Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR - KS 141501

ANALISIS SENTIMEN MEDIA SOSIAL UNTUK TEKS BERBAHASA INDONESIA MENGGUNAKAN ALGORITMA CNN (*CONVOLUTIONAL NEURAL NETWORK*) (STUDI KASUS: POLITIK)

SOCIAL MEDIA SENTIMENT ANALYSIS IN INDONESIAN LANGUAGE USING CNN (CONVOLUTIONAL NEURAL NETWORK) ALGORITHM (CASE STUDY: POLITICS)

PRASETYO WAHYU ADI WIANTO
NRP 05211440000030

Dosen Pembimbing :
Renny Pradina K., S.T, M.T., SCJP

DEPARTEMEN SISTEM INFORMASI
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR – KS 141501

ANALISIS SENTIMEN MEDIA SOSIAL UNTUK TEKS BERBAHASA INDONESIA MENGGUNAKAN ALGORITMA CNN (*CONVOLUTIONAL NEURAL NETWORK*) (STUDI KASUS: POLITIK)

PRASETYO WAHYU ADI WIANTO
NRP 05211440000030

Dosen Pembimbing :
Renny Pradina K., S.T, M.T., SCJP

DEPARTEMEN SISTEM INFORMASI
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

TUGAS AKHIR – KS 141501

SOCIAL MEDIA SENTIMENT ANALYSIS IN INDONESIAN LANGUAGE USING CNN (CONVOLUTIONAL NEURAL NETWORK) ALGORITHM (CASE STUDY: POLITICS)

PRASETYO WAHYU ADI WIANTO
NRP 05211440000030

SUPERVISOR :
Renny Pradina K., S.T, M.T., SCJP

DEPARTMENT OF INFORMATION SYSTEMS
Faculty of Information Technology and Communication
Institut Teknologi Sepuluh Nopember
Surabaya 2018

LEMBAR PENGESAHAN

ANALISIS SENTIMEN MEDIA SOSIAL UNTUK TEKS BERBAHASA INDONESIA MENGGUNAKAN ALGORITMA CNN (CONVOLUTIONAL NEURAL NETWORKS) (STUDI KASUS: POLITIK)

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:

PRASETYO WAHYU ADI WIANTO
NRP. 05211440000030

Surabaya, Juli 2018

KEPALA
DEPARTEMEN SISTEM INFORMASI

Dr. Ir. Aris Tjahyanto. M.Kom
NIP. 19650310 199102 1 001

LEMBAR PERSETUJUAN

ANALISIS SENTIMEN MEDIA SOSIAL UNTUK TEKS BERBAHASA INDONESIA MENGGUNAKAN ALGORITMA CNN (CONVOLUTIONAL NEURAL NETWORKS) (STUDI KASUS: POLITIK)

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 :

PRASETYO WAHYU ADI WIANTO
NRP. 05211440000030

Disetujui Tim Penguji :Tanggal Ujian : 10 Juli 2018
Periode Wisuda : September 2018

Renny Pradina K., S.T, M.T., SCJP **(Pembimbing I)**

Nur Aini R., S.Kom, M.Sc.Eng, Ph.D **(Penguji I)**

Radityo Prasetianto W., S.Kom, M.Kom **(Penguji II)**

ANALISIS SENTIMEN MEDIA SOSIAL UNTUK TEKS BERBAHASA INDONESIA MENGGUNAKAN ALGORITMA CNN (CONVOLUTIONAL NEURAL NETWORK) (STUDI KASUS: POLITIK)

Nama Mahasiswa : Prasetyo Wahyu Adi Wianto
NRP : 05211440000030
Jurusan : Sistem Informasi FTIK-ITS
Pembimbing 1 : Renny Pradina K., S.T, M.T., SCJP

ABSTRAK

Hampir setengah pengguna internet di Indonesia adalah penggiat media sosial, di mana 85% di antaranya mengakses media sosial melalui perangkat seluler. Media sosial akan memberikan pengaruh besar terhadap masyarakat dalam berbagai aspek khususnya aspek politik. Semakin berkembangnya media sosial untuk kegiatan maupun sarana politik, membuka peluang pada akses informasi yang didapatkan terutama opini publik terhadap objek politik. Salah satu metode yang dapat digunakan untuk mengetahui tingkat penilaian publik terhadap objek politik adalah analisis sentimen. Klasifikasi sentimen media sosial dengan tingkat keberagaman bahasa yang cukup variatif dilakukan dengan menggunakan metode Convolutional Neural Network (CNN). Penelitian sebelumnya melakukan percobaan pada tugas klasifikasi teks menggunakan CNN (Convolutional Neural Network) dan mencapai hasil yang sangat baik. Teks yang digunakan sebagai input pelatihan model CNN diubah menjadi representasi vektor kata menggunakan word2vec dan fasttext sebagai pembanding. Tugas klasifikasi sentimen terdiri dari 3 subtask dengan perbedaan jumlah kelas yang digunakan. Berdasarkan hasil eksperimen pelatihan yang dilakukan, akurasi paling baik yang didapat pada subtask A 0.833, subtask B 0.709, dan subtask C 0.658. Kemudian performa klasifikasi teks oleh

model CNN lebih unggul dibandingkan dengan SVM, dan naive bayes pada setiap subtask.

Kata kunci: Analisis Sentimen, Politik, Media Sosial, Deep Learning, Convolutional Neural Network, Word Embedding, Word2Vec

SOCIAL MEDIA SENTIMENT ANALYSIS IN INDONESIAN LANGUAGE USING CNN (CONVOLUTIONAL NEURAL NETWORK) ALGORITHM (CASE STUDY : POLITICS)

Student Name	**: Prasetyo Wahyu Adi**
NRP	**: 05211440000030**
Department	**: Sistem Informasi FTIK-ITS**
Supervisor 1	**: Renny Pradina K., S.T, M.T., SCJP**

ABSTRACT

Nearly half of internet users in Indonesia are social media activists, with 85% of them accessing social media via mobile devices. Social media will give a big influence on society in various aspect, especially the political aspect. Increasingly, the development of social media for political activities and facilities opens opportunities for access to information obtained, especially public opinion on political objects. One method that can be used to determine the level of public assessment of political objects is the analysis of sentiment. The classification of social media sentiment with varying levels of language diversity is done using the Convolutional Neural Network (CNN) method. Previous research experimented on text classification tasks using CNN (Convolutional Neural Network) and achieved excellent results. The text used as a CNN model training input is converted to a word vector representation using word2vec and fasttext as a comparison. The sentence classification task consists of 3 subtasks with a different number of classes used. Based on the results of the training experiments conducted, the best accuracy obtained in subtask A 0.833, subtask B 0.709, and subtask C 0.658. Then the text classification performance by the CNN model is superior to the SVM, and the Naive Bayes on each subtask.

Keywords: Sentiment Analysis, Politic, Social Media, Deep Learning, Convolutional Neural Network, Word Embedding, Word2Vec

KATA PENGANTAR

Puji dan syukur penulis tuturkan ke hadirat Allah SWT, Tuhan Semesta Alam yang telah memberikan karunia dan hidayah-Nya kepada penulis sehingga penulis mendapatkan kelancaran dalam menyelesaikan tugas akhir dengan judul:

ANALISIS SENTIMEN MEDIA SOSIAL UNTUK TEKS BERBAHASA INDONESIA MENGGUNAKAN ALGORITMA CNN (*CONVOLUTIONAL NEURAL NETWORK*) (STUDI KASUS: POLITIK)

yang merupakan salah satu syarat kelulusan pada Departemen Sistem Informasi, Fakultas Teknologi Informasi dan Komunikasi, Institut Teknologi Sepuluh Nopember Surabaya.

Terima kasih penulis sampaikan kepada pihak-pihak yang telah mendukung, memberikan saran, motivasi, semangat, dan bantuan baik berupa materiil maupun moril demi tercapainya tujuan pembuatan tugas akhir ini. Tugas akhir ini tidak akan pernah terwujud tanpa bantuan dan dukungan dari berbagai pihak yang sudah melauangkan waktu, tenaga dan pikirannya. Secara khusus penulis akan menyampaikan ucapan terima kasih yang sebanyak-banyaknya kepada:

1) Orang tua penulis, serta keluarga yang telah memberikan motivasi, kasih sayang, serta doa sehingga penulis dapat menyelesaikan pendidikan S1 di Departemen Sistem Informasi ITS dengan baik.
2) Ibu Renny Pradina, S.T, M.T. selaku dosen pembimbing yang telah dengan sabar dan penuh perhatian memberikan ilmu, petunjuk, dan motivasi yang sangat besar sehingga penulis dapat menyelesaiakan tugas akhir ini.
3) Ibu Nur Aini, S.Kom, M.Sc.Eng, Ph.D. dan bapak Radityo Prasetianto. W, S.Kom, M.Kom. selaku dosen penguji yang telah memberikan kritik dan masukan yang membangun sehingga dapat menyempurnakan tugas akhir ini.

