pada entitas yang telah ditentukan. Masing-masing entitas memiliki data dengan *nameSpans* masing-masing dengan jumlah *minimum data* yang dibutuhkan untuk setiap entitasnya adalah 15,000 *rows* dengan *nameSpans*. Entitas yang digunakan dalam penelitian ini adalah *Actor*, *Location*, Keterangan, dan *Time*. Untuk entitas *Actor* dan Keterangan akan dilakukan secara manual menggunakan *find and replace*. Sedangkan untuk entitas *Location* dan *Time* akan menggunakan kode program *python* untuk otomasi. Setelah proses pelabelan selesai dilakukan, akan dilakukan penghapusan *rows* yang tidak mengandung *tag* atau *label*. Akan dilakukan dengan kode program *python* dengan mengecek per *rows model* satu per satu, jika tidak mengandung *tag* atau *label*, akan dihapus.

4.5.1.1 *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>.

Data lokasi yang didapatkan berbentuk 4 tabel, yaitu tabel provinsi, tabel kabupaten/kota, tabel kecamatan, dan tabel lokasi singkatan. Berikut pada Gambar 4.6-Gambar 4.9 merupakan contoh dari tabel 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 Tabel Lokasi Provinsi

	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 4.7 Tabel Lokasi Kabupaten/Kota

	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 4.8 Tabel Lokasi Kecamatan

	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 Tabel Lokasi Singkatan

4.5.2 *Token Tagging*

Tahap ini akan melakukan 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.

4.5.3 *Model Training*

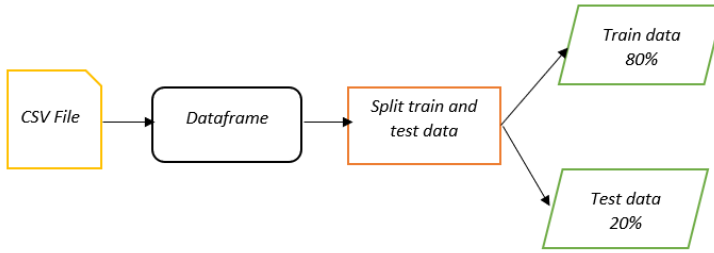
Tahap ini akan melakukan proses *training* pada *dataset* yang digunakan guna mendapatkan *output file data model* (dengan ekstensi ***.bin*** / ***.train***). Proses *training* dilakukan dengan menggunakan *library OpenNLP* pada *Command Line*. *TokenNameFinderTrainer* untuk train model *NameFinder* atau *NER* untuk label *Actor, Location,* Keterangan, dan *Time, TokenizerTrainer* untuk *train model token*.

4.5.4 *Parser* dengan *OpenNLP API*

Tahap 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.

4.6 Pengujian Model

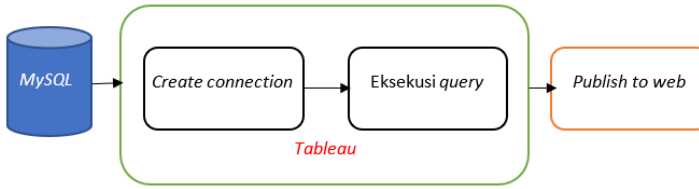
Data hasil ekstrasi pada tahapan sebelumnya akan diuji performanya dengan beberapa metode, yaitu *Precision*, *Recall*, dan *F-Measure*. Prosedur pengujian akan membagi data menjadi 2 bagian, yaitu 80 bagian untuk pelatihan, dan 20 bagian untuk melakukan pengujian. Berikut pada Gambar 4.10 merupakan alur pengujian model.



Gambar 4.10 Alur pengujian model

4.7 Visualisasi dengan Tableau

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 *Tableau*. Rancangan dashboard yang akan ditampilkan dari hasil ekstraksi informasi mengenai kejahatan/gangguan keamanan berdasarkan wilayah, periode waktu, dan jumlah kecelakaan yang terjadi. Alur data dengan model dashboard visualisasi dapat dilihat pada Gambar 4.11.



Gambar 4.11 Alur data visualisasi

BAB V
IMPLEMENTASI

Bab ini menjelaskan hasil dari perancangan studi kasus atau hasil dari proses pelaksanaan penelitian. Hasil yang akan dijabarkan adalah hasil eksperimen terhadap data yang digunakan sebagai acuan penelitian.

5.1 Lingkungan Implementasi

Dalam pelaksanaan tugas akhir mengenai kejahatan pada asset PT. Pertamina di media sosial *Twitter* ini membutuhkan perangkat keras dan lunak yang dapat membantu berjalannya pengerjaan tugas akhir ini. Adapun perangkat-perangkat tersebut dapat berupa perangkat keras yang spesifikasinya ditunjukkan pada 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

Nama	HP Pavilion 14-CE2010TX
Prosesor	Intel Core i5 8265U 4C 8T 1,60 GHz upto 3,90 GHz
Memory	8 GB x 1 2400 MHz LPDDR4 SDRAM (8 GB)
GPU	Intel UHD Graphics 620 Nvidia GeForce MX250
OS	Windows 10 1903 May 2019 Update
Arsitektur	64bit

Tabel 5.2 Perangkat lunak yang digunakan

Software	**Penggunaan**	**Versi**
Anaconda	Sebagai environment Python untuk pemrosesan data	2019.03
Jupyter IDE	Menjalankan *pre-proccessing*	5.7.8
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
RDBMS MySQL	Database hasil *crawling*	10.1.40-MariaDB
TweetScraper	*Crawler Twitter*	0.1

Tabel 5.3 *Library* yang digunakan

Library	**Penggunaan**	**Versi**
Python		3.7
pandas	Manipulasi dan analisa data	0.24.2
preprocessor	Membersihkan *tweet*	0.1.2
scrapy	*Crawling twitter*	1.6.0
Java		1.8.0_211
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 Pengumpulan Data

Pada tahap ini akan dilakukan pengumpulan data dari media sosial *Twitter* yang membahas atau mengandung keywords

kejahatan sesuai buku panduan PT. Pertamina. Pengumpulan data dilakukan dengan *crawler TweetScraper* yang menggunakan *library scrapy*. *Crawler* ini tidak membutuhkan *Twitter API* dalam penggunaannya. *TweetScraper* dapat di unduh di https://github.com/jonbakerfish/TweetScraper.

Sebelum memulai proses *crawling*, perlu dilakukan penyuntingan setelan agar hasil *crawl* disimpan dalam *MySQL*. Buka file *setting.py*, hapus tanda pagar (#) untuk mengaktifkan *pipeline.SavetoMySQL* dan beri tanda pagar (#) untuk menonaktifkan *pipeline.SaveToFile*. Berikut potongan kode *setting.py* pada Kode Program 5.1

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

# !!! # Crawl responsibly by identifying yourself (and
your website/e-mail) on the user-agent
USER_AGENT = 'bennybtc03@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/'
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 Kode program *setting.py TweetScraper*

Untuk menjalankan *crawler*, buka *cmd* lalu ketikkan *scrapy crawl TweetScraper -a query="query"*. Berikut pada Kode Program 5.2 adalah *query* yang digunakan untuk mengambil *data* pada media sosial *Twitter*. *Keywords query* didapatkan dari Buku Panduan PT. Pertamina.

```
scrapy crawl TweetScraper -a query="demonstrasi
pertamina"
scrapy crawl TweetScraper -a query="pembajakan
pertamina"
scrapy crawl TweetScraper -a query="pembunuhan
pertamina"
scrapy crawl TweetScraper -a query="sabotase
pertamina"
scrapy crawl TweetScraper -a query="penganiayaan
pertamina OR perkelahian pertamina"
scrapy crawl TweetScraper -a query="pencurian
pertamina"
scrapy crawl TweetScraper -a query="kebakaran
pertamina"
scrapy crawl TweetScraper -a query="terorisme
pertamina"
```

Kode Program 5.2 Kumpulan *query crawler*

Kemudian untuk hasil *crawl* kedua, penulis melakukan *crawling* kembali untuk *keywords* "Kilang Pertamina", "Tangki Pertamina", dan "Truk Pertamina" dari perhitungan *frequency* per kata dan menggunakan subjektifitas penulis. Hasil dari *crawling* kedua hanya akan digunakan sebagai penyusun *model* tambahan agar akurasi *model* bertambah. Berikut pada Tabel 6.2 adalah *query keywords* tambahan yang digunakan untuk mengambil *data* tambahan.

```
scrapy crawl TweetScraper -a query="kilang
pertamina"
scrapy crawl TweetScraper -a query="tangki
pertamina"
scrapy crawl TweetScraper -a query="truk
pertamina"
```

Kode Program 5.3 Kumpulan *query crawler* untuk *keywords* tambahan

5.3 *Data Preprocessing*

Pada tahap ini, akan dibuat kode program *python* untuk membersihkan hasil *crawl.* Bagian yang dibersihkan adalah simbol, mention, *tweet* yang duplikat, tidak termasuk *tweet* asli (bukan *bot*), dan menghapus *link website*. Kode program dijalankan dalam *IDE Jupyter Notebook.*

Pertama perlu dilakukan *import library* yang dibutuhkan, yaitu *library pandas* dan *preprocessor*. *Library preprocessor* dapat diunduh pada github pengembang https://github.com/s/preprocessor. Berikut merupakan potongan kode untuk melakukan *import library* pada Kode Program 5.4.

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

Kode Program 5.4 *Library* untuk *data preprocessing*

Langkah selanjutnya adalah melakukan pemuatan data hasil *crawl Twitter* ke dalam *IDE* menggunakan *pandas read_csv().* Berikut merupakan potongan kode program dalam Kode Program 5.5.

```
3.  #load file ascii (menghilangkan kanji, mandarin dll
    .)
4.  with open("D:/S1 Sistem Informasi ITS/Semester 8/TU
    GAS AKHIR/kebakaran.csv", encoding='ascii', 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 Memasukkan data ke dalam *IDE*

Setelah hasil *crawl* dimuat ke dalam *dataframe*, akan dilakukan penghapusan *link* yang terdapat di setiap *tweet. Link* tersebut memiliki pola yang sama, yaitu terletak di akhir setiap *tweet.* Untuk menghapus *link* tersebut, akan dihapus kata-kata setelah kata '*http'.* Berikut adalah

potongan kode program untuk melakukan penghapusan *link http* pada Kode Program 5.6. Langkah ini terkait dengan penjelasan pada sub-bab 4.3.3.

```
#menghilangkan http dst
without_http = []
x = 0
while x in range(0, total_rows):
    text = str(test.loc[x])
    head, sep, tail = text.partition('http')
    without_http.append(head)
    print(x , 'DEL : ', tail)
    x += 1
```

Kode Program 5.6 Penghapusan *link http* pada *tweet*

Langkah selanjutnya adalah menggabungkan tanggal *tweet* ketika di posting dengan isi dari *tweet* tersebut. Hal ini bertujuan untuk mempermudah proses *NER* pada tahap selanjutnya. Berikut potongan kode program proses tersebut pada Kode Program 5.7. Langkah ini terkait dengan penjelasan pada sub-bab 4.3.4.

```
#menambahkan date
with_date = []
x = 0
for x in range(0, total_rows):
    dates =  ' !'+  str(date.loc[x])[:10]
    textmain = str(without_http[x])
    text = textmain + dates
    with_date.append(text)
    print(x, text)
    x += 1
```

Kode Program 5.7 Menambahkan *date* ke dalam *tweet*

Kemudian akan dilakukan pembersihan *tweet* dari simbol dan tanda baca dengan menggunakan *sets mapping* yang sudah ada. Untuk setiap tanda baca dan symbol dilakukan mapping terlebih dahulu dengan menyebutkan tanda baca dan simbol yang ingin dibersihkan. Setelah proses pembersihan tersebut, dilakukan proses menghitung banyak kata dalam suatu kalimat dengan tujuan untuk menghilangkan *tweet* yang tidak memiliki

informasi dengan mengecek komposisi kalimat (SPOK). Langkah yang ditempuh adalah membagi *tweet* menjadi dua kelompok yaitu *cleaned_text* dan *cleaned_text_short*. *cleaned_text_short* berisi *array* yang telah diperpendek menjadi 30 karakter saja dengan tujuan untuk menghapus tweet yang duplikat berdasarkan 30 karakter pertama, sedangkan *cleaned_text* berisikan array dengan *tweet* utuh. Berikut potongan kode program tersebut pada Kode Program 5.8.

```
#membersihkan text
def clear(text):
    mapping = [ (' rt ',''),(',',''), ('~','')
,('{',''),('}',''),('|',''), ('.',''), (':',''
), ('(',''), (')',''), ("'",''), ('?',''), ('m
arker',''), (';',''), ('#',''),('\\n',''), ('[
',''),(']',''),('/',''),('=',''),('$',''),('_'
,''),('+',''),('"',''),('*',''),('!',''),('%',
''), ('<',''),('>',''),('^','')]
    for m, n in mapping:
        text = text.replace(m, n)
    return text

cleaned_text_short = []
cleaned_text = []

#memasukkan text ke array
x = 0
while x in range(0, total_rows):
    hasil = clear(p.clean(str(with_date[x])))

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

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

Kode Program 5.8 *Cleaning tweet* dan membagi menjadi *short & long tweet*

Langkah selanjutnya adalah memasukkan *cleaned_text_short* ke dalam *dataframe* untuk melakukan *drop_duplicate tweet* dengan menggunakan *cleaned_text* sebagai subset. Berikut potongan kode program pada Kode Program 5.9. Langkah ini terkait dengan penjelasan pada sub-bab 4.3.2.

```
#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 5.9 *Drop duplicate* dengan *subset short & long tweet*

Selanjutnya adalah melakukan pencocokan *tweet* utuh dengan *tweet* yang sudah dipotong menjadi 30 karakter pertama, tujuan dari kode program ini adalah untuk mendapatkan *tweet* utuh tanpa duplikat. Setelah itu akan dilakukan proses *export dataframe* ke dalam bentuk *CSV*. Berikut potongan kode program pada Kode Program 5.10.