4) Seluruh dosen pengajar beserta staf dan karyawan Departemen Sistem Informasi, Fakultas Teknologi Informasi dan Komunikasi ITS Surabaya yang telah memberikan ilmu dan pengalaman berharga kepada penulis.
5) Rekan-rekan mahasiswa Departemen Sistem Informasi OSIRIS serta anggota lab Akuisisi Data dan Diseminasi Informasi (ADDI) atas bantuan dan pertemanannya kepada penulis hingga saat ini.
6) Teman-teman seperjuangan anak bimbing Ibu Renny, Alden, Adrian, Putra, Ocha, Dewangga, Arif, dan Hans yang selalu menemani penulis dan memberi motivasi dalam pengerjaan tugas akhir.
7) Teman-teman yang menemani kehidupan kampus penulis Fauzan, Fanny, Alim, Risha, Ayusha yang selalu mengingatkan penulis dalam hal-hal kebaikan.
8) Teman-teman bermain penulis Calvin, Endar, Rysma, teman futsal, dan teman-teman lainnya yang tidak dapat disebutkan penulis satu-satu yang sudah memberikan penulis pengalaman berharga.
9) Teman-teman sosial masyarakat HMSI yang sudah membangun karakter penulis untuk lebih peduli dengan masyarakat.
10) Teman-teman BSO IECC BEM ITS yang sudah menjadi keluarga penulis dan memberikan banyak pembelajaran dimasa perkuliahan.
11) Serta semua pihak yang telah membantu dalam pengerjaan tugas akhir ini yang belum mampu penulis sebutkan diatas.

Terima kasih atas segala bantuan, dukungan, serta doa yang telah diberikan. Penulis menyadari bahwa tugas akhir ini masih belum sempurna dan memiliki banyak kekurangan di dalamnya. Oleh karena itu, penulis juga memohon maaf atas segala kesalahan penulis buat dalam buku tugas akhir ini. Penulis membuka pintu selebar-lebarnya bagi pihak yang ingin memberikan kritik maupun saran, serta penelitian selanjutnya yang ingin menyempurnakan

karya dari tugas akhir ini. Semoga buku tugas akhir ini bermanfaat bagi seluruh pembaca.

Surabaya, 15 Juli 2018

Penulis

Halaman ini sengaja dikosongkan

DAFTAR ISI

Halaman ini sengaja dikosongkan

DAFTAR GAMBAR

DAFTAR DIAGRAM

Halaman ini sengaja dikosongkan

DAFTAR TABEL

DAFTAR KODE

BAB I
PENDAHULUAN

Pada bagian ini akan dijelaskan mengenai latar belakang masalah, perumusan masalah, batasan masalah, tujuan penelitian, manfaat, dan relevansi yang mendasari penelitian tugas akhir.

1.1. Latar Belakang

Indonesia merupakan negara dengan populasi terbesar ke 4 di dunia, dengan jumlah penduduk lebih dari 262 juta jiwa[1]. Laju pertumbuhan penduduk yang selalu meningkat juga berjalan secara linear dengan laju pertumbuhan pengguna internet di Indonesia. Menurut hasil survei yang dilakukan oleh APJII (Asosiasi Penyelenggara Jasa Internet Indonesia) pada tahun 2017, pengguna internet di Indonesia adalah sebesar 143,26 juta jiwa, jumlah tersebut merupakan 54,68% dari total populasi penduduk Indonesia[2] dan diperkirakan merupakan jumlah pengguna internet terbesar nomor 5 di dunia[3]. Fenomena tersebut tak lepas dari semakin berkembangnya teknologi ponsel dan koneksi *broadband mobile* yang terjangkau, sehingga mendorong masyarakat semakin mudah untuk mendapatkan akses internet. Pemanfaatan internet oleh masyarakat Indonesia di dominasi oleh layanan *chatting*, media sosial, dan *search engine*[2]. Menurut laporan Tetra Pak Indeks 2017, hampir setengah pengguna internet di Indonesia adalah penggiat media sosial, atau berkisar di angka 40%, di mana 85% di antaranya mengakses media sosial melalui perangkat seluler [4].

Dengan jumlah pengguna media sosial di Indonesia yang cukup besar, maka media sosial akan memberikan pengaruh besar terhadap masyarakat dalam berbagai aspek. Salah satu aspek yang tak lepas dari pengaruh media sosial adalah politik. Bahkan menurut studi yang dilakukan oleh Chavez pada tahun 2012, menunjukkan bahwa media sosial adalah alat kampanye politik yang sangat efektif[5]. Tidak sedikit dari beberapa tokoh politik di Indonesia memanfaatkan

media sosial untuk melakukan interaksi dengan publik sekaligus untuk memasarkan gagasan yang dimiliki. Media sosial dapat menjadi cara yang potensial dalam mendobrak politik demokrasi massa yang menyuarakan suara dari bawah ke atas[6]. Media sosial juga mampu meningkatkan tingkat elektabilitas seorang calon pemimpin/pejabat publik dan golongan tertentu (partai) secara masif, hal ini terlihat dari beberapa konstelasi politik, salah satunya pemilihan kepala daerah pada tahun 2017 yang mayoritas menggunakan media sosial sebagai sarana komunikasi politik. Semakin berkembangnya media sosial untuk kegiatan maupun sarana politik, juga membuka peluang terhadap akses informasi yang didapatkan terutama opini publik terhadap objek politik.

Opini publik pada media sosial terhadap objek politik tertentu dapat digunakan untuk mengetahui bagaimana penilaian publik secara keseluruhan terhadap objek tersebut. Salah satu metode yang dapat digunakan untuk mengetahui tingkat penilaian publik terhadap objek politik adalah analisis sentimen[7]. Namun keberagaman gaya bahasa yang digunakan pada media sosial di Indonesia, menjadi tantangan tersendiri terhadap metode analisis sentimen saat ini, khususnya ketika melakukan klasifikasi teks pada opini publik di media sosial.

Sehingga dari permasalahan tersebut penelitian ini akan menawarkan solusi dalam melakukan analisis sentimen media sosial pada opini publik terhadap objek yang berkaitan dengan topik politik. Penelitian ini akan menggunakan metode *Deep Learning*[8] untuk klasifikasi teks yang merupakan pengembangan dari *Machine Learning*, di mana performa dalam proses pembelajarannya lebih baik. Untuk mendapatkan tingkat akurasi yang tinggi pada model klasifikasi teks, algoritma yang digunakan adalah *Convolutional Neural Network*[7], dengan melibatkan model *word2vec*[9] pada proses *word embeddings*. Diharapkan dari analisis sentimen dengan metode *Deep Learning for* NLP menggunakan algoritma CNN dan *Word2Vec*, dapat membantu menilai opini publik terhadap objek politik secara lebih tepat.

1.2. Rumusan Masalah

Berdasarkan uraian pada latar belakang, maka rumusan masalah dari penelitian tugas akhir ini adalah sebagai berikut:

1. Bagaimana metode untuk mendapatkan kumpulan data teks dari media sosial sesuai dengan topik yang telah ditentukan ?
2. Bagaimana metode untuk melakukan pra-pemrosesan terhadap kumpulan data teks yang diperoleh ?
3. Bagaimana metode untuk membuat model *embeddings* menggunakan algoritma *word2vec* pada kumpulan data teks ?
4. Bagaimana metode untuk membuat model *Convolutional Neural Network* dengan tugas klasifikasi teks ?
5. Bagaimana cara untuk mengukur dan mengevaluasi performa model *word2vec* dan *Convolutional Neural Network* ?

1.3. Batasan Permasalahan

Dari permasalahan yang disebutkan di atas, maka batasan masalah pada penelitian tugas akhir ini adalah sebagai berikut:

1. Penelitian ini menggunakan kumpulan data yang berasal dari media sosial *twitter* dengan topik politik.
2. Kumpulan data teks yang digunakan mengandung bahasa Indonesia.
3. Proses penghapusan data *twitter* yang terduplikasi hanya dapat mendeteksi data dengan karakter teks yang sama persis.

1.4. Tujuan Penelitian

Tujuan besar dari pengerjaan penelitian tugas akhir ini adalah sebagai berikut:

1. Penerapan metode *crawling* data menggunakan *library* pada Python dan API pada media sosial *twitter*.
2. Membuat metode pra-pemrosesan pada kumpulan data *twitter* yang diperoleh.
3. Membuat model *word2vec* dari kumpulan data teks yang diperoleh.
4. Membuat model *Convolutional Neural Network* untuk klasifikasi teks.
5. Melakukan analisis performa model *word2vec* dan *Convolutional Neural Network* yang dibangun dengan membuat skenario perubahan parameter pada saat *training* model.