```
#mengubah dataframe short tweet menjadi array
d = data['cleaned_text']
d_array= d.values
d_array[10]
print(len(d_array))
#mengubah dataframe long tweet menajadi array
c = CTdf['cleaned_text']
c_array= c.values
c_array[10]
#mencocokkan short tweet dengan long tweet
final_text = []
x = 0
y = 0
while x in range(0, len_cleaned):
    strUtama = str(c_array[x])
    for y in range(0, len(d_array)):
        strPembanding = str(d_array[y])
        if(strUtama[:30] == strPembanding):
            final_text.append(strUtama)
            d_array[y] = ['tes']
            print(y ,strUtama)
            y += 1
        else:
            y += 1
    x += 1
#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)
```

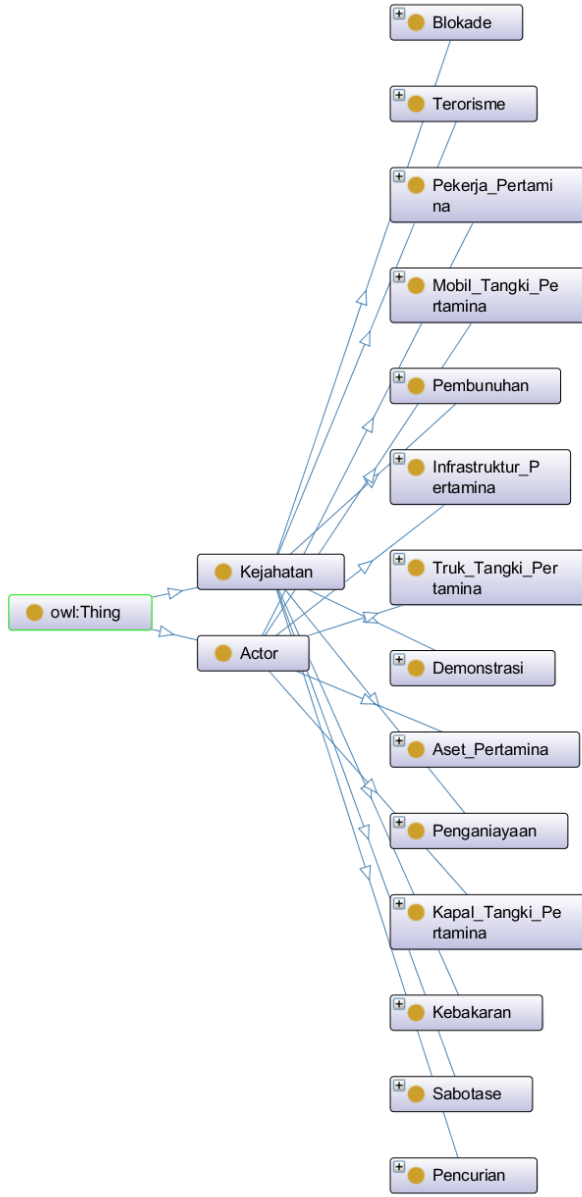
Kode Program 5.10 Mencocokan *short tweet* dan *long tweet*

5.4 Pembuatan Desain Ontologi

Pada tahap ini akan menggunakan *software* yang bernama *Protégé* untuk membuat *class* ontologi yang berguna untuk mengkategorikan hasil ekstraksi informasi ke dalam kategori *class* ontologi yang sudah dibuat. Hasil tahap ini menghasilkan dua *class* yaitu *Actor* dan Kejahatan. *Class Actor* memiliki beberapa *SubClass* kategori *Actor*, sedangkan *Class* Kejahatan memiliki beberapa *SubClass* berdasarkan kategori kejahatan/gangguan keamanan. Tahap ini menghasilkan *file* berekstensi *.owl*. Pada Gambar 5.1 merupakan tangkapan gambar struktur *class* dan *subclass*.

Gambar 5.1 Struktur *Class* dan *SubClass* ontologi pada *Protégé*

Kemudian untuk mempermudah relasi/hubungan antar *class/subclass* dapat menggunakan fitur *OntoGraf*. Berikut pada Gambar 5.2 merupakan tangkapan gambar *OntoGraf*.



Gambar 5.2 *OntoGraf* ontologi

5.4.1 *Actor*

Class actor memiliki 6 *subclass* kategori yang dibuat berdasarkan jenis-jenis *instances. Subclass* ditentukan berdasarkan subjektifitas penulis.

Berikut adalah daftar *subclass* dan *instances* pada *class actor* pada Tabel 5.4.

Tabel 5.4 Daftar *subclass* dan *instances* pada kelas *Actor*

Class	***Subclass***	***Instances***
Actor	Aset_Pertamina	• anjungan • bensin • kilang_minyak • kilang_minyak_pertamina • kilang_pertamina • minyak • penampung_minyak • pertalite • pertamax • pipa_gas • pipa_gas_pertamina • pipa_minyak • pipa_minyak_pertamina • pipa_pertamina • premium • solar • sumur • sumur_gas_pertamina • sumur_minyak • tabung • tabung_gas • tangki
	Infrastruktur_Pertamina	• depo • depo_pertamina • gedung • gedung_pertamina • gedung_pertamina_tower • gudang

		• gudang_pertamina • gudang_pipa_pertamina • kantor_pertamina • pertamina • pertamina_tower • pom • pom_bensin • rumah_pertamina • spbu • spbu_pertamina • terminal_bbm • terminal_pertamina
	Kapal_Tangki_Pertamina	• kapal • kapal_bbm • kapal_minyak • kapal_pertamina • kapal_tangki_pertamina • kapal_tanker • tanker
	Mobil_Tangki_Pertamina	• mobil • mobil_pertamina • mobil_tangki • mobil_tangki_pertamina
	Pekerja_Pertamina	• karyawan_pertamina • pejabat_pertamina • sopir
	Truk_Tangki_Pertamina	• truck_pertamina • truck_tangki_pertamina • truk_bbm • truk_minyak • truk_pertamina • truk_tangki • truk_tangki_pertamina

5.4.2 Kejahatan

Class kejahatan memiliki 8 *subclass* kategori yang dibuat berdasarkan jenis-jenis gangguan keamanan yang didapat dari buku panduan PT. Pertamina.

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

Tabel 5.5 Daftar *subclass* dan *instances* pada kelas Kejahatan

Class	***Subclass***	***Instances***
Kejahatan	Blokade	• blokade • pembajakan
	Demonstrasi	• demonstrasi
	Kebakaran	• dibakar • kebakaran • terbakar
	Pembunuhan	• pembunuhan
	Pencurian	• bocor • dicuri • kebocoran • pencurian
	Penganiayaan	• penganiayaan • perkelahian
	Sabotase	• pengrusakan • sabotase
	Terorisme	• ancaman • di teror • diteror • teror • teror bom • terorbom • terorisme

5.5 Ekstraksi Informasi

5.5.1 *NER Tagging*

Hasil dari *data preprocessing* akan digunakan juga untuk pembuatan model untuk melakukan proses ekstraksi informasi menggunakan metode *NER.* Model berfungsi untuk mendeteksi entitas dari suatu data dengan mengecek kata entitas tersebut dan kata di sekitar entitas tersebut. *Apache Opennlp* sudah menyediakan model-model yang dapat digunakan seperti model *person, name*, etc. Namun model yang tersedia hanya tersedia dalam bahasa selain Bahasa Indonesia, oleh karena itu dibutuhkan pembuatan *custom model* dengan menggunakan modul *NameFinderTrainer* dari *OpenNLP*. Syarat dari *custom model* adalah *minimal* 15,000 *rows* untuk menghasilkan model yang baik. *Data* untuk *model* akan menggunakan data hasil *preprocessing*. *Tag* yang akan kami gunakan dalam proses ekstraksi informasi ada 4, yaitu *Actor, Location, Time*, dan Keterangan. Berikut adalah contoh *tag* entitas pada Tabel 5.6.

Tabel 5.6 *Tag e*ntitas *NER* dan penggunaannya

Entities	***Tag***	**Penggunaan**
Actor	<START:actor> *entity_name* <END>	*Data* yang dibutuhkan seperti truk pertamina, kapal tanker, pipa pertamina, tower pertamina, pom bensin, dan karyawan pertamina.
Location	<START:location> *entity_name* <END>	*Data* 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 kejadian tersebut.

Hasil dari *tagging* akan dibuat menjadi *file* dengan ekstensi ***.train.***

Kemudian setelah pelabelan data selesai dilakukan, akan dilakukan penghapusan *rows data model* yang tidak mengandung *label*, karena tidak semua *tweet* mengandung entitas yang diinginkan, Berikut pada Kode Program 5.11 dapat dilihat pada *row* 13 proses cek *label* per *row*, jika di dalam partisi sebuah *text* terdapat *tag* yang diinginkan, maka *tweet* tersebut akan di *append* ke dalam *dataframe* dan di *export* menjadi *file*. Tahap ini terkait dengan penjelasan pada sub-bab 4.5.1.

```
1.  import pandas as pd
2.  with open("path_file", encoding='ascii', errors='ignor
    e') as infile:
3.      df = pd.read_csv(infile, names = ["text"], error_b
    ad_lines=False)
4.  df.head(5)
5.  test = df['text']
6.  print(test.loc[30])
7.  #cek jumlah tweet
8.  total_rows = len(test)
9.  print(total_rows)
10. #menghilangkan tweet yang tidak memiliki tag
11. with_tag = []
12. x = 0
13. while x in range(0, total_rows):
14.     text = str(test.loc[x])
15.     head, sep, tail = text.partition('<START:tag>')
16.     if(tail == ''):
17.         print(x , 'DEL: ', text)
18.     else:
19.         with_tag.append(text)
20.         print(x , 'GOOD: ', text)
21.     x += 1
22. #cek jumlah tweet
23. total_rows = len(with_tag)
24. print(total_rows)
25. #mengubah array menjadi dataframe
26. final_df = pd.DataFrame(data=with_tag, columns=None)
27. #mengubah dataframe menjadi CSV
28. final_df.to_csv('path_file/nama_file', sep='\t', encod
    ing='utf-8', header=False, index=False)
```

Kode Program 5.11 *Check label model*

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 5.7 merupakan contoh dari tahap ini.

Tabel 5.7 Contoh *model NER Tagging*

Entities	**Sebelum *NER Tagging***	**Setelah *NER Tagging***
Actor	lima tersangka ditangkap polisi atas pembajakan dua truk tangki pertamina bbm di jakarta utara kelima tersangka adalah mantan karyawan pertamina patra niaga download tvone connect untuk update berita harian anda android 2019-03-20	lima tersangka ditangkap polisi atas pembajakan dua <START:actor> truk tangki pertamina <END> bbm di jakarta utara kelima tersangka adalah mantan karyawan pertamina patra niaga download tvone connect untuk update berita harian anda android 2019-03-20
Location	video amatir kebakaran krl akibat tabrakan dengan mobil tangki pertamina di bintaro tangerang selatan 2013-12-09	video amatir kebakaran krl akibat tabrakan dengan mobil tangki pertamina di bintaro <START:location> tangerang <END> selatan 2013-12-09
Time	polisi tahan tersangka pembajakan truk tangki pertamina 2019-03-22	polisi tahan tersangka pembajakan truk tangki pertamina <START:time> 2019-03-22 <END>
Keterangan	soal kasus pembajakan truk pertamina polisi sebut ada tersangka 2019-03-19	soal kasus <START:keterangan> pembajakan <END> truk pertamina polisi sebut ada tersangka 2019-03-19

5.5.1.1 *Location Tagging*

Setelah *data* lokasi/daerah di Indonesia sudah didapatkan, akan dilakukan proses *replace string* lokasi yang terdapat pada *tweet* per baris. Kode program akan memasukkan *data* hasil *preprocessing* ke dalam *dataframe*, kemudian akan dicocokkan dengan *data* lokasi/daerah per *rows*, jika *rows* tersebut mengandung nama daerah/lokasi, akan diberi *label NER* seperti pada Tabel 5.7. Berikut merupakan potongan kode program untuk melakukan proses *location tagging* pada Kode Program 5.12.

Proses pencocokan *tweet* dari *dataframe* dengan data daerah/lokasi dibagi menjadi 4 bagian, yaitu cek nama provinsi terlebih dahulu pada *row* 50. Kemudian nama kabupaten kota pada *row* 60, kemudian nama kecamatan pada *row* 69, dan nama daerah/lokasi singkatan pada *row* 101. Tahap ini terkait dengan penjelasan pada sub-bab 4.5.1.1.