1.5. Manfaat Penelitian

Berikut adalah manfaat yang diharapkan dari pengerjaan penelitian tugas akhir ini:

1.5.1. Bagi Penulis

Manfaat yang diperoleh penulis adalah sebagai berikut:

1. Mengetahui dan memahami metode *deep learning* pada NLP (*Natural Language Processing*).
2. Mengetahui dan memahami metode *word embedding* menggunakan algoritma *word2vec*.
3. Mengetahui dan memahami arsitektur *Convolutional Neural Network* dan menerapkannya pada tugas klasifikasi teks.

1.5.2. Bagi Masyarakat

Manfaat yang diperoleh masyarakat adalah sebagai salah satu bentuk penelitian dengan menggunakan metode *deep learning* pada NLP khususnya bahasa Indonesia, sehingga memungkinkan untuk terdapat penelitian selanjutnya yang dapat dikembangkan ke dalam beberapa permasalahan yang muncul pada NLP bahasa Indonesia.

1.6. Relevansi

Topik pada penelitian tugas akhir ini adalah analisis sentimen, sehingga relevan dengan bidang keilmuan pada laboratorium Akuisisi Data dan Diseminasi Informasi. Mata kuliah yang berkaitan dengan penelitian tugas akhir ini di antaranya adalah Sistem Cerdas, Penggalian Data dan Analitik Bisnis, Sistem Pendukung Keputusan, dan Kecerdasan Bisnis.

Halaman ini sengaja dikosongkan

BAB II
TINJAUAN PUSTAKA

Bab ini berisi tinjauan pustaka yang digunakan dalam penelitian tugas akhir yang mencakup penelitian-penelitian sebelumnya, dasar teori, dan metode yang akan digunakan selama pengerjaan penelitian tugas akhir.

2.1. Penelitian Sebelumnya

Terdapat beberapa penelitian sebelumnya yang memiliki topik serupa dengan penelitian tugas akhir ini, penelitian tersebut dapat dilihat pada Tabel 2.1 dan Tabel 2.2.

Tabel 2.1 *Convolutional Neural Networks for Sentence Classification*

Judul 1	*Convolutional Neural Networks for Sentence Classification*
Nama, Tahun	Yoon Kim, 2014
Gambaran Umum Penelitian	Penelitian ini membahas tentang kinerja dari algoritma CNN dalam melakukan tugas analisis sentimen pada berbagai macam jenis data. Dan didapatkan hasil yang sangat baik. Hasil dari penelitian ini membuktikan bahwa *pre-training* dari vektor kata merupakan unsur yang penting dalam penggunaan *deep learning* untuk NLP (*Natural Language Processing*)
Keterkaitan Penelitian	Memiliki keterkaitan dalam implementasi CNN pada teks

Tabel 2.2 A *Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification*

Judul 2	*A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification*
Nama, Tahun	Zhang, Y., & Wallace, B., 2015
Gambaran Umum Penelitian	Penelitian ini membahas tentang pengaruh dari *hyper parameter* pada proses pembuatan model CNN. Parameter tersebut adalah *input* vektor, ukuran *feature* region, ukuran *feature* maps, dan strategi pada *pooling layer*.
Keterkaitan Penelitian	Memiliki keterkaitan dalam penggunaan CNN pada klasifikasi teks

2.2. Landasan Teori

Landasan teori berisi teori-teori yang digunakan dalam pengerjaan penelitian tugas akhir. Dalam landasan teori, acuan yang digunakan adalah berdasarkan penelitian dan buku.

2.2.1. Media Sosial

Sosial media merupakan layanan berbasis internet. Berbeda dengan prinsip media tradisional di mana informasi hanya berjalan 1 arah dari pembuat informasi ke pembaca saja, sebaliknya prinsip media sosial memungkinkan sebuah informasi dapat berjalan lebih dari 1 arah, di mana pengguna dapat menjadi pembaca dan penulis informasi sekaligus[10]. Banyak publikasi ilmiah yang ditulis dengan topik sosial media, yang memiliki tujuan untuk mendefinisikan secara teori tentang arti sosial media yang sebenarnya. Menurut Andreas Kaplan pada penelitian yang dilakukannya, menyatakan bahwa sosial media adalah sekelompok aplikasi berbasis internet yang membangun fondasi ideologi dan teknologi web 2.0, kemudian

sosial media memungkinkan terjadinya penciptaan dan pertukaran konten yang dihasilkan oleh pengguna[11]. Kaplan juga mengemukakan tentang konsep klasifikasi tipe media sosial berdasarkan dimensi sosial. Dimensi yang pertama adalah media sosial berdasarkan *social presence* atau *media richness*, kemudian dimensi yang kedua adalah media sosial berdasarkan *self-presentation* atau *self-disclosure* [11]. Untuk visualisasi klasifikasi media sosial berdasarkan gabungan dimensi tersebut dapat dilihat pada Tabel 2.3.

Tabel 2.3 Klasifikasi Media Sosial Berdasarkan *Social Presence/Media Richness* dan *Self-Presentation/Self-Disclosures*

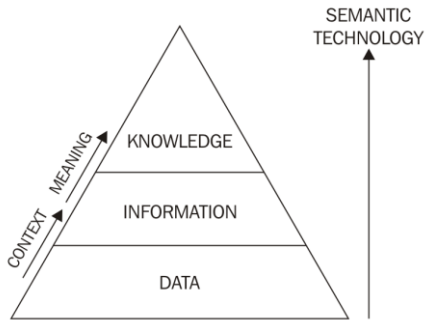
		Social Presence / Media Richness		
		Low	***Medium***	***High***
Self Presentation / Self Disclosure	***High***	*Blogs*	*Social Networking Sites (e.g., Facebook)*	*Virtual Social Worlds (e.g., Second Life)*
	Low	*Collaborative Projects (e.g., Wikipedia)*	*Content Communities (e.g., Youtube)*	*Virtual Game (e.g., World of Warcraft)*

Dengan beberapa fitur utama yang terdapat pada beberapa platform, media sosial digunakan untuk berbagai tujuan, diantara-Nya adalah sebagai berikut:

a. Tetap terhubung dengan teman dan keluarga. Contohnya : via *Facebook*.
b. *Microblogging* dan mendapatkan berita terbaru. Contohnya : via *Twitter*.
c. Terhubung dengan jaringan profesional. Contohnya : via *LinkedIn*.
d. Berbagi konten multimedia. Contohnya : via *Youtube*, *Instagram*, dan *Vimeo*.
e. Menemukan jawaban dari sebuah pertanyaan. Contohnya : *Stack Overflow*, dan *Quora*.

f. Menemukan hal-hal menarik. Contohnya : *Pinterest*, dan *Behance*.

Data yang berasal dari media sosial dapat diekstraksi dan menjadi sebuah *knowledge*. konsep dari *knowledge* sendiri dapat divisualisasikan berupa bagian dari sebuah hierarki piramida, di mana data sebagai bagian dasarnya, kemudian informasi sebagai bagian tengah, dan *knowledge* berada diposisi atas. Dapat dilihat pada Gambar 2.1**Error! Reference source not found.**.



Gambar 2.1 Visualisasi Hierarki Piramida Data, Informasi, dan *Knowledge*

Knowledge yang didapat dari ekstraksi data media sosial dapat membantu untuk proses pengambilan keputusan dalam sebuah strategi bisnis[10].

2.2.2. *Data Mining*

Data mining juga biasa disebut sebagai KDD (*Knowledge Discovery from Data*). Banyak sekali istilah yang digunakan untuk mendefinisikan apa itu *data mining*, contoh istilah yang digunakan adalah *knowledge mining from data*, *knowledge extraction*, *data/pattern analysis*, *data archaeology*, dan *data dredging*[12]. Sementara menurut pandangan lain, *data mining* hanya sebagai langkah-langkah penting dalam

proses *knowledge discovery*[12]. Langkah-langkah yang terdapat pada proses *knowledge discovery* adalah sebagai berikut:

1. *Data cleaning* (menghilangkan *noise* dan data yang tidak konsisten)
2. *Data integration* (menggabungkan data dari beberapa sumber)
3. *Data selection* (memilih data dari *database* yang relevan dengan tugas analisis)
4. *Data transformation* (melakukan transformasi dan konsolidasi data kedalam bentuk yang sesuai untuk dilakukan *mining* dengan operasi agregasi)
5. *Data mining* (proses penting untuk melakukan ekstraksi pola data)
6. *Pattern evaluation* (mengidentifikasi pola penting yang merepresentasikan *knowledge*)
7. *Knowledge presentation* (menggunakan teknik visualisasi untuk menyajikan *knowledge* yang ditemukan kepada pengguna)

Pada tahapan *data cleaning* dan *data integration* biasa disebut sebagai proses *preprocessing*, di mana data yang dihasilkan akan disimpan pada *data warehouse*. Pada industri, media, dan lingkungan riset, istilah *data mining* sering digunakan mengenai seluruh proses *knowledge discovery*. Oleh karena itu *data mining* adalah proses menemukan pola yang menarik dan *knowledge* dari data yang jumlahnya sangat besar[12]. Sumber data dapat berupa *database, data warehouse*, website dan lain-lain. *Data mining* juga memiliki beberapa fungsi di antaranya adalah sebagai berikut:

1. *Characterization* dan *discrimination*
2. *Associations* dan *correlations*
3. *Classification* dan *regretion*
4. *Clustering analysis*
5. *Outlier analysis*

Fungsi *data mining* digunakan untuk menentukan jenis pola data yang dapat ditemukan pada *data mining tasks*. Secara umum, *tasks* tersebut dapat dikategorikan menjadi 2 jenis, yaitu prediktif dan deskriptif[12]. *Mining tasks* jenis deskriptif mencirikan sifat data pada *data set*. Sedangkan *Mining tasks* jenis prediktif melakukan induksi pada data yang ada untuk membuat sebuah prediksi. Pada penelitian tugas akhir ini, akan menggunakan fungsi *data mining* yaitu klasifikasi. Klasifikasi adalah proses untuk menemukan sebuah model yang menggambarkan dan membedakan kelas data[12]. Model diturunkan berdasarkan analisis dari *training data set*. Kemudian model tersebut digunakan untuk memprediksi label kelas dari objek yang tidak diketahui label kelasnya.

Bagaimana representasi sebuah model ? model dapat direpresentasikan dalam berbagai bentuk, contohnya *classification rules* (*IF THEN rules*), *decision trees*, formula matematika, atau *neural network*[12]. Banyak sekali metode klasifikasi yang diusulkan oleh peneliti dengan menggunakan *machine learning*, *pattern recognition*, dan statistik.

2.2.3. *Classification*

Konsep dasar klasifikasi menggambarkan pendekatan secara umum dengan membagi 2 tahapan[12]. Tahapan pertama adalah tahapan pembelajaran, membuat model klasifikasi berdasarkan data yang sudah ada sebelumnya. Kemudian tahapan kedua adalah tahapan klasifikasi, menentukan apakah akurasi dari model dapat diterima, dan jika demikian, maka model akan digunakan untuk mengklasifikasikan data baru dengan memprediksi label kelasnya.

Pada tahapan pertama, *classifier* dibangun untuk menggambarkan kelas *data set* yang telah ditentukan sebelumnya. Ini adalah langkah pembelajaran atau fase pelatihan, di mana algoritma klasifikasi membangun *classifier* dengan menganalisis atau melakukan pembelajaran dari *training set* yang terdiri dari *database tuples* dan label kelas

yang berhubungan[12]. Sebuah *tuple* ***X*** diwakili oleh *n-dimensional* ***attribute vector,*** ***X*** = (x_1, x_2, . . . ,x_n), menggambarkan *n* pengukuran yang dilakukan pada *tuple* dari *n database attributes*, masing-masing, A_1, A_2, . . , A_n. Setiap *tuple* ***X***, diasumsikan termasuk dalam kelas yang telah ditentukan seperti penentuan oleh *database attribute* lain yang disebut dengan atribut label kelas[12]. Atribut label kelas bernilai diskrit dan tidak teratur, bersifat kategorial atau nominal karena setiap nilai berfungsi sebagai kategori atau kelas. *Individual tuple* yang digunakan untuk menyusun *training set* disebut sebagai *training tuples* dan diambil secara acak dari *database* yang sedang dianalisis. Dalam konteks klasifikasi, *data tuple* dapat disebut sebagai *sample*, *instances*, atau objek[12]. Karena label kelas dari masing-masing *training tuple* telah tersedia, langkah ini juga dikenal sebagai *supervised learning*. Langkah pertama dari proses klasifikasi juga dapat dilihat sebagai *learning of function or mapping*, $y = f(\boldsymbol{X})$, yang dapat memprediksi label kelas *y* dari *tuple* ***X***.

Untuk akurasi klasifikasi, pertama, akurasi prediksi dari *classifier* akan diestimasi. Jika *training set* digunakan untuk mengukur akurasi dari *classifier*, estimasi yang muncul kemungkinan akan sangat baik, karena *classifier* cenderung *overfit* pada data. Oleh karena itu untuk mengetahui akurasi prediksi pada *classifier* menggunakan *test set*. *Test set* merupakan bagian dari *data tuple* yang tidak digunakan untuk proses pembelajaran[12].

Setelah membangun model untuk klasifikasi, akan muncul pertanyaan baru tentang seberapa bagus/akurat model yang dibangun. Setelah model dibangun maka akan dilakukan evaluasi terhadap kinerja *classifier*, evaluasi tersebut terdiri dari *accuracy* (juga dikenal sebagai tingkat pengenalan), *sensitivity* (atau *recall*), *specificity*, *precision*, F_1, dan F_β. Pada Gambar 2.2 akan dijelaskan mengenai formula yang digunakan untuk pengukuran evaluasi[12].

Measure	Formula
accuracy, recognition rate	$\frac{TP+TN}{P+N}$
error rate, misclassification rate	$\frac{FP+FN}{P+N}$
sensitivity, true positive rate, recall	$\frac{TP}{P}$
specificity, true negative rate	$\frac{TN}{N}$
precision	$\frac{TP}{TP+FP}$
F, F_1, F-score, harmonic mean of precision and recall	$\frac{2 \times precision \times recall}{precision + recall}$
F_β, where β is a non-negative real number	$\frac{(1+\beta^2) \times precision \times recall}{\beta^2 \times precision + recall}$

		Predicted class		
		yes	*no*	Total
Actual class	*yes*	*TP*	*FN*	*P*
	no	*FP*	*TN*	*N*
	Total	P'	N'	$P + N$

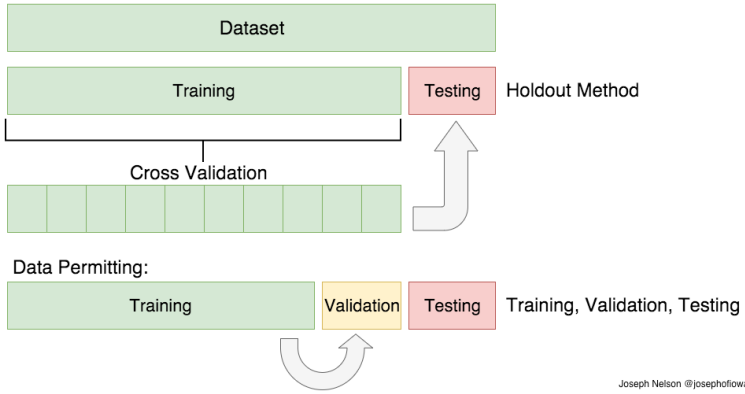
Gambar 2.2 Matriks Evaluasi Pengukuran Performa

Selain pengukuran berbasis akurasi, *classifier* juga dapat dibandingkan dengan beberapa aspek sebagai berikut:

a. *Speed*
   Mengacu pada biaya komputasi yang digunakan dalam menghasilkan dan menggunakan *classifier*
b. *Robustness*
   Kemampuan *classifier* untuk melakukan prediksi secara tepat dengan data yang memiliki *noise* atau *missing values*.
c. *Scalability*
   Mengacu pada kemampuan membangun model secara efisien dengan data yang besar.
d. *Interpretability*
   Mengacu pada tingkat pemahaman dan wawasan yang diberikan oleh *classifier*. Interpretabilitas bersifat subyektif dan oleh karena itu lebih sulit untuk dinilai. *Decision tree* dan *classification rules* dapat ditafsirkan secara mudah.

Ada beberapa teknik umum yang digunakan untuk menilai akurasi berdasarkan sampel secara acak dari data yang diberikan, yaitu *Holdout and random subsampling*, *cross-validation*, dan *bootstrap methods*. Metode *holdout* adalah apa yang telah disinggung pada pemaparan di atas mengenai akurasi. Dalam metode ini, data dipecah menjadi dua set independen, satu untuk *training set*, dan yang kedua untuk *test set*. Biasanya dua pertiga data dialokasikan untuk *training set*, dan sisanya sepertiga dialokasikan untuk *test set*. *Random subsampling* adalah variasi metode *holdout* di mana metode *holdout* diulang sebanyak *k* kali. Estimasi akurasi secara keseluruhan diambil dari akurasi rata-rata yang diperoleh dari setiap iterasi[12].