```
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 5.12 *Location Tagging*

5.5.2 *Token Tagging*

Model *Token* juga sudah disediakan oleh *Apache Opennlp* namun belum ada yang berbahasa Indonesia, oleh karena itu perlu dibuat *custom model* dengan menggunakan modul *TokenizerTrainer* dari *Opennlp.* Syarat dari *custom model* adalah minimal 15000 *rows* untuk menghasilkan model yang baik. Data untuk model akan menggunakan data hasil *crawl* juga. *Syntax* yang digunakan adalah <*SPLIT*>, yang akan

ditaruh di setiap akhir *rows*, dan diakhiri dengan titik (.), hasil dari *tagging* akan dibuat menjadi *file* dengan ekstensi ***.train.*** Berikut pada Kode Program 5.13 *row 13* merupakan kode untuk memberi *whitespace* berbentuk <*SPLIT*> pada setiap baris *dataframe*.

```
1.  import pandas as pd
2.
3.  #load file ascii (menghilangkan kanji, mandarin dl
    l.)
4.  with open("path_file", encoding='ascii', errors='i
    gnore',) as infile:
5.      df = pd.read_csv(infile,  names = ["text"])
6.  df.head(5)
7.  #cek jumlah tweet
8.  total_rows = len(test)
9.  print(total_rows)
10. #menambahkan <SPLIT>
11. with_split = []
12. x = 0
13. for x in range(0, total_rows):
14.     split =  ' <SPLIT>.'
15.     textmain = str(test.loc[x])
16.     text = textmain + split
17.     with_split.append(text)
18.     print(x, text)
19.     x += 1
20. #cek jumlah tweet
21. total_rows = len(with_split)
22. print(total_rows)
23. #mengubah array menjadi dataframe
24. final_df = pd.DataFrame(data=final_text, columns=[
    'text'])
25. final_df.head(5)
26. #mengubah dataframe menjadi CSV
27. final_df.to_csv('path_file/nama_file', sep='\t', e
    ncoding='utf-8', header=False, index=False)
```

Kode Program 5.13 *Token Tagging*

5.5.3 *Model Training*

5.5.3.1 *NameFinderTrainer*

Setelah file *tagging* sudah selesai dibuat, maka akan dilakukan training untuk mengubah menjadi file model yang dapat digunakan, dengan modul *NameFinderTrainer*. Berikut adalah

deskripsi argumen yang terdapat dalam modul pada Kode Program 5.14.

```
$ 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 5.14 Fungsi *command TokenNameFinderTrainer*

Secara urut ketikkan *script* berikut pada *command prompt*, seperti pada Kode Program 5.15*:*

```
$ opennlp TokenNameFinderTrainer -model
[nama_model_yang_akan_dihasilkan].bin -lang
id -data [file_tag_manual].train -encoding
```

Kode Program 5.15 Struktur *query TokenNameFinderTrainer*

Berikut adalah potongan *script* pada *command prompt Opennlp* pada Gambar 5.3.

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

Gambar 5.3 Contoh tampilan *script NameFinderTrainer* pada *console*

Kemudian tekan *enter* untuk melakukan proses *training* sampai selesai. Berikut tampilan *console* hasil *training NameFinderTrainer* pada Gambar 5.4.

```
C:\windows\system32\cmd.exe
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
Indexing events with TwoPass using cutoff of 0

       Computing event counts...  done. 476399 events
       Indexing...  done.
Collecting events... Done indexing in 13.37 s.
Incorporating indexed data for training...
done.
       Number of Event Tokens: 476399
           Number of Outcomes: 3
         Number of Predicates: 560629
Computing model parameters...
Performing 300 iterations.
  1:  . (475314/476399) 0.9977224973184242
  2:  . (475741/476399) 0.998618804825367
  3:  . (475867/476399) 0.9988832890077435
  4:  . (475957/476399) 0.9990722062808696
  5:  . (475997/476399) 0.9991561695133 7
  6:  . (476007/476399) 0.9991771603214952
  7:  . (476028/476399) 0.999221241018558
  8:  . (476064/476399) 0.9992968079278084
  9:  . (476081/476399) 0.9993324923016211
 10:  . (476094/476399) 0.9993597803521838
Stopping: change in training set accuracy less than 1.0E-5
Stats: (472917/476399) 0.9926910006108325
...done.

Training data summary:
#Sentences: 34289
#Tokens: 476399
#actor entities: 36853

Writing name finder model ... Compressed 560629 parameters to 9907
4 outcome patterns
done (0.270s)

Wrote name finder model to
path: C:\apache-opennlp-1.9.1\id-ner-actor-test.bin
```

Gambar 5.4 Tampilan *console* hasil *training NameFinderTrainer*

Pada folder *command prompt* dijalankan akan muncul *output file* dari proses *training*, yaitu model dari *NameFinderTrainer* dengan nama ***id-ner-actor.bin.***

5.5.3.2 *TokenizerTrainer*

Setelah *file tagging* sudah selesai dibuat, maka akan dilakukan *training* untuk mengubah menjadi *file model* yang dapat digunakan, dengan modul *TokenizerTrainer.* Berikut adalah deskripsi argumen yang terdapat dalam modul pada Kode Program 5.16.

```
$ 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 5.16 Fungsi *command TokenizerTrainer*

Secara urut ketikkan script berikut pada *command prompt*, seperti pada Kode Program 5.17:

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

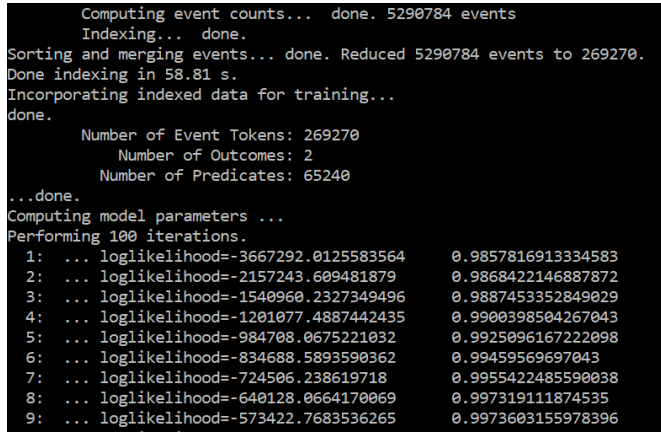
Kode Program 5.17 Struktur *query TokenizerTrainer*

Berikut adalah potongan *script* pada *command prompt Opennlp* pada gambar 5.3

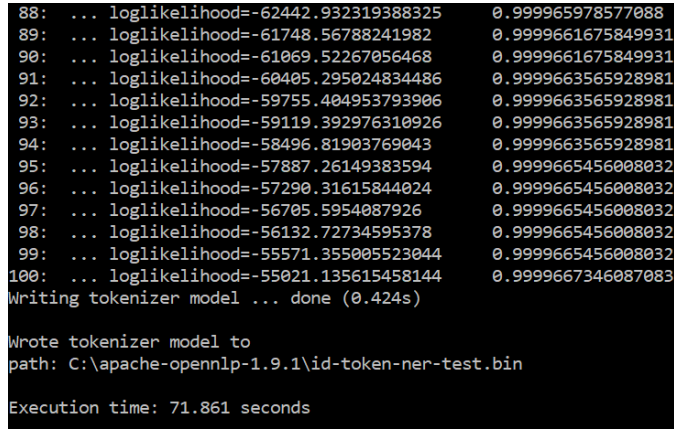
```
C:\apache-opennlp-1.9.1>opennlp TokenizerTrainer -model id-token-ner.bin
-alphaNumOpt 2 -lang id -data token_all.train -encoding UTF-8
```

Gambar 5.5 Contoh tampilan *script TokenizerTrainer* pada *console*

Kemudian tekan *enter* untuk melakukan proses *training* sampai selesai. Berikut tampilan *console* hasil *training TokenizerTrainer* pada Gambar 5.6 dan Gambar 5.7.



Gambar 5.6 Proses *training model token (1)*

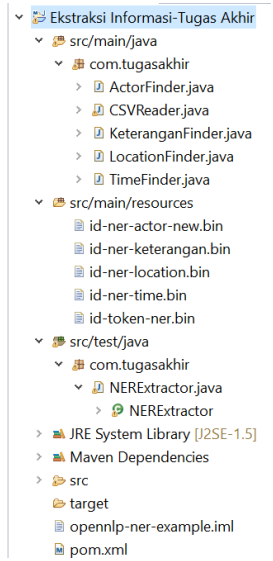


Gambar 5.7 Proses *training model token (2)*

Pada folder *command prompt* dijalankan akan muncul *output file* dari proses *training*, yaitu model dari *NameFinderTrainer* dengan nama ***id-token-ner.bin.***

5.5.4 *Parser* dengan *OpenNLP API*

Hasil dari *model training* pada proses sebelumnya akan menghasilkan 4 *file model NER* dan 1 *file model token* yang akan digunakan untuk mengekstrak informasi pada *tweet* yang sudah di *crawl.* Proses *parsing* menggunakan kode program berbasis *Java.* Langkah pertama adalah membuat fungsi yang dapat memanggil model menggunakan *NameFinder API* dan *Tokenizer API.* Berikut adalah potongan kode program untuk memanggil *model* untuk *parsing* dan struktur *project* pada Gambar 5.8 dan Kode Program 5.20-Kode Program 5.22. Tahap ini terkait dengan penjelasan pada sub-bab 4.5.4.



Gambar 5.8 Struktur *project NER Parser* pada *IDE Eclipse*

```
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 *Dependencies* untuk *NameFinder API*

```
1.  public class TimeFinder {
2.  public static ArrayList<String> time = new ArrayLis
    t();
3.      public ArrayList<String> getTweets() {
4.          return time;
5.      }
6.      public void setTweets(ArrayList<String> time) {
7.          this.time = time;
8.      }
9.      public void findTime(String paragraph) throws I
    OException {
10.         InputStream inputStreamNameFinder = getClas
    s().getResourceAsStream("/id-ner-time.bin");
11.         TokenNameFinderModel model = new TokenNameF
    inderModel(inputStreamNameFinder);
12.         NameFinderME nameFinder = new NameFinderME(
    model);
13.         StringBuilder sb = new StringBuilder();
14.
15.             String[] tokens = tokenize(paragraph);

16.
17.             Span nameSpans[] = nameFinder.find(toke
    ns);
18.
19.             for(Span s: nameSpans) {
20.                 for(int i = s.getStart(); i < s.get
    End(); i++ ) {
21.                     sb.append(tokens[i] + " ");
22.                 }
23.                 time.add(sb.toString());
24.                 sb.setLength(0);
25.             }
26.     }
27.     public void clearArray() {
28.         time.clear();
29.     }
30.     public String[] tokenize(String sentence) throw
    s IOException{
31.         InputStream inputStreamTokenizer = getClass
    ().getResourceAsStream("/id-token-ner.bin");
32.         TokenizerModel tokenModel = new TokenizerMo
    del(inputStreamTokenizer);
33.         TokenizerME tokenizer = new TokenizerME(tok
    enModel);
34.         return tokenizer.tokenize(sentence);
35.     }
36. }
```

Kode Program 5.19 Kode program *NameFinder API* untuk model *Time*

```
1.  public class ActorFinder {
2.      public static ArrayList<String> actor = new ArrayList();
3.      public ArrayList<String> getTweets() {
4.          return actor;   }
5.      public void setTweets(ArrayList<String> actor) {
6.          this.actor = actor; }
7.      static int wordcount(String string)  {
8.        int count=0;
9.          char ch[]= new char[string.length()];
10.         for(int i=0;i<string.length();i++) {
11.             ch[i]= string.charAt(i);
12.             if( ((i>0)&&(ch[i]!=' ')&&(ch[i-
    1]==' ')) || ((ch[0]!=' ')&&(i==0)) )  count++;
13.         }  return count;
14.     }
15.     public void findActor(String paragraph) throws IOException
    {
16.         InputStream inputStream = getClass().getResourceAsStrea
    m("/id-ner-actor-new.bin");
17.         TokenNameFinderModel model = new TokenNameFinderModel(i
    nputStream);
18.         NameFinderME nameFinder = new NameFinderME(model);
19.         StringBuilder sb = new StringBuilder();
20.         String[] tokens = tokenize(paragraph);
21.         Span nameSpans[] = nameFinder.find(tokens);
22.         int v = 0;
23.         for(Span s: nameSpans) {
24.             for(int i = s.getStart(); i < s.getEnd(); i++ ) {
25.                 sb.append(tokens[i] + " "); }
26.             int cwords = wordcount(sb.toString());
27.             sb.setLength(0);
28.             for(int i = s.getStart(); i < s.getEnd(); i++ ) {
29.             if(cwords > 1) {
30.             if(v == 0) {
31.                 sb.append(tokens[i] + "_");
32.                     v++;
33.             } else if(cwords == 3 && v == 1) {
34.                 sb.append(tokens[i] + "_");
35.                     v++;
36.             } else {
37.                 sb.append(tokens[i]); }
38.             } else {
39.                 sb.append(tokens[i]); }
40.             }
41.             actor.add(sb.toString());
42.             sb.setLength(0); }
43.             V = 0;  }  }
44.     public void clearArray() {
45.         actor.clear();
46.     }
47.     public String[] tokenize(String sentence) throws IOExceptio
    n {
48.         InputStream inputStreamTokenizer = getClass().getResour
    ceAsStream("/id-token-ner.bin");
49.         TokenizerModel tokenModel = new TokenizerModel(inputStr
    eamTokenizer);
50.         TokenizerME tokenizer = new TokenizerME(tokenModel);
51.         return tokenizer.tokenize(sentence);   }
52. }
```

Kode Program 5.20 Kode program *NameFinder API* untuk model *Actor*

```
1.  public class LocationFinder {
2.      public static ArrayList<String> actor = new ArrayList();
3.      public ArrayList<String> getTweets() {
4.          return location;   }
5.      public void setTweets(ArrayList<String> actor) {
6.          this.location = location; }
7.        static int wordcount(String string)  {
8.        int count=0;
9.          char ch[]= new char[string.length()];
10.         for(int i=0;i<string.length();i++) {
11.             ch[i]= string.charAt(i);
12.             if( ((i>0)&&(ch[i]!=' ')&&(ch[i-
    1]==' ')) || ((ch[0]!=' ')&&(i==0)) )  count++;
13.         }  return count;
14.     }
15.     public void findLocation(String paragraph) throws IOExcepti
    on {
16.         InputStream inputStream = getClass().getResourceAsStrea
    m("/id-ner-location-baru.bin");
17.         TokenNameFinderModel model = new TokenNameFinderModel(i
    nputStream);
18.         NameFinderME nameFinder = new NameFinderME(model);
19.         StringBuilder sb = new StringBuilder();
20.         String[] tokens = tokenize(paragraph);
21.         Span nameSpans[] = nameFinder.find(tokens);
22.         int v = 0;
23.         for(Span s: nameSpans) {
24.             for(int i = s.getStart(); i < s.getEnd(); i++ ) {
25.                 sb.append(tokens[i] + " "); }
26.             int cwords = wordcount(sb.toString());
27.             sb.setLength(0);
28.             for(int i = s.getStart(); i < s.getEnd(); i++ ) {
29.             if(cwords > 1) {
30.             if(v == 0) {
31.                 sb.append(tokens[i] + "_");
32.                     v++;
33.             } else if(cwords == 3 && v == 1) {
34.                 sb.append(tokens[i] + "_");
35.                     v++;
36.             } else {
37.                 sb.append(tokens[i]); }
38.             } else {
39.                 sb.append(tokens[i]); }
40.             }
41.             location.add(sb.toString());
42.             sb.setLength(0)
43.             V = 0;  }  }
44.     public void clearArray() {
45.         location.clear();
46.     }
47.     public String[] tokenize(String sentence) throws IOExceptio
    n {
48.         InputStream inputStreamTokenizer = getClass().getResour
    ceAsStream("/id-token-ner.bin");
49.         TokenizerModel tokenModel = new TokenizerModel(inputStr
    eamTokenizer);
50.         TokenizerME tokenizer = new TokenizerME(tokenModel);
51.         return tokenizer.tokenize(sentence);   }
52. }
```

Kode Program 5.21 Kode program *NameFinder API* untuk model *Location*

```
1.  public class KeteranganFinder {
2.  public static ArrayList<String> keterangan = new ArrayList();
3.      public ArrayList<String> getTweets() {
4.          return keterangan; }
5.      public void setTweets(ArrayList<String> keterangan) {
6.          this.keterangan = keterangan; }
7.      static int wordcount(String string) {
8.          int count=0;
9.          char ch[]= new char[string.length()];
10.         for(int i=0;i<string.length();i++) {
11.             ch[i]= string.charAt(i);
12.             if( ((i>0)&&(ch[i]!=' ')&&(ch[i-
    1]==' ')) || ((ch[0]!=' ')&&(i==0)) )
13.                 count++; } return count;  }
14.     public void findKeterangan(String paragraph) throws IOExcep
    tion {
15.         InputStream inputStream = getClass().getResourceAsStrea
    m("/id-ner-keterangan-groundup4.bin");
16.         TokenNameFinderModel model = new TokenNameFinderModel(i
    nputStream);
17.         NameFinderME nameFinder = new NameFinderME(model);
18.         StringBuilder sb = new StringBuilder();
19.         String[] tokens = tokenize(paragraph);
20.         Span nameSpans[] = nameFinder.find(tokens);
21.         int v = 0;
22.         for(Span s: nameSpans) {
23.             for(int i = s.getStart(); i < s.getEnd(); i++ ) {
24.                 sb.append(tokens[i] + " "); }
25.             int cwords = wordcount(sb.toString());
26.             sb.setLength(0);
27.             for(int i = s.getStart(); i < s.getEnd(); i++ ) {
28.                 if(cwords > 1) {
29.                     if(v == 0) {
30.                         sb.append(tokens[i] + "_");
31.                         v++;
32.                     } else if(cwords == 3 && v == 1) {
33.                         sb.append(tokens[i] + "_");
34.                         v++;
35.                     } else {
36.                         sb.append(tokens[i]);  }
37.                 } else {
38.                     sb.append(tokens[i]);  }   }
39.         keterangan.add(sb.toString());
40.         sb.setLength(0);
41.         v = 0;  }  }
42.     public void clearArray() {
43.         keterangan.clear();
44.     }
45.     public String[] tokenize(String sentence) throws IOExceptio
    n{
46.         InputStream inputStreamTokenizer = getClass().getResour
    ceAsStream("/id-token-ner.bin");
47.         TokenizerModel tokenModel = new TokenizerModel(inputStr
    eamTokenizer);
48.         TokenizerME tokenizer = new TokenizerME(tokenModel);
49.         return tokenizer.tokenize(sentence);
50.     }
51. }
```

Kode Program 5.22 Kode program *NameFinder API* untuk model Keterangan

Proses selanjutnya adalah data hasil *crawl* dimasukkan dengan fungsi *CSVReader* dan dimasukkan ke dalam *ArrayList*, setelah itu data yang sudah masuk list akan di cek satu per satu apakah mengandung entitas dengan fungsi *NameFinder API* yang sudah dibuat sebelumnya. Langkah selanjutnya adalah memanggil *ontology* yang telah dibuat pada sub-bab 5.4. *Ontology* tersebut akan disimpan pada *array* yang berisi kolom *class* dan kolom *instance*. Berikut potongan kode program *IDE Eclipse* pada Kode Program 5.23. Tahap ini terkait dengan penjelasan pada sub-bab 4.5.4.

Ontology yang sudah dibuat 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 menjalankan hal tersebut dengan membuat fungsi *load ontology* pada *row* 40.

Untuk menentukan *class* dari *instance* keterangan yang telah diparsing, maka membuat fungsi untuk menentukan *class* dari sebuah *instance* yang didapatkan seperti pada *row* 72.

Pada *row* 123 merupakan cara untuk melakukan load file tweet yang telah dibersihkan

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 *row* 128.

Tahap terakhir adalah dengan melakukan *insert* ekstraksi data yang telah didapatkan ke dalam database *mySQL*. Untuk melakukan insert tersebut dapat dilihat pada *row* 81.

```
1.  package com.tugasakhir;
2.  import java.sql.*;
3.  import java.io.IOException;
4.  import java.util.ArrayList;
5.  import org.apache.jena.ontology.Individual;
6.  import org.apache.jena.ontology.OntClass;
7.  import org.apache.jena.ontology.OntDocumentManager;
8.  import org.apache.jena.ontology.OntModel;
9.  import org.apache.jena.rdf.model.Model;
10. import org.apache.jena.rdf.model.ModelFactory;
11. import org.apache.jena.util.FileManager;
12. import org.apache.jena.util.iterator.ExtendedIterator;

13. import org.junit.Test;
14.
15. public class NERExtractor {
16.     private String dataActor = "";
17.     private String dataLoc = "";
18.     private String dataKet = "";
19.     private String dataTime = "";
20.
21.     TimeFinder timeFinder = new TimeFinder();
22.     LocationFinder locFinder = new LocationFinder();
23.     KeteranganFinder ketFinder = new KeteranganFinder()
    ;
24.     ActorFinder actorFinder = new ActorFinder();
25.
26.     ArrayList<String> actor = actorFinder.getTweets();

27.     ArrayList<String> location = locFinder.getTweets();

28.     ArrayList<String> keterangan = ketFinder.getTweets(
    );
29.     ArrayList<String> time = timeFinder.getTweets();
30.
31.     private int x;
32.     public static String[][] ontoDomain = new String[15
    0][2];
33.
34.     public static String getOntoDomain(int x, int y) {

35.         return ontoDomain[x][y];
36.     }
37. static String defaultNameSpace = "http://semanticweb.or
    g/ontologies";
38.     Model schema = null;
39.     OntDocumentManager mgr = new OntDocumentManager();
```

```
40.  private void loadontology() throws IOException {
41.          schema = ModelFactory.createOntologyModel();
42.          java.io.InputStream inschema = FileManager.get(
     ).open("C:/Users/benny/Documents/tugasakhir.owl");
43.          schema.read(inschema, defaultNameSpace);
44.          setX(0);
45.
46.          ExtendedIterator classes = ((OntModel) schema).
     listClasses();
47.          while (classes.hasNext()) {
48.            OntClass thisClass = (OntClass)classes.next()
     ;
49.            String ontoclass = thisClass.toString();
50.            String value = ontoclass.substring(30);
51.            if(!value.equals("Kejahatan")) {
52.                ExtendedIterator instances = thisClass.li
     stInstances();
53.                while (instances.hasNext()) {
54.                  Individual thisInstance = (Individual)
     instances.next();
55.                  String ontoinst = thisInstance.toString
     ();
56.                  ontoDomain[getX()][0] = ontoclass.subst
     ring(30);
57.                  ontoDomain[getX()][1] = ontoinst.substr
     ing(30);
58.                  setX(getX() + 1);
59.                }
60.            }
61.          }
62.      }
63.
64.      public int getX() {
65.          return x;
66.      }
67.
68.      public void setX(int x) {
69.          this.x = x;
70.      }
71.
72.      public String compareTo(String input) {
73.          String output = null;
74.          for(int i=0; i < getX(); i++) {
75.              if(input.equals(getOntoDomain(i,1))) {
76.                  output = getOntoDomain(i,0);
77.              }
78.          }
79.          return output;
80.      }
```

```
81.  public void insertSQL(String table, String column, String i
     d, String data, String category) {
82.          try {
83.            // create a mysql database connection
84.            String myDriver = "com.mysql.jdbc.Driver";
85.            String myUrl = "jdbc:mysql://localhost:3306/tugas
     akhir_kejahatan2?useSSL=false";
86.            Class.forName(myDriver);
87.            Connection conn = DriverManager.getConnection(myU
     rl, "root", "");
88.
89.            // the mysql insert statement
90.            String query;
91.
92.            if(category != null) {
93.                 query = " insert into " + table + " (id, "+
     column +", category)"
94.                       + " values (?, ?, ?)";
95.            } else {
96.                 query = " insert into " + table + " (id, "+
     column +")"
97.                  + " values (?, ?)";
98.            }
99.
100.           // create the mysql insert preparedstatement
101.           PreparedStatement preparedStmt = conn.prepareStat
      ement(query);
102.           preparedStmt.setString (1, id);
103.           preparedStmt.setString (2, data);
104.           if(category != null) {
105.               preparedStmt.setString (3, category);
106.           }
107.
108.           // execute the preparedstatement
109.           preparedStmt.execute();
110.
111.           conn.close();
112.         }
113.         catch (Exception e) {
114.           System.err.println("Got an exception!");
115.           System.err.println(e.getMessage());
116.         }
117.     }
118.
119.     @Test
120.     public void nameFinderTest() throws Exception {
121.         loadontology();
122.
123.         CSVReader bacafile = new CSVReader();
124.         bacafile.baca("D:/S1 Sistem Informasi ITS/Semester
     8/TUGAS AKHIR/Cleaned/cleaned_gabung8.csv");
125.         ArrayList<String> tweets = bacafile.getTweets();
126.
127.         int x = 1;
```

```
for(String token : tweets) {
                actorFinder.findActor(token);
                locFinder.findLocation(token);
                timeFinder.findTime(token);
                ketFinder.findKeterangan(token);
                String id = "tweet" + x;
                for(int i=0; i < actor.size(); i++ ) {
                    dataActor = actor.get(i);
                    if(dataActor.length() > 2) {
                        System.out.println(":Tweet" + x + " :ha
sActor :" + dataActor + "." + compareTo(dataActor.toString(
)));
                        insertSQL("actor", "actorName", id, dat
aActor, compareTo(dataActor.toString()));
                    }
                }
                for(int i=0; i < location.size(); i++ ) {
                    dataLoc = location.get(i);
                    if(dataLoc.length() > 2) {
                        System.out.println(":Tweet" + x + " :ha
sLocation :" + location.get(i) + ".");
                        insertSQL("location", "locationName", i
d, dataLoc, null);
                    }  }
                for(int i=0; i < keterangan.size(); i++ ) {
                    dataKet = keterangan.get(i);
                    if(compareTo(dataKet.toString()) != null) {

                        System.out.println(":Tweet" + x + " :ha
sKeterangan :" + dataKet + "." + compareTo(dataKet.toString
()));
                        insertSQL("keterangan", "keteranganName
", id, dataKet, compareTo(dataKet.toString()));
                    }  }
                for(int i=0; i < time.size(); i++ ) {
                    dataTime = time.get(i);
                    if(dataActor.length() > 1 || dataLoc.length
() > 1 || dataKet.length() > 1  ) {
                        if(dataTime.length() > 2) {
                            System.out.println(":Tweet" + x + "
 :hasTime :" + time.get(i) + ".");
                            insertSQL("time", "time", id, dataT
ime, null);
                        }  }  }
                dataActor = "";
                dataLoc = "";
                dataKet = "";
                dataTime = "";
                actorFinder.clearArray();
                locFinder.clearArray();
                timeFinder.clearArray();
                ketFinder.clearArray();
                x++;
            }  }   }
```

Kode Program 5.23 *NER Extractor*

5.6 Pengujian Model

5.6.1 *Split Training data* dan *Test data*

Pada tahap pengujian model *NER* yang sudah dibuat dengan menggunakan modul *NameFinderTrainer,* data model akan dibagi menjadi 2 bagian terlebih dahulu sebelum dilakukan pengujian. Bagian data *training* mempunyai proporsi 80 dari keseluruhan data model, sedangkan data *testing* mempunyai proporsi 20% dari keseluruhan data model.

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 5.24.