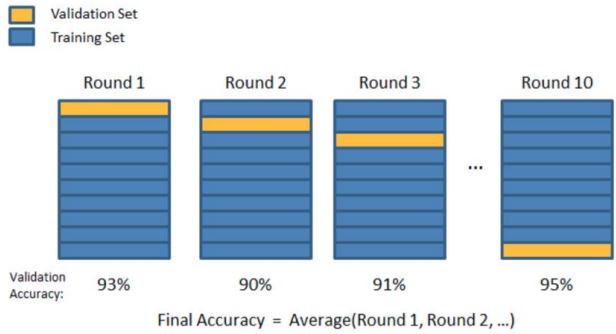
Metode *cross validation* hampir sama dengan metode *test/train set split*, tetapi diterapkan pada lebih banyak *subsets*. Artinya data akan dipecah ke *k-subsets*, dan *train set* ada pada *k-1* salah satu dari *subset*. Kemudian *subset* terakhir akan digunakan untuk *test set*.



Gambar 2.3 Matriks Evaluasi Pengukuran Performa

Ada berbagai jenis metode *cross validation*, yang pertama adalah *k-folds cross validation* dan yang kedua adalah *Leave One Out Cross Validation*. Pada *k-fold cross validation*,

data awal dibagi secara acak menjadi *subset* yang saling berbeda (*folds*), D_1, D_2, . ., D_k, masing-masing berukuran kurang lebih sama. *Training* dan *testing* dilakukan sebanyak *k* kali. Pada iterasi *i*, bagian D_i dicadangkan sebagai *test set*, dan bagian yang tersisa secara kolektif digunakan untuk *training* model. Visualisasi dari *k-folds cross validation* dapat dilihat pada Gambar 2.4.



Gambar 2.4 Visualisasi K-*Folds*

Tidak seperti metode *holdout* dan *random subsampling*, di sini setiap sampel menggunakan jumlah waktu *training* yang sama dan 1 kali untuk *testing*. Untuk klasifikasi, estimasi akurasi adalah jumlah keseluruhan klasifikasi yang benar dari iterasi *k*, dibagi dengan jumlah total *tuple* pada data awal. *Leave one out cross validation* adalah kasus khusus dari *k-fold cross validation* di mana *k* (lipatan) diatur ke jumlah *tuple* awal. Artinya hanya 1 sampel yang ditinggalkan untuk *test set*. Secara umum, *stratified 10-fold cross validation* direkomendasikan untuk mengestimasi akurasi karena bisa yang relatif terlalu rendah dan varians. Metode ini sangat membutuhkan komputasi yang mahal dan harus digunakan pada *data set* kecil. Melakukan *cross validation* dapat membantu menghindari *overfitting* lebih dari sekedar *underfitting*[12].

Berbeda dengan metode estimasi akurasi sebelumnya, metode *bootstrap* melakukan sampel pada *training tuple* secara seragam dengan penggantian. Setiap kali ada *tuple* terpilih,

kemungkinan besar akan dipilih kembali dan ditambahkan kembali ke *training set*[12].

Akurasi klasifikasi juga dipengaruhi oleh keseimbangan kelas data. Dengan data yang memiliki 2 kelas, *Class-imbalanced* terjadi ketika salah satu kelas direpresentasikan oleh hanya beberapa *tuple*, sedangkan kelas yang lain direpresentasikan oleh mayoritas *tuple*. kemudian *multiclass-imbalanced* terjadi pada data yang memiliki banyak kelas dan distribusi data setiap kelas berbeda secara substansial. Pendekatan umum yang biasa digunakan untuk meningkatkan akurasi klasifikasi dari *class-imbalanced data* adalah sebagai berikut:

a. *oversampling, undersampling*
   Oversampling dan *undersampling* mengubah distribusi *tuple* pada *training set*. Misalkan pada *training set* asli berisi 100 *tuple* positif dan 100 *tuple* negatif. *Oversampling, tuple* yang kelasnya lebih sedikit dilakukan replikasi untuk membentuk *training set* baru yang berisi 1000 *tuple* positif dan 1000 *tuple* negatif. *Undersampling*, menghilangkan *tuple* negatif secara acak sehingga *training set* baru berisi 100 *tuple* positif dan 100 *tuple* negatif.
b. *treshold moving*
   mempengaruhi model dalam membuat keputusan saat mengklasifikasikan data baru.
c. *ensemble techniques*
   membuat kombinasi untuk *classifier*. *Ensemble tehcniques* secara umum terdiri dari *bagging*, *boosting*, dan *random forest*.

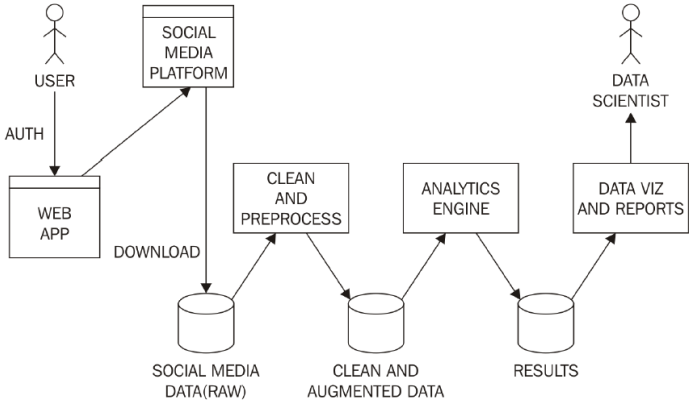
Masalah *Multiclass-imbalanced* lebih kompleks lagi, di mana *oversampling* dan *tresshold moving* kurang efektif untuk menyelesaikan masalah tersebut[12].

2.2.4. *Social Media Mining*

Keunikan data dari media sosial memerlukan teknik *data mining* baru yang dapat menangani konten pengguna

secara efisien. Sehingga memunculkan teknik baru bernama *social media mining*. *Social media mining* adalah proses merepresentasikan, menganalisis, dan mengekstrak pola data yang berasal dari media sosial[13]. Untuk melakukan *data mining* pada sosial media secara efektif, hal yang harus dilakukan adalah mengumpulkan informasi tentang individu dan entitas, mengukur interaksi mereka, dan menemukan pola untuk memahami perilaku atau kebiasaan manusia. Secara keseluruhan proses dalam *social media mining* adalah sebagai berikut:

a. *Authentication*
b. *Data collection*
c. *Data cleaning dan pre-processing*
d. *Modeling and analysis*
e. *Result presentation*



Gambar 2.5 Visualisasi Seluruh Proses dari *Social Media Mining*

Langkah *authentication* biasanya dilakukan dengan menggunakan sebuah standar yang disebut *Open Authentication* (*OAuth*). Menurut perspektif *developer*, ekosistem Python adalah salah satu *libraries* yang paling baik untuk sebagian besar platform media sosial. Sebagai *developer*, setelah

mendaftarkan aplikasi dengan layanan targetnya, platform akan memberikan *token* otorisasi yang diperlukan untuk aplikasi yang akan dibangun. Tahapan *data collection*, *cleaning*, dan *pre-processing* juga bergantung pada setiap platform media sosial [10].

2.2.5. *Opinion Mining* (Analisis Sentimen)

Istilah *opinion mining* atau biasa disebut sentimen analisis termasuk di dalam teknik *data mining*, namun akan lebih spesifik mengenai *data mining* berupa opini publik yang memiliki sentimen. *Opinion Mining* menganalisis pendapat seseorang, penilaian, sikap, dan emosi terhadap entitas, individu, isu, peristiwa, topik, dan atributnya[14]. Menurut Khusal Dave[15], menjelaskan bahwa *opinion mining tool* yang ideal akan memproses satu set hasil pencarian untuk item tertentu, menghasilkan daftar atribut produk (kualitas, fitur, dll.) dan kemudian menggabungkan pendapat tentang masing-masing (*poor*, *mixed*, *good*). Dengan pertumbuhan media sosial saat ini seperti *review*, forum diskusi, blogs, dan *social networks* di web, perseorangan maupun organisasi menggunakan konten media tersebut untuk membantu dalam proses pengambilan keputusan. Diketahui juga bahwa analisis dan evaluasi manusia terhadap informasi teks memiliki bias subjektif yang cukup besar, misalnya orang sering memberi perhatian lebih besar pada pendapat yang sesuai dengan preferensi mereka sendiri[14].

Pembahasan dilanjutkan dengan penjelasan mengenai istilah-istilah di dalam *opinion mining*. Dimulai dengan *opinion target*, secara umum opini diungkapkan untuk apa pun, misal, produk, layanan, individu, organisasi, acara, atau sebuah topik. Istilah entitas digunakan untuk menunjukkan objek target yang telah dievaluasi. Sebuah entitas memiliki kumpulan komponen dan atribut. Setiap komponen mungkin saja memiliki sub komponen dan sub atributnya. Komponen dan atribut ini disebut sebagai aspek. Kemudian seseorang yang mengungkapkan opini disebut sebagai *opinion sources* atau *opinion holder*[14]. Ada

dua tipe dari opini yaitu *regular opinion* dan *comparative opinion*. *Comparative opinion* mengekspresikan hubungan persamaan dan perbedaan antara 2 atau lebih entitas[16]. Positif, negatif, dan netral disebut *opinion orientation*, *sentiment orientation*, *semantic orientation*, atau *polarity*. Netral sering kali dianggap sebagai bukan opini. Secara formal opini adalah *quintuple*[17]. Dengan struktur:

$$(e_i, a_{ij}, oo_{ijkl}, h_k, t_l)$$

Di mana e_i adalah nama entitas, a_{ij} adalah aspek, oo_{ijkl} adalah *opinion orientation* tentang aspek a_{ij} dari entitas e_i, h_k adalah *opinion holder*, dan t_l adalah waktu ketika opini di sampaikan oleh h_k[14].

Klasifikasi sentimen dapat diformulasikan sebagai *supervised learning* dengan tiga kelas, positif, negatif, dan netral. Seperti kebanyakan *machine learning*, tugas utama dari klasifikasi sentimen adalah merancang serangkaian fitur yang efektif. Beberapa contoh fitur yang digunakan dalam penelitian adalah sebagai berikut [14]:

a. *Opinion words and phrases*
   Opinion words adalah kata yang sering digunakan untuk mengungkapkan sentimen positif atau negatif. Meskipun banyak kata opini adalah kata sifat dan kata keterangan, kata benda dan kata kerja juga dapat mengandung opini. Selain kata individual, ada juga *opinion phrases* dan *idioms* yang juga sangat penting untuk analisis sentimen.
b. *Rules of opinion*
   Ungkapan lain yang tidak mengandung kata atau ungkapan opini namun mengindikasikan pendapat atau sentimen.
c. *Negations*
   Kata-kata negasi penting karena dapat mengubah orientasi dari opini.
d. *Syntactic dependency*
   Kata-kata dalam *dependecy-based features* dihasilkan dari *dependency tree*.

Untuk mengumpulkan daftar kata opini, terdapat 3 pendekatan utama yang sudah pernah diteliti, yaitu pendekatan

secara manual, *dictionary-based*, dan *corpus-based*. pendekatan secara manual membutuhkan waktu lebih lama, biasanya dikombinasikan dengan pendekatan secara otomatis pada saat *final checking*. *Dictionary-based* adalah salah satu teknik sederhana yang berdasarkan pada *bootstraping* dengan menggunakan sekumpulan kata opini dan kamus online, misalnya *WordNet*[18]. Strategi nya adalah mengumpulkan kata-kata opini secara manual dengan orientasi yang telah diketahui, kemudian mengembangkan kumpulan kata-kata opini tersebut dengan cara mencari di *WordNet* untuk sinonim dan antonimnya. Kata-kata baru yang ditemukan ditambahkan ke daftar. Lalu iterasi berikutnya akan berjalan. Proses iterasi akan berhenti ketika tidak ada kata baru yang ditemukan[14]. Kelemahan dari *dictionary-based* ini adalah tidak mampu menemukan kata-kata opini dengan orientasi domain atau konteks yang spesifik, misal untuk *speaker handphone*, jika senyap maka negatif, namun untuk mobil, senyap adalah positif.

2.2.6. *Text Mining*

Text mining adalah proses mendapatkan informasi terstruktur dari data tekstual yang tidak terstruktur. Beberapa contoh *text mining* adalah sebagai berikut [10]:

a. *Document classification*
   Adalah tugas menentukan dokumen pada 1 atau lebih kategori
b. *Document clustering*
   Adalah tugas untuk mengelompokkan dokumen pada *subset* atau kluster yang koheren dan berbeda dengan yang lain, contohnya berdasarkan topik atau sub topik
c. *Document summarization*
   Adalah tugas untuk membuat versi pendek dari sebuah dokumen untuk mengurangi informasi yang *overload* pada pengguna, sambil tetap mempertahankan aspek terpenting yang dijelaskan pada sumber aslinya
d. *Entity extraction*

Adalah tugas untuk menemukan dan mengklasifikasikan entitas dari sebuah teks ke dalam beberapa kategori yang diinginkan seperti orang, lokasi, atau organisasi.

e. *Sentiment analysis*
Adalah tugas untuk mengidentifikasi dan mengategorikan sentimen dan opini yang diungkapkan pada teks untuk memahami sikap terhadap produk, topik, pelayanan, dan sebagainya.

2.2.7. *Natural Language Processing*

Natural Language Processing adalah disiplin ilmu yang berkaitan dengan metode untuk menganalisis secara otomatis dan memahami bahasa alami, yaitu bahasa yang ditulis atau diucapkan secara alami oleh manusia[10]. NLP selalu dikontraskan dengan *computational linguistics*. Dalam proses *natural language processing* terdapat beberapa kendala karena semakin bertambah dan berkembangnya bahasa alami manusia. Berikut ini adalah sub area dari NLP [19] :

a. *Phonology / Phonetic*
Sub area yang terkait dengan suara atau ucapan yang diinterpretasikan menjadi kata (*speech based system*). Dalam hal ini terdapat 3 aturan fonetik (*phonetic rule*) yang berfungsi untuk membedakan suara setiap kata, mengenali variasi pengucapan yang diucapkan secara bersama, dan mengenali intonasi ucapan.

b. *Morphology*
Sub area yang membahas struktur dari kata, bentuk kata dasar dan bentuk kata dengan imbuhan (prefiks dan sufiks).

c. *Syntax*
Sub area yang terkait dengan cara kata-kata digunakan untuk membentuk frase dan urutan kata dalam sebuah kalimat.

d. *Semantics*
Sub area yang terkait dengan arti kata dari sebuah kalimat.

e. *Pragmatics*

Sub area yang membahas konteks kata atau kalimat yang memiliki hubungan dengan suatu keadaan.

Kemudian terdapat 4 tugas standar NLP yang telah diteliti yaitu [20]: *Part-Of-Speech tagging* (POS), *chunking* (CHUNK), *named entity recognition* (NEC), dan *semantic role labeling* (SRL).

2.2.8. *Word Embedding*

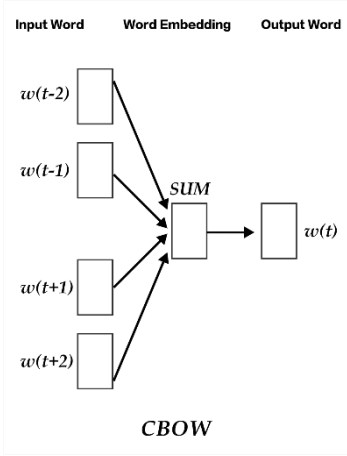
Word embedding adalah istilah untuk permodelan bahasa dan teknik pembelajaran dalam NLP di mana kata atau frase dari kosa kata direpresentasikan ke dalam sebuah *low-dimentional vector*. *Word embedding* dikenal sebagai *word representation* pada awal penemuannya oleh Rumelhart, Hinton dan Williams[21]. *word representation* adalah objek matematika yang terkait dengan setiap kata, dan sering kali merupakan vektor. Setiap nilai dimensi memiliki interpretasi semantik atau tata bahasa[22]. Selain itu *word embedding* juga dikenal dengan nama *distributed representation* karena memiliki kerapatan, menggunakan *low-dimentional vector* dan *real valued*. Vektor representasi dari kata atau frase bisa sesuai dengan dokumen di mana kata tersebut muncul dan sesuai dengan konteks kata yang ada di sekitarnya.

2.2.9. *Word2Vec*

Word2Vec adalah salah satu model yang digunakan untuk *vector representation of words* atau *word embeddings* yang dikembangkan oleh Tomas Mikolov, dkk[9]. Tomas Mikolov melakukan modifikasi hasil dari NNLM (*Neural Networks Language Model*) di mana *neural networks* dapat bekerja dengan baik dalam keteraturan linear antara kata-kata dibandingkan LSA (*Latent Semantic Analysis*) dan lebih baik dalam menangani data yang besar dibanding *LDA* (*Latent Dirichlet Allocation*)[9]. Kelebihan *Word2Vec* dalam *word embedding* atau *word representation* adalah kecepatan dalam proses *training* dan dapat menangani kumpulan data yang besar.