```
1.  import pandas as pd
2.  import numpy as np
3.
4.  #load file ascii (menghilangkan kanji, mandarin dll.)
5.  with open("C:/apache-opennlp-
    1.9.1/TESTMODEL/train_new_lokasi_carabaru.train", encoding
    ='ascii', errors='ignore') as infile:
6.      df = pd.read_csv(infile, header=None)
7.
8.  df.head(5)
9.  len(df)
10.
11. dfTes = pd.DataFrame(np.random.randn(100, 2))
12. msk = np.random.rand(len(df)) < 0.8
13. train = df[msk]
14. test = df[~msk]
15.
16. len(train)
17. len(test)
18. #Save training data
19. df_train = pd.DataFrame(data=train, columns=None)
20. df_train.head(5)
21. df_train.to_csv('C:/apache-opennlp-
    1.9.1/TESTMODEL/train_new_lokasi_carabaru_model80.train',
    sep='\t', encoding='utf-8', header=False, index=False)
22. #Save testing data
23. df_test = pd.DataFrame(data=test, columns=None)
24. df_test.head(5)
25. df_test.to_csv('C:/apache-opennlp-
    1.9.1/TESTMODEL/train_new_lokasi_carabaru_test20.test', se
    p='\t', encoding='utf-8', header=False, index=False)
```

Kode Program 5.24 *Split training testing data*

File training dengan proporsi 80% akan diolah menjadi *file model* dengan menggunakan modul *NameFinderTrainer* yang ada pada sub-bab 5.5.3.1, dan *file test* dengan proporsi 20% akan digunakan sebagai *data test* untuk evaluasi performa *model.*

5.6.2 *Evaluation API OpenNLP TokenNameFinderTrainer*

Pada tahap ini akan dilakukan pengujian kualitas model dengan menggunakan modul *Evaluation* dari *Opennlp TokenNameFinderTrainer*. Berikut adalah deskripsi argumen yang terdapat dalam modul pada Kode Program 5.25.

```
$ 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 5.25 Fungsi *command TokenNameFinderEvaluator*

Data model yang akan digunakan adalah data model dengan proporsi 80% yang sudah diolah, dan data yang akan diuji adalah data *testing* dengan proporsi 20%.

Secara urut ketikkan script berikut pada *command prompt*, seperti pada Kode Program 5.26.