Menurut eksperimen yang pernah dilakukan, pengaturan yang dapat mempengaruhi kinerja dari *Word2Vec* adalah pemilihan model, ukuran vektor, *subsampling rate*, dan *training window*[23].

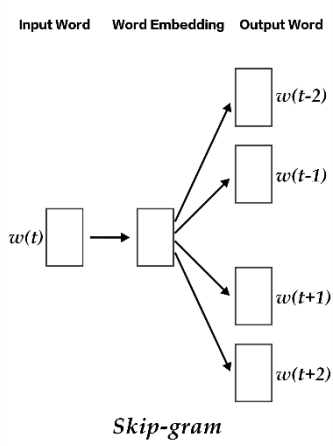
Arsitketur model yang digunakan dalam *Word2vec* ada 2 yaitu *Continous Bag-of-Words* (CBOW) dan *Continous skip-gram* di mana keduanya menggunakan model *log-linear*. Model CBOW merupakan pengembangan dari *neural net language model* yang memiliki *input layer*, *projection layer*, dan *output layer*.



Gambar 2.6 Model CBOW

Penjelasan pada gambar di atas ketika *N=2*, *w(t-2)*, *w(t-1)*, *w(t+1)*, *w(t+2)* adalah kata-kata sebelum dan sesudah yang menjadi *input word* pada *input layer*, setelah itu proses pada *projection layer* berupa operasi SUM dan menghasilkan 1 kata prediksi. Istilah yang biasa digunakan pada model CBOW adalah *predicting word given its context*. CBOW digunakan untuk memprediksi kata *(t)* berdasarkan konteks kata-kata di sekitarnya *(c)*, yang bertujuan memperbesar peluang *P=(t | c)* pada *training dataset*. Kelebihan dari CBOW adalah proses *training* yang lebih cepat daripada model *Skip-Gram*, sedikit

lebih bagus akurasinya untuk kata yang sering muncul, dan sangat efektif untuk *data set* kecil[9].
Jika model CBOW memprediksi kata yang berasal dari konteks, model *Skip-Gram* justru sebaliknya, memprediksi konteks dari kata yang diberikan.



Gambar 2.7 Model Skip-Gram

Penjelasan dari gambar di atas ketika *N=2*, sama seperti model CBOW, namun *w(t)* sebagai *input word* pada *input layer*, kemudian proses pada *projection layer* berupa operasi SUM dan menghasilkan *w(t-2)*, *w(t-1)*, *w(t+1)*, *w(t+2)* yaitu kata-kata yang berada di sekitarnya. Kelebihan dari model *Skip-Gram* adalah dapat bekerja dengan baik pada data yang tidak terlalu besar, dan mampu merepresentasikan dengan baik kata/frasa yang jarang[9].
Word2Vec juga memiliki 2 metode *training*, yaitu *negative-sampling* dan *hierarchial softmax*. Menurut Tomas Mikolov *Negatif sampling* merupakan metode yang cocok untuk model *Skip-gram*[23]. Metode *training Hierarchial Softmax* merupakan pendekatan perhitungan yang efisien dari *full softmax*. Kelebihan metode *Hierarchial softmax* adalah untuk mengevaluasi hanya sekitar *log2(W) nodes* daripada harus

mengevaluasi seluruh *output nodes W* pada *neural network* untuk mendapatkan kemungkinan distribusi.

2.2.10. *Machine Learning*

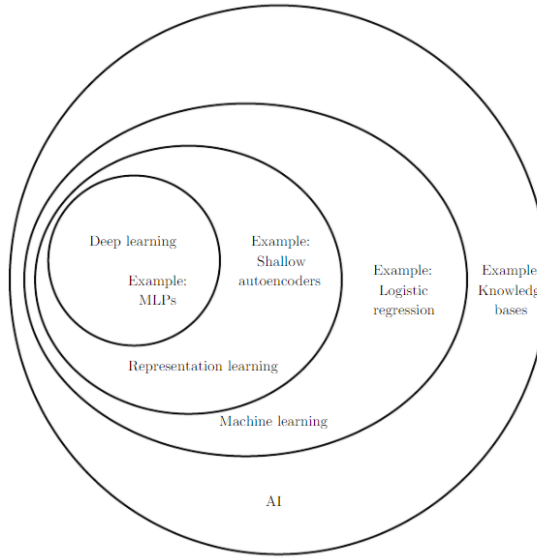
Machine learning adalah bidang ilmu yang mempelajari dan mengembangkan algoritma untuk pembelajaran, dan membuat prediksi pada data[10]. Beberapa aplikasi yang menerapkan *machine learning* contohnya adalah sebagai berikut:

a. Memutuskan apakah email yang masuk adalah *spam* atau bukan.
b. Memilih topik artikel berita dari daftar subjek yang diketahui.
c. Menganalisis transaksi bank untuk mengidentifikasi upaya penipuan.

Dan masih banyak contoh aspek lain dari penerapan *machine learning* dalam kehidupan sehari-hari. Beberapa metodologi yang paling populer dapat dikategorikan secara sederhana menjadi *supervised learning* dan *unsupervised learning*. *Supervised learning* digunakan untuk memecahkan permasalahan seperti klasifikasi, di mana data di lengkapi dengan atribut yang ingin diprediksi, misalnya label kelas. Algoritma yang biasa digunakan meliputi *Naive Bayes*, *Support Vector Machine*, *Neural Network*, dan masih banyak lagi. *Unsupervised learning* digunakan pada permasalahan di mana data yang masuk tanpa nilai keluaran yang sesuai seperti *clustering*[10].

2.2.11. *Deep Learning*

Deep Learning juga dikenal sebagai *deep structured learning* atau *hierarchical learning* adalah bagian dari metode *machine learning*[8].

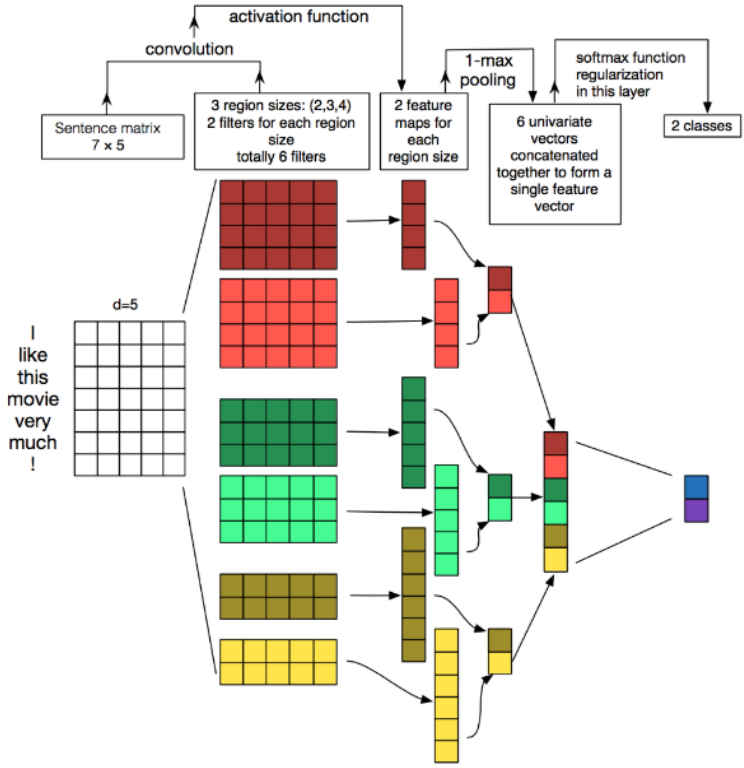


Gambar 2.8 Diagram Venn Menunjukkan *Deep Learning* Merupakan Bagian dari *Representation Language*

Deep learning merupakan pengembangan dari pendekatan terhadap *artificial Intelligence*. *Deep learning* adalah jenis *machine learning* tertentu yang menghasilkan kekuatan dan fleksibilitas yang besar.

2.2.12. *Convolutional Neural Network*

Istilah *Convolutional Neural Network* pertama kali digunakan oleh LeCun, dkk. yang sebenarnya diciptakan untuk *computer vision*. CNN memanfaatkan layer dengan filter konvolusi yang diterapkan pada *local features*[24]. Model CNN kemudian terbukti efektif untuk *Natural Language Processing* termasuk untuk melakukan tugas klasifikasi teks[7][25][26].