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

Kode Program 5.26 *Script evaluation API TokenNameFinderTrainer*

Berikut adalah potongan *script* pada *command prompt Opennlp* pada Gambar 5.9.

```
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 5.9 Contoh tampilan *script TokenNameFinderEvaluator* pada *console*

Kemudian tekan *enter* untuk melakukan proses *training* sampai selesai.

5.7 Visualisasi dengan Tableau

Hasil *parsing OpenNLP API* akan menghasilkan 4 tabel, yaitu tabel *Actor*, *Location*, *Time*, dan Keterangan. Untuk menampilkan hasil tersebut menjadi informasi yang berguna, diperlukan *join table* sebelum diolah dalam bentuk visualisasi. Proses *join table* menggunakan *Tableau* yang mengambil *data* langsung dari *MySQL*. Berikut pada Gambar 5.10-Gambar 5.13 merupakan tangkapan gambar dari tabel *MySQL Actor*, Keterangan, *Location*, dan *Time*.

id	actorName	category
tweet1	truk_tangki_pertamina	Truk_Tangki_Pertamina
tweet4	truk_tangki_pertamina	Truk_Tangki_Pertamina
tweet5	truk_tangki_pertamina	Truk_Tangki_Pertamina
tweet7	truk_tangki_pertamina	Truk_Tangki_Pertamina
tweet8	truk_pertamina	Truk_Tangki_Pertamina
tweet9	truk_tangki_pertamina	Truk_Tangki_Pertamina
tweet10	mobil_tangki_pertamina	Mobil_Tangki_Pertamina
tweet11	truk_tangki_pertamina	Truk_Tangki_Pertamina
tweet12	truk_pertamina	Truk_Tangki_Pertamina
tweet13	truk_pertamina	Truk_Tangki_Pertamina
tweet14	mobil_tangki_pertamina	Mobil_Tangki_Pertamina
tweet15	truk_tangki_pertamina	Truk_Tangki_Pertamina
tweet17	truk_pertamina	Truk_Tangki_Pertamina
tweet19	mobil_tangki_pertamina	Mobil_Tangki_Pertamina
tweet21	mobil_tangki_pertamina	Mobil_Tangki_Pertamina

Gambar 5.10 Tabel *Actor*

id	keteranganName	category
tweet1	pembajakan	Blokade
tweet2	pembajakan	Blokade
tweet3	pembajakan	Blokade
tweet4	pembajakan	Blokade
tweet5	pembajakan	Blokade
tweet6	pembajakan	Blokade
tweet7	pembajakan	Blokade
tweet9	pembajakan	Blokade
tweet10	pembajakan	Blokade
tweet13	pembajakan	Blokade
tweet14	pembajakan	Blokade
tweet15	pembajakan	Blokade
tweet18	pembajakan	Blokade
tweet19	pembajakan	Blokade
tweet20	pembajakan	Blokade

Gambar 5.11 Tabel Keterangan

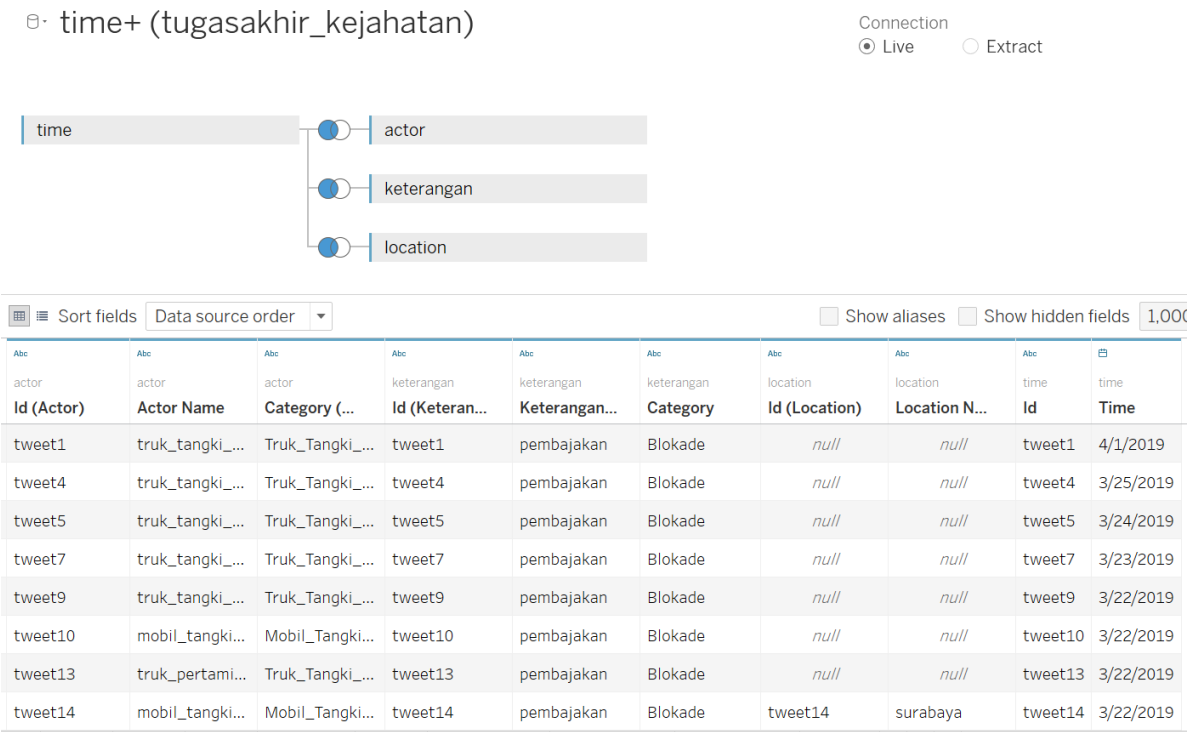
id	locationName
tweet14	surabaya
tweet33	metro
tweet34	metro
tweet43	metro
tweet43	metro
tweet44	metro
tweet49	metro
tweet56	metro
tweet64	jabar
tweet72	metro
tweet82	bogor
tweet82	bogor
tweet83	bogor
tweet83	bogor
tweet98	metro

Gambar 5.12 Tabel *Location*

id	time
tweet1	2019-04-01
tweet2	2019-03-30
tweet3	2019-03-28
tweet4	2019-03-25
tweet5	2019-03-24
tweet6	2019-03-23
tweet7	2019-03-23
tweet8	2019-03-23
tweet9	2019-03-22
tweet10	2019-03-22
tweet11	2019-03-22
tweet12	2019-03-22
tweet13	2019-03-22
tweet14	2019-03-22
tweet15	2019-03-22

Gambar 5.13 Tabel *Time*

Berikut pada Gambar 5.14 merupakan gambar dari *join table* ke 4 tabel tersebut dengan menggunakan tabel *time* sebagai *pivot.*

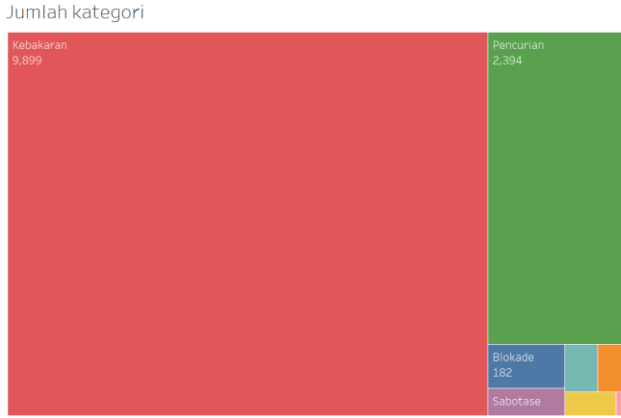


actor Id (Actor)	actor Actor Name	actor Category (...	keterangan Id (Keteran...	keterangan Keterangan...	keterangan Category	location Id (Location)	location Location N...	time Id	time Time
tweet1	truk_tangki_...	Truk_Tangki_...	tweet1	pembajakan	Blokade	*null*	*null*	tweet1	4/1/2019
tweet4	truk_tangki_...	Truk_Tangki_...	tweet4	pembajakan	Blokade	*null*	*null*	tweet4	3/25/2019
tweet5	truk_tangki_...	Truk_Tangki_...	tweet5	pembajakan	Blokade	*null*	*null*	tweet5	3/24/2019
tweet7	truk_tangki_...	Truk_Tangki_...	tweet7	pembajakan	Blokade	*null*	*null*	tweet7	3/23/2019
tweet9	truk_tangki_...	Truk_Tangki_...	tweet9	pembajakan	Blokade	*null*	*null*	tweet9	3/22/2019
tweet10	mobil_tangki...	Mobil_Tangki...	tweet10	pembajakan	Blokade	*null*	*null*	tweet10	3/22/2019
tweet13	truk_pertami...	Truk_Tangki_...	tweet13	pembajakan	Blokade	*null*	*null*	tweet13	3/22/2019
tweet14	mobil_tangki...	Mobil_Tangki...	tweet14	pembajakan	Blokade	tweet14	surabaya	tweet14	3/22/2019

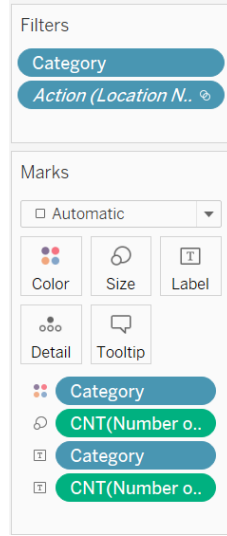
Gambar 5.14 *Join Table* untuk visualisasi dengan *Tableau*

Setelah *join table* berhasil dilakukan, selanjutnya dilakukan proses visualisasi setiap topik yang sudah dibuat. Setiap hasil topik yang didapatkan kemudian disimpan dalam 1 basis data, yang kemudian dilakukan pembuatan visualisasi dengan Tableau. Adapun bentuk visualisasi yang digunakan adalah *Area Chart, Bar Chart,* dan *Bubble Chart.*

Visualisasi dengan *Area Chart* berguna untuk menampilkan kuantitas suatu kejadian dengan volume berbentuk kotak, dicontohkan pada Gambar 5.15 yang menggambarkan jumlah kategori kejahatan/gangguan keamanan yang terjadi. Pada visualisasi ini menggunakan *Marks category* dan *count number of records*, serta menggunakan *filters category*. Dapat dilihat pada Gambar 5.16.

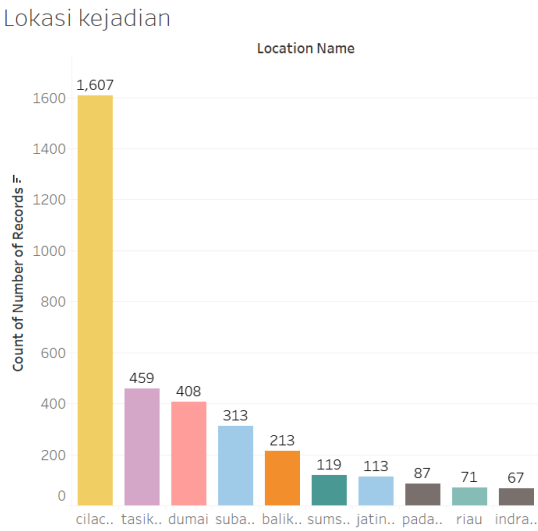


Gambar 5.15 Contoh Visualisasi *Area Chart*

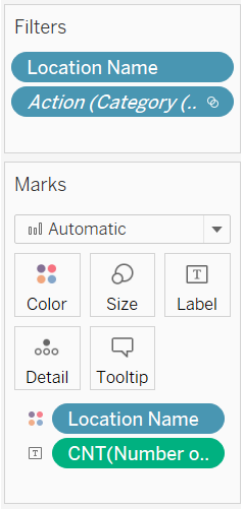


Gambar 5.16 *Marks* dan *Filters Area Chart*

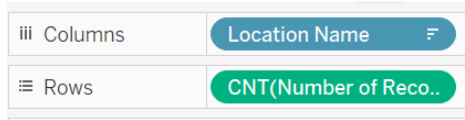
Visualisasi dengan *Bar Chart* berguna untuk menampilkan hasil visualisasi dengan menampilkan perbandingan kuantitas dalam bentuk *bar*, dicontohkan pada Gambar 5.17 yang menggambarkan jumlah lokasi kejadian berdasarkan kejadian yang terjadi pada lokasi tersebut. Pada visualisasi ini menggunakan *Marks location name* dan *count number of records*, serta menggunakan *filters location name.* Serta menggunakan *Columns location name* dan *Rows count number of records.* Dapat dilihat pada Gambar 5.18 dan Gambar 5.19.



Gambar 5.17 Contoh Visualisasi *Bar Chart*

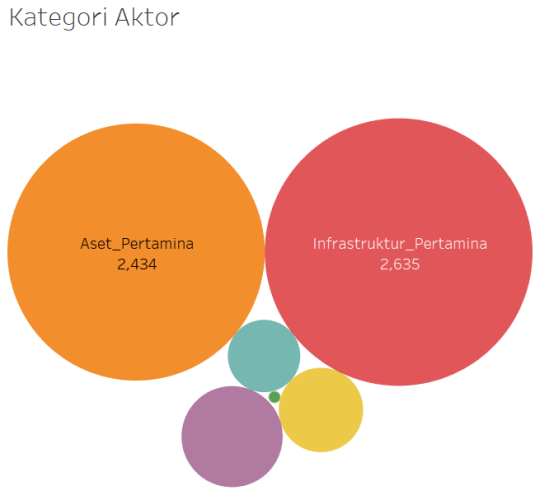


Gambar 5.18 *Marks* dan *Filters Bar Chart*

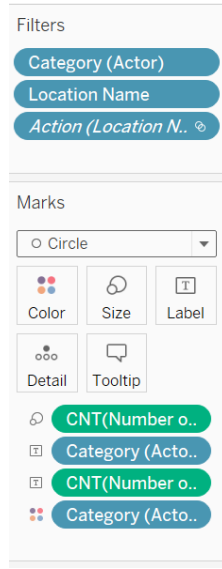


Gambar 5.19 *Columns* dan *Rows Bar Chart*

Visualisasi dengan *Bubble Chart* berguna untuk menampilkan hasil visualisasi dengan menampilkan perbandingan kuantitas dalam bentuk *bubble*, dicontohkan pada Gambar 5.20 yang menggambarkan jumlah kategori aktor berdasarkan kejadian. Pada visualisasi ini menggunakan *Marks category (actor)* dan *count number of records*, serta menggunakan *filters location name* dan *category (actor).* Dapat dilihat pada Gambar 5.20.



Gambar 5.20 Contoh Visualisasi *Bubble Chart*



Gambar 5.21 *Marks* dan *Filters Bubble Chart*

.

BAB VI
HASIL DAN PEMBAHASAN

Bab ini akan menjabarkan hasil dari implementasi dan pengujian yang dilakukan

6.1 Hasil Pengumpulan Data

Dengan menggunakan *TweetScraper* dan *query* yang sudah ditulis pada Kode Program 5.2, didapatkan total *tweet* sebesar 42,055 *tweet* untuk ke 8 *keywords*. Berikut adalah rincian jumlah *tweet* yang didapatkan pada periode tahun 2009-2019 pada Tabel 6.1.

Tabel 6.1 *Keywords* dan jumlah *tweets* yang didapatkan

Keywords	**Jumlah *Tweets***
Blokade Pertamina	818
Demonstrasi Pertamina	1,108
Pencurian Pertamina	12,018
Pembunuhan Pertamina	608
Penganiayaan Pertamina	524
Terorisme Pertamina	244
Sabotase Pertamina	928
Kebakaran Pertamina	25,807
Total	**42,055**

Berikut adalah hasil *crawling* kedua pada Tabel 6.2.

Tabel 6.2 *Keywords* tambahan untuk penyusun *model*

Keywords	**Jumlah *Tweets***
Kilang Pertamina	84,604
Tangki Pertamina	59,231
Truk Pertamina	65,126
Total	**208,961**

6.2 Hasil *Data Preprocessing*

Tahap ini dilakukan dengan melakukan *case folding*, penghapusan penghapusan *tweet* duplikat. Hasil *crawl* akan diproses untuk membersihkan *tweet* dari simbol, tanda baca, *link, rt*. Pada Tabel 6.3 merupakan contoh dari *preprocessing tweets.*

Tabel 6.3 Contoh hasil *preprocessing tweets*

Sebelum *preprocessing*	**Setelah *preprocessing***
# breakingNews via WAG TCT_Bandung : Terjadi kebakaran alang2 di Jl Soetta (Seberang Pertamina Gedebage). Saat ini Petugas Damkar sedang berusaha mengendalikan kebakaran tsb pic.twitter.com/cZpDBiqnp8	breakingnews via wag tctbandung terjadi kebakaran alang2 di jl soetta seberang pertamina gedebage saat ini petugas damkar sedang berusaha mengendalikan kebakaran tsb 2018-05-28

Berikut rincian jumlah *preprocessing tweets* pada Tabel 6.4.

Tabel 6.4 Jumlah *tweets* per *keywords* setelah *preprocessing*

Keywords	**Jumlah *Tweets***	**Jumlah setelah *preprocessing***
Blokade Pertamina	818	298
Demonstrasi Pertamina	1,108	307
Pencurian Pertamina	12,018	2,343
Pembunuhan Pertamina	608	197
Penganiayaan Pertamina	524	83
Terorisme Pertamina	244	80
Sabotase Pertamina	928	313
Kebakaran Pertamina	25,807	9,597
Total	**42,055**	**13,219**

Terdapat selisih sebesar 28,837 *rows tweet* setelah dilakukan *data preprocessing.* Dapat dilihat bahwa jumlah *tweet* setelah dilakukan *preprocessing* menyusut menjadi 13,219. *Data* setelah *preprocessing* akan digunakan juga sebagai penyusun *model*, sedangkan jumlah *minimal rows* untuk penyusun *model* menggunakan *OpenNLP* adalah 15,000. Oleh karena itu hasil *crawling* kedua akan ditambahkan agar lebih dari kebutuhan *minimal.* Berikut pada Tabel 6.5 merupakan jumlah *tweets crawling* kedua setelah dilakukan *preprocessing.*

Tabel 6.5 Jumlah *tweets crawling kedua* setelah *preprocessing*

Keywords	**Jumlah *Tweets***	**Jumlah setelah *preprocessing***
Kilang Pertamina	84,604	21,906
Tangki Pertamina	59,231	20,359
Truk Pertamina	65,126	19,742
Total	**208,961**	**62,007**

Berikut pada Tabel 6.6 merupakan rincian jumlah *tweets* yang didapatkan per tahun.

Tabel 6.6 Jumlah *tweets* per tahun

Tahun	**Jumlah *Tweets***
2019	389
2018	438
2017	325
2016	626
2015	464
2014	2,095
2013	3,227
2012	2,027
2011	3,058
2010	490
2009	80
Total	**13,219**

6.3 Hasil Ekstraksi Informasi

Tahap ini akan menghasilkan *model NER Tagging* dan menghasilkan informasi menggunakan *model NER* yang sudah dibuat

6.3.1 *NER Tagging*

Hasil dari tahap ini adalah *model* yang siap dilakukan *training* menggunakan *NameFinderTrainer* yang dijelaskan pada sub-bab 5.5.3.1. Berikut pada Tabel 6.7 merupakan jumlah *rows* dari setiap *model* sebelum dan sesudah dilakukan pengecekan *label* per *row* pada *model.*

Tabel 6.7 Komposisi *model NER*

Model	**Jumlah *Tweets***	**Jumlah setelah *check model***
Actor	75,226	34,289
Location	75,226	19,722
Keterangan	75,226	24,835
Time	75,226	75,226

6.3.2 *Parser* dengan *OpenNLP API*

Hasil dari tahap ini adalah *data* dari sebuah *tweet* yang mengandung *Actor, Location, Time,* dan Keterangan terjadinya suatu kejadian kejahatan/gangguan keamanan. Tahap ini juga melakukan penghapusan *tweet* yang tidak memiliki informasi, yaitu *tweet* yang tidak memiliki *Actor, Location,* dan Keterangan. Berikut pada Tabel 6.8 merupakan contoh hasil *parsing* suatu *tweet.*

Tabel 6.8 Contoh hasil *parser OpenNLP API*

Tweet	***Actor***	***Location***	**Keterangan**	***Time***	***ID***
humasskkmigas pertamina ep fokus bahas investigasi dan penanggulangan insiden kebakaran sumur di aceh timur 2018-04-26	sumur	aceh	kebakaran	2018-04-26	Tweet235
pipa pertamina bocor penyebab kebakaran tak terdaftar ilegal 2018-04-19	pipa_pertamina	*null*	kebakaran	2018-04-19	Tweet423
halahh tasikkpertamina atasi kebakaran yang terjadi akibat upaya pencurian di terminal bbm tasikmalaya 2013-08-01	terminal_bbm	tasikmalaya	kebakaran pencurain	2013-08-01	Tweet1230

Jumlah *tweets* yang digunakan sebagai *data sample* untuk di ekstrak informasi yang terkandung di dalam setiap *tweet* dapat adalah seluruh hasil *crawl* yang sudah dilakukan *data preprocessing*, yaitu sejumlah 13,219 *rows.*

Setelah dilakukan proses *parsing* menggunakan *OpenNLP* dan *Jena Ontology*, didapatkan jumlah *tweet* per kategori gangguan keamanan pada Tabel 6.9, serta jumlah *tweet* per kategori aktor pada Tabel 6.10.

Tabel 6.9 Jumlah *tweets* per kategori gangguan keamanan

Kategori gangguan keamanan	**Jumlah *Tweets***
Kebakaran	9,896
Pencurian	2,394
Blokade	182
Sabotase	113
Pembunuhan	84
Demonstrasi	83
Penganiayaan	66
Terorisme	19
Total	**12,837**

Tabel 6.10 Jumlah *tweets* per kategori aktor

Kategori aktor	**Jumlah *Tweets***
Infrastruktur_Pertamina	2,655
Aset_Pertamina	2,490
Truk_Tangki_Pertamina	382
Mobil_Tangki_Pertamina	263
Kapal_Tangki_Pertamina	193
Pekerja_Pertamina	5
Total	**5,988**

6.4 Hasil Pengujian Model

Tahap ini akan menghasilkan nilai dari pengujian *model* yang dilakukan terhadap ke 4 *model NER,* yaitu *model Actor, Location, Time,* dan Keterangan. Pengujian dilakukan dengan *command* yang sudah ditulis pada sub-bab 5.6.2.

Komposisi *data* yang akan digunakan sebagai model adalah gabungan dari seluruh hasil *data preprocessing* pada Tabel 6.4. Dikarenakan jumlah *tweet* per *keywords* tidak sama, maka akan dilakukan penyeimbangan komposisi agar tidak terjadi *data imbalancing.* Penyeimbangan dilakukan berdasarkan subjektifitas penulis hingga dirasa mendapatkan akurasi yang tinggi. Berikut pada Tabel 6.11 merupakan komposisi jumlah *rows* dari *file model* yang akan di evaluasi.

Tabel 6.11 Jumlah *rows model NER*

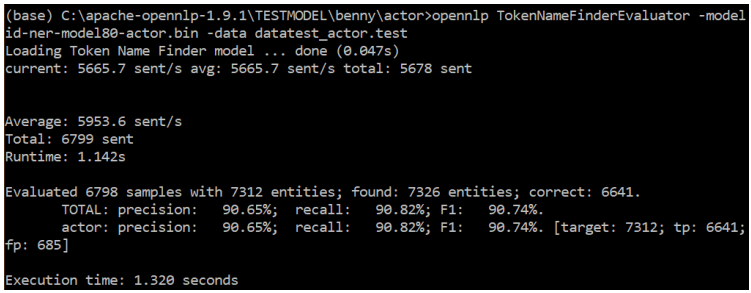
Model	**Jumlah *rows***	**Jumlah *rows* setelah penambahan**
Actor	34,289	34,289
Location	19,722	78,387
Keterangan	24,835	804,144
Time	75,226	75,226

Setelah itu akan dilakukan proses membagi *file model* menjadi 2 bagian yaitu 80% untuk *train* dan 20% untuk *test.* Berikut pada Tabel 6.12 merupakan detil jumlah *rows.*

Tabel 6.12 Jumlah *rows model* untuk pengujian

Model	**Jumlah *rows data train***	**Jumlah *rows data test***
Actor	27,491	6,798
Location	62,635	15,752
Keterangan	642,639	161,505
Time	60,115	15,451

Berikut pada Gambar 6.1 adalah contoh tangkapan gambar dari *command prompt* untuk *TokenNameFinderEvaluator*.



Gambar 6.1 Contoh tangkapan gambar *TokenNameFinderEvaluator*

Berikut pada Tabel 6.13 dapat dilihat rincian dari nilai pengujian tiap *model* yang sudah dilakukan proses *training* dan diuji menggunakan *data test* pada Tabel 6.12. *Samples* merupakan jumlah *data test* yang digunakan, *Entities* adalah jumlah entitas yang terkandung di dalam *data test.*

Model actor memiliki 6,798 *samples* dengan 7,326 *entities*, dan *entities* yang berhasil ditemukan oleh *model* sebanyak 6,641.

Model location memiliki 15,752 *samples* dengan 19,787 *entities*, dan *entities* yang berhasil ditemukan oleh *model* sebanyak 19,695.

Model keterangan memiliki 161,505 *samples* dengan 178,497 *entities*, dan *entities* yang berhasil ditemukan oleh *model* sebanyak 171,104.

Model time memiliki 15,451 *samples* dengan 15,452 *entities*, dan *entities* yang berhasil ditemukan oleh *model* sebanyak 15,451.

Tabel 6.13 Hasil pengujian *model NER*

actor	
Evaluated 6798 samples with 7312 entities; found: 7326 entities; correct: 6641.	
precision	90.65%
recall	90.82%
f1	90.74%
[target: 7312; tp: 6641; fp: 685]	
location	
Evaluated 15752 samples with 20022 entities; found: 19787 entities; correct: 19695.	
precision	99.54%
recall	98.37%
f1	98.95%
[target: 20022; tp: 19695; fp: 92]	
keterangan	
Evaluated 161505 samples with 171527 entities; found: 178497 entities; correct: 171104.	
precision	95.86%
recall	99.75%
f1	97.77%
[target: 171527; tp: 171104; fp: 7393]	
time	
Evaluated 15451 samples with 15451 entities; found: 15452 entities; correct: 15451.	
precision	99.99%
recall	100%
f1	100%
[target: 15451; tp: 15451; fp: 1]	

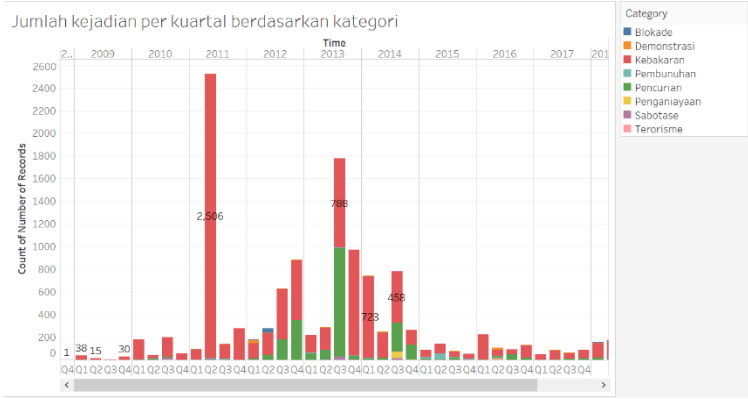
6.5 Hasil Visualisasi dengan Tableau

Hasil visualisasi memiliki rentang waktu dari tahun 2009-2019. Pada visualisasi di Gambar 6.2 merupakan visualiasi total kategori kejadian kejahatan/gangguan keamanan selama tahun 2009-2019. Visualisasi menunjukkan bahwa kategori kejadian kejahatan/gangguan keamanan yang mendominasi adalah Kategori Kebakaran dan Pencurian. Untuk kategori Kebakaran terjadi dengan jumlah kejadian sebesar 9,899. Lalu kategori kejadian terbesar kedua adalah Pencurian dengan jumlah kejadian sebesar 2,394. Untuk kategori kejadian yang lain seperti Blokade, Sabotase, Pembunuhan, Penganiayaan, Demonstrasi, dan Terorisme hanya mengisi sedikit dari proporsi kejadian kejahatan/gangguan keamanan yang terjadi selama rentang tahun 2009-2019.

Gambar 6.2 Visualisasi jumlah kategori kejahatan/gangguan keamanan

Pada Gambar 6.3 merupakan visualisasi jumlah kejadian kategori kejahatan/gangguan keamanan per kuartal dalam rentang tahun 2009-2019. Visualisasi menunjukkan bahwa pada kategori Kebakaran menyebar pada setiap tahunnya, dengan kejadian paling banyak terjadi pada tahun 2011 kuartal 2 dengan jumlah kejadian sebesar 2,506. Setelah itu untuk kategori

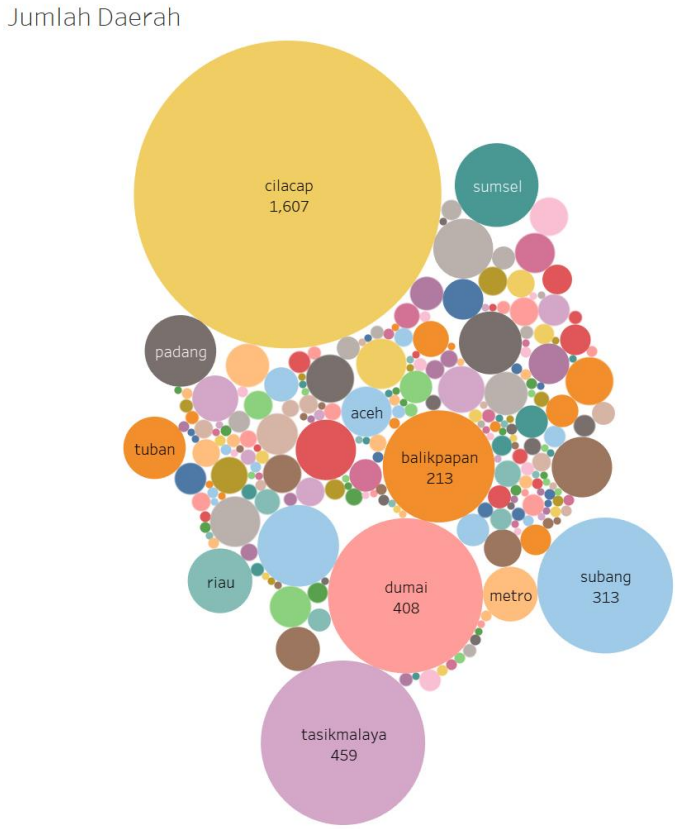
Pencurian paling banyak terjadi pada tahun 2013 kuartal 3 dengan jumlah kejadian sebesar 961. Untuk tahun dengan kejadian paling banyak adalah tahun 2013 sebesar 3,259 kejadian, lalu selanjutnya adalah tahun 2011 sebesar 3,039 kejadian. Untuk kategori kejahatan/gangguan keamanan lainnya tidak mendominasi proporsi data *tweets.*



Gambar 6.3 Visualisasi jumlah kategori kejahatan/gangguan keamanan per kuartal tahun

Pada Gambar 6.4 merupakan visualisasi jumlah daerah di Indonesia berdasarkan jumlah kejadian kategori kejahatan/gangguan keamanan menggunakan *Bubble Chart.* Visualisasi menunjukkan daerah Cilacap merupakan daerah yang tercatat paling banyak kejadian kejahatan/gangguan keamanan dengan jumlah kejadian sebesar 1,607 kejadian. Setelah itu terdapat daerah lainnya yang cukup besar jumlah kejadiannya seperti Tasikmalaya dengan jumlah kejadian sebesar 459, Dumai dengan jumlah kejadian sebesar 408,