Gambar 2.9 Ilustrasi Arsitektur CNN untuk Klasifikasi Teks

Pada CNN untuk tugas NLP, *input* nya adalah kalimat atau dokumen yang direpresentasikan sebagai matriks. Setiap baris pada matriks merupakan *token*, biasanya sebuah kata, atau bisa berupa karakter yang berbentuk vektor. Penjelasan pada gambar di atas, kalimat yang akan dijadikan *input* adalah "*I like this movie very much !"* dan dijadikan matriks dengan ukuran 7x5. Kemudian digambarkan 3 ukuran wilayah filter 2,3, dan 4, yang masing-masing memiliki 2 filter. Setiap filter melakukan konvolusi pada matriks kalimat dan menghasilkan *feature-maps*. Kemudian *1-max pooling* dilakukan di setiap *map*, angka terbesar dari hasil *pooling* setiap *map* ditangkap. Dengan

demikian *unvariate feature vector* dihasilkan dari seluruh 6 *maps*, dan setelah itu digabungkan untuk membentuk *feature vector* untuk *penultimate layer*. Layer *softmax* terakhir kemudian menerima *feature vector* sebagai *input* dan menggunakannya untuk klasifikasi kalimat[26].

Terdapat 4 model yang telah diteliti oleh *Yoon Kim*, yaitu sebagai berikut [7]:

a. CNN-rand
   Model dasar di mana semua kata diinasalisasi secara acak dan kemudian dimodifikasi selama proses *training*.
b. CNN-static
   Model dengan *pre-trained vector* dari *Word2Vec*. Semua kata tetap statis dan hanya parameter lain dari model yang dipelajari.
c. CNN-non-static
   Sama seperti sebelumnya tetapi *pre-trained vector* disetel untuk masing-masing tugas.
d. CNN-multichannel
   Model dengan dua set *word vector*. Setiap set *vector* diperlakukan sebagai *channel* dan setiap filter diterapkan.

Membangun arsitektur CNN berarti ada banyak pilihan *hyperparamater*, beberapa di antaranya adalah : *input representation* (seperti *Word2Vec*, *GloVe*, dan *one-hot*), jumlah dan ukuran filter konvolusi, *pooling strategies* (seperti *max*, *average*, *k-max*, dan *dynamic*), dan fungsi aktivasi (*RelU*, *tanh*)[26].

2.2.13. Politik

Secara umum politik adalah berbagai macam kegiatan dalam sistem negara yang menyangkut proses penentuan dan pelaksanaan tujuan dari sistem-sistem yang ada. Konsep pokok dalam ilmu politik adalah [27] : negara, kekuasaan, pengambilan keputusan, kebijakan umum, dan pembagian. Dalam *contemporary political science*, yang diterbitkan UNESCO pada tahun 1950, ilmu politik dibagi menjadi 4

bidang yaitu teori politik, lembaga-lembaga politik, partai atau golongan, dan hubungan internasional.

BAB III
METODOLOGI PENELITIAN

Pada bab ini akan dijelaskan tentang metodologi yang digunakan. Metodologi dibuat sebagai panduan agar sistematis dan terstruktur. Adapun tahapan dalam pengerjaan penelitian tugas akhir ini adalah sebagai berikut:

3.1. Tahapan Pelaksanaan Penelitian Tugas Akhir

Pada bagian ini akan dijelaskan mengenai metodologi atau tahapan pelaksanaan penelitian tugas akhir.

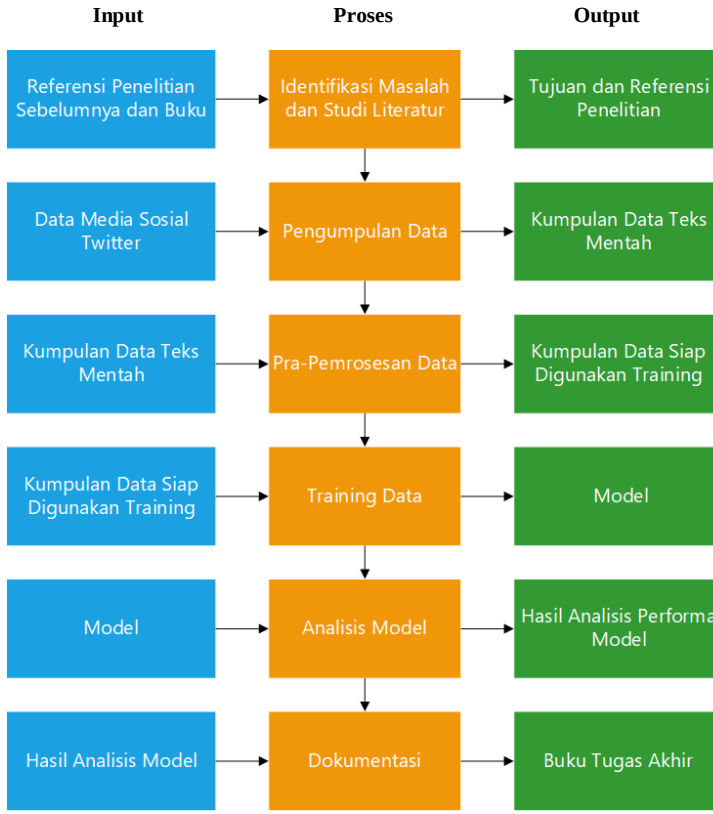


Diagram 3.1 Tahapan Pelaksanaan Penelitian Tugas Akhir

Untuk menjelaskan ringkasan pada keseluruhan proses penelitian tugas akhir ini, maka akan dijelaskan pada Diagram 3.2.

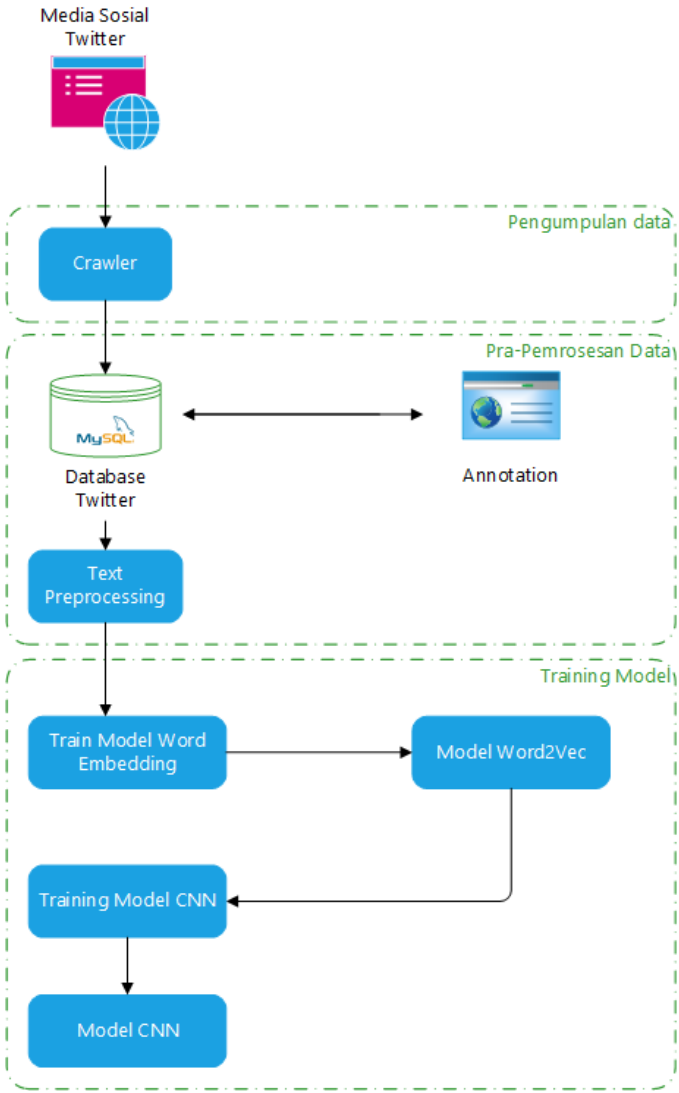


Diagram 3.2 Arsitektur Penelitian

3.1.1. Identifikasi Masalah dan Studi Literatur

Aktivitas pertama yang akan dilakukan pada tahap ini adalah melakukan identifikasi masalah terkait dengan objek penelitian dan metode yang akan digunakan. Kemudian aktivitas selanjutnya adalah melakukan studi literatur menggunakan referensi penelitian sebelumnya dan buku yang berkaitan dengan topik penelitian tugas akhir ini. Literatur yang digunakan adalah penelitian dengan topik *Convolutional Neural Network* dan *Natural Language Processing*. *Paper* utama yang dijadikan pedoman pada penelitian tugas akhir ini lebih spesifik membahas tentang *Convolutional Neural Network for Text* yang berjudul "*Convolutional Neural Networks for Sentence Classification*", dan "*A Sensitivity Analysis of (and Practitioner's Guide to) Convolutional Neural Networks for Sentence Classification*". Metode pada