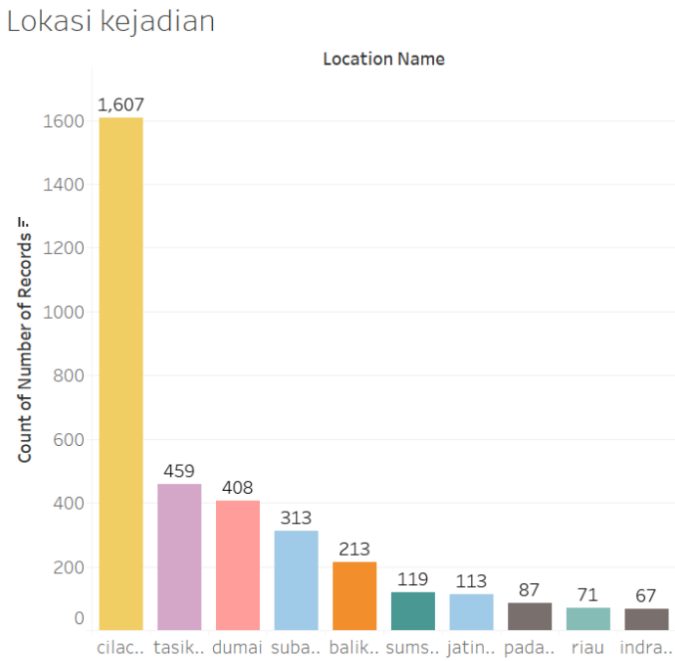
Subang dengan jumlah kejadian sebesar 313, dan Balikpapan sebesar 213 kejadian.



Gambar 6.4 Visualisasi jumlah kejadian kategori berdasarkan daerah

Pada Gambar 6.5 merupakan visualisasi *Bar Chart* yang menunjukkan 10 besar lokasi daerah kejadian berdasarkan kategori kejahatan/gangguan keamanan. Untuk proporsi daerah

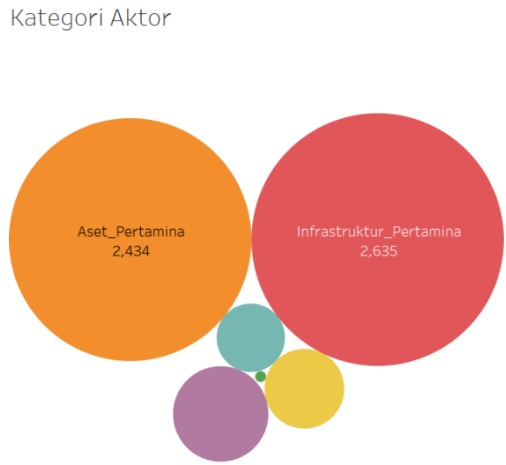
yang mendominasi sama seperti visualisasi *Bubble Chart* pada penjelasan Gambar 6.4.



Gambar 6.5 Visualisasi 10 besar daerah berdasarkan jumlah kejadian

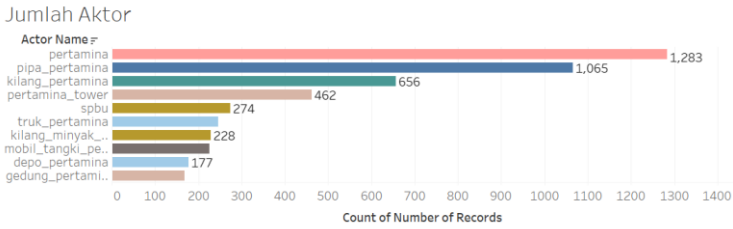
Gambar 6.6 merupakan *Bubble Chart* yang menunjukkan jumlah kategori aktor berdasarkan jumlah kejadian. Terdapat 2 kategori aktor yang mendominasi yaitu Aset_Pertamina dan Infrastruktur_Pertamina dengan jumlah masing-masing sebesar 2,434 dan 2,635 kejadian. Untuk penjelasan mengenai aktor apa saja yang terdapat pada kategori aktor, dapat dilihat pada

Tabel 5.4 di bagian *instances* per *subclass Actor*. Visualisasi ini digunakan sebagai filter *slice* nanti di dashboard.



Gambar 6.6 Visualisasi kategori aktor berdasarkan jumlah kejadian

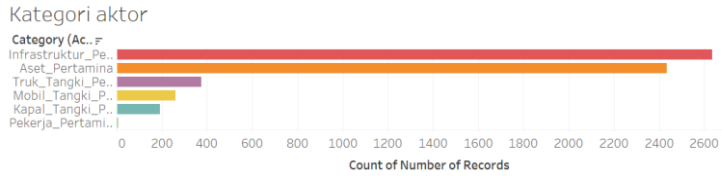
Gambar 6.7 merupakan visualisasi berbentuk *Bar Chart* yang menunjukkan jumlah 10 besar aktor berdasarkan jumlah kejadian. Proporsi jumlah aktor dapat dilihat pada visualisasi, nanti visualisasi akan digunakan bersama visualisasi jumlah kategori aktor pada dashboard.



Gambar 6.7 Visualisasi 10 besar jumlah actor berdasarkan jumlah kejadian

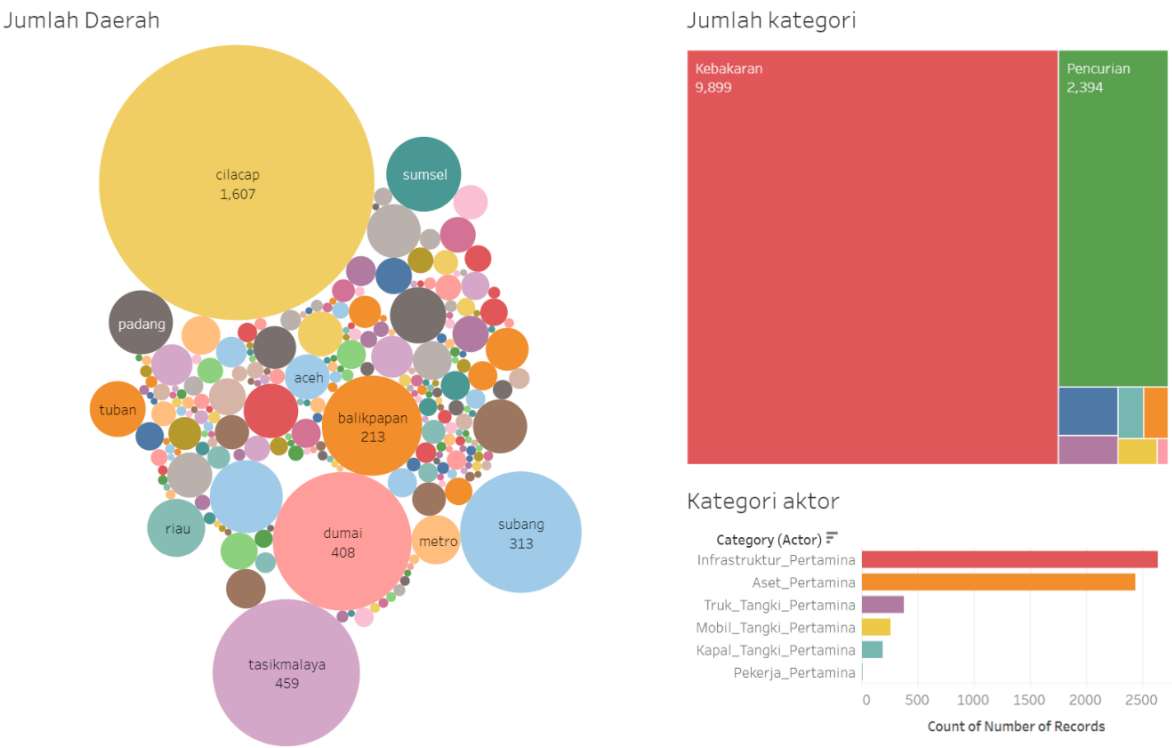
Gambar 6.8 merupakan visualisasi *Bar Chart* yang menunjukkan hal yang sama dengan visualisasi *Bubble Chart*

kategori aktor pada penjelasan Gambar 6.6. Visualisasi ini akan digunakan sebagai *slice* pada dashboard.

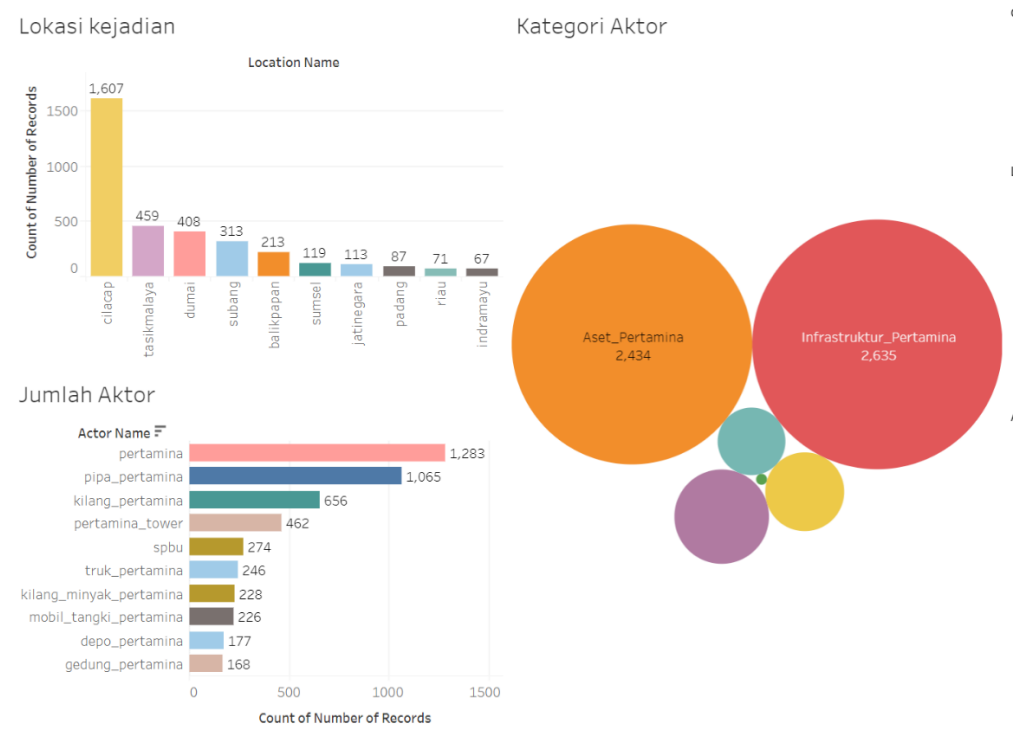


Gambar 6.8 Visualisasi *Bar Chart* kategori aktor berdasarkan jumlah kejadian

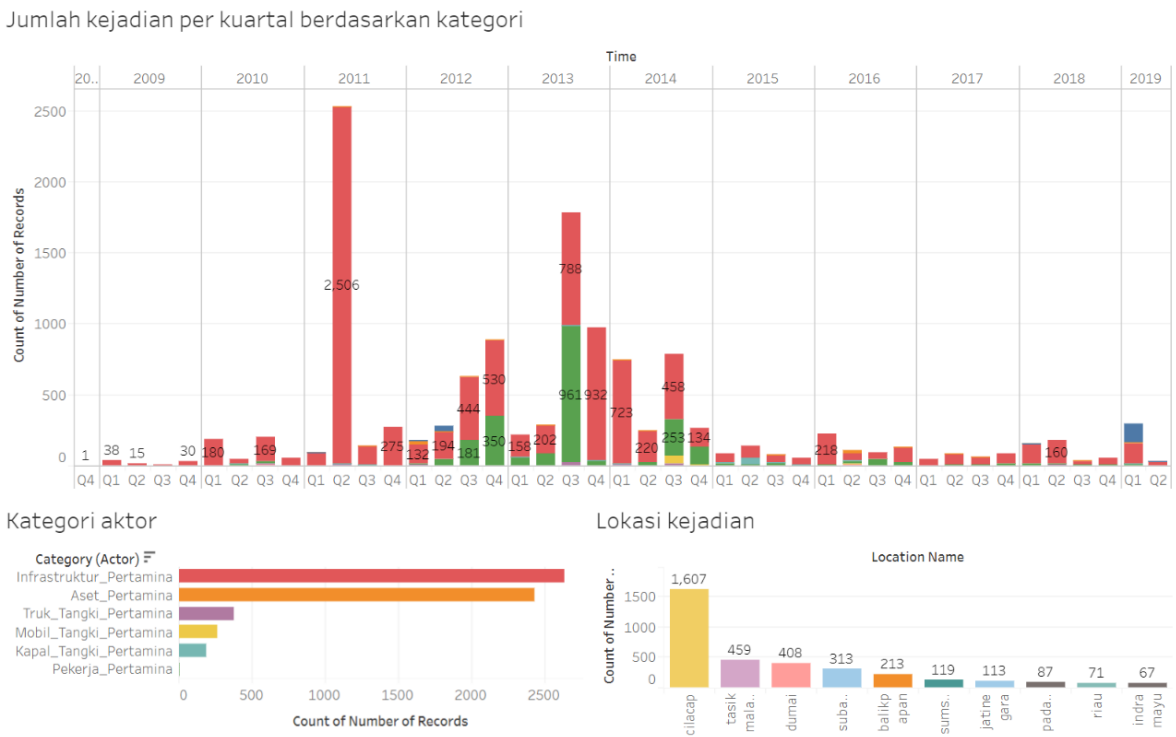
Penulis telah membuat 3 *dashboard* yang menggunakan seluruh visualisasi yang sudah dibuat pada Gambar 6.2-Gambar 6.8. Untuk *dashboard 1* dapat dilihat padaGambar 6.9, diharapkan dengan adanya *dashboard 1* mampu mendapatkan informasi jumlah kategori kejadian per lokasi kejadian. Untuk *dashboard 2* dapat dilihat pada Gambar 6.10, diharapkan dengan adanya *dashboard 2* mampu memberikan informasi dimana saja lokasi kejadian pada suatu kategori aktor, dan memberi informasi jumlah aktor apa yang mendominasi suatu kategori aktor. Untuk *dashboard 3* dapat dilihat pada Gambar 6.11, diharapkan dengan adanya *dashboard 3* dapat memberikan informasi jumlah kategori aktor apa yang terjadi per kuartal tahun dan lokasi kejadian apa yang terdapat pada kuartal tahun tersebut.



Gambar 6.9 *Dashboard 1*, Perbandingan daerah dengan kategori kejahatan/gangguan keamanan



Gambar 6.10 *Dashboard 2*, Perbandingan kategori aktor dengan lokasi kejadian dan jumlah aktor



Gambar 6.11 *Dashboard 3*, Perbandingan jumlah kejadian per kuartal berdasarkan kategori aktor dan lokasi kejadian

BAB VII
KESIMPULAN DAN SARAN

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

7.1 Kesimpulan

Dari proses penelitian pada pengerjaan tugas akhir yang sudah dilakukan oleh penulis, dapat disimpulkan beberapa hal sebagai berikut, yaitu:

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 90,65%, *recall* sebesar 90,82%, dan *F1 score* sebesar 90,74%. 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 95,86%, *recall* sebesar 99,75%, dan *F1 score* sebesar 97,77%. Untuk model *Time* memiliki akurasi untuk *precision* sebesar 99,99%, *recall* sebesar 100%, dan *F1 score* sebesar 100%.
2. Untuk model *Time* dapat memiliki akurasi yang tinggi dikarenakan pola waktu yang terdapat pada setiap *tweets* sama yaitu pada akhir *tweets.* Hal ini dilakukan pada tahap *Date Merging* di sub-bab *4.3.4.*
3. Untuk model *Location,* pemberian *tag* secara otomatis dengan penggunaan *data* daerah di Indonesia terbukti mampu memberikan *model* dengan akurasi yang cukup tinggi.
4. Untuk model *Actor* dan Keterangan dilakukan secara *tag manual*, sehingga hasil model yang didapatkan tidak setinggi *model Location* dan *Time.*
5. Penggunaan ontologi yaitu *knowledge-based framework* sebagai *classifier* pada proses ekstraksi

informasi dengan *NER* dan *library Jena* dapat menentukan kategori aktor maupun kejadian berdasarkan *class* dan *subclass* yang ada pada ontologi.
6. Selain itu dengan menggunakan ontologi, memungkinkan untuk penggunaan selanjutnya sebagai *model* awal dan dapat dikembangkan pula pada penelitian selanjutnya
7. Berdasarkan hasil ekstraksi informasi dan visualisasi, menunjukkan bahwa kategori kejadian yang paling besar adalah Kebakaran dengan kategori aktor yang berpengaruh adalah Infrastruktur_Pertamina dan aktornya adalah Pertamina, serta lokasi kejadian paling besar berada di Cilacap.

7.2 Saran

Dalam pengerjaan penelitian tugas akhir, terdapat beberapa saran yang diharapkan dapat bermanfaat bagi pengembangan penelitian ke depan, yaitu:

1. Dalam proses *training* dan *testing model NER* masih menggunakan *command line*, sebaiknya menggunakan *OpenNLP Evaluation API* sehingga memiliki dokumentasi kode yang lebih baik
2. Proses *data preprocessing* masih memberi hasil *data* yang tidak sesuai harapan, seperti *tweet* yang tidak memiliki informasi yang berkaitan dengan kejadian kejahatan/gangguan keamanan. Diharapkan pada penelitian selanjutnya dapat diterapkan dan memberikan hasil *data* yang minim *noise*.
3. Pada penelitian ini setiap proses masih dilakukan secara *individual*, diharapkan pada penelitian selanjutnya dapat diintegrasikan dan dilakukan secara *real-time* menjadi suatu aplikasi yang dapat menjadi sumber informasi.

DAFTAR PUSTAKA

[1] W. Respati, "Transformasi Media Massa Menuju Era Masyarakat Informasi Di Indonesia," *Humaniora*, vol. VOL.5, no. 9, pp. 39–51, 2014.

[2] A. P. J. Internet, "Penetrasi & Perilaku Pengguna Internet Indonesia," *Apjii*, vol. 2018, no. 31 August 2018, p. Hasil Survey, 2017.

[3] "We Are Social - Digital Report 2018." [Online]. Available: https://digitalreport.wearesocial.com/. [Accessed: 11-Feb-2019].

[4] "Social Media Stats Indonesia | StatCounter Global Stats." [Online]. Available: http://gs.statcounter.com/social-media-stats/all/indonesia. [Accessed: 11-Feb-2019].

[5] "Statistik Kriminal 2018," *Badan Pus. Stat.*, p. 2016, 2018.

[6] D. Murthy, "Twitter: Social communication in the Twitter age," *Int. J. Commun.*, vol. 7, no. December, pp. 1240–1242, 2013.

[7] M. Zappavigna, "8. Twitter," *Pragmat. Soc. Media*, no. December, 2017.

[8] A. Labrinidis and H. V Jagadish, "Challenges and Opportunities with Big Data," pp. 2032–2033.

[9] K. Coyle, T. Open, and C. Alliance, "MANAGING TECHNOLOGY ! Mass Digitization of Books," vol. 32, no. 6, pp. 641–645, 2006.

[10] D. Nadeau and S. Sekine, "A survey of named entity recognition and classification."

[11] A. Maedche and S. Staab, "Ontology learning for the Semantic Web," *IEEE Intell. Syst.*, vol. 16, no. 2, pp. 72–79, Mar. 2001.

[12] S. R. Kruk, M. Synak, and M. Dabrowski, "Semantic

Web and Ontologies," *Semant. Digit. Libr.*, pp. 41–54, 2009.

[13] G. Pant, P. Srinivasan, and F. Menczer, "Crawling the Web," in *Web Dynamics*, 2004, pp. 153–177.

[14] T. BERNERS-LEE, JAMES HENDLER, and O. LASSILA, "The Semantic Web," vol. 309, no. 1, pp. 62–67.

[15] W3C, "Semantic Web Standards," pp. 2–3, 2013.

[16] P. Patel-Schneider and J. Siméon, "The Yin/Yang web: XML syntax and RDF semantics," *Proc. 11th Int. Conf. World Wide Web, WWW '02*, no. November, pp. 443–453, 2002.

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

[18] "Apache OpenNLP Developer Documentation." [Online]. Available: https://opennlp.apache.org/docs/1.5.3/manual/opennlp.html. [Accessed: 28-Mar-2019].

BIODATA PENULIS

Penulis lahir di Mataram pada tanggal 28 September 1997. Penulis merupakan anak kedua dari 2 bersaudara. Pendidikan formal yang ditempuh oleh penulis yaitu SD Kristen Aletheia Ampenan, SMP Kristen Aletheia Ampenan, dan SMA Negeri 1 Mataram.

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 05211540000055. Selama masa perkuliahan, penulis aktif mengikuti beberapa kegiatan kemahasiswaan, antara lain sebagai Pengurus Inti Humas Persekutuan Mahasiswa Kristen ITS (PMK ITS). Selain itu, dalam hal akademik penulis juga menjadi asisten praktikum pada mata kuliah Pengantar Sistem Operasi. Penulis juga menjadi juara 3 lomba Binus International ISCC 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: bennybtc03@gmail.com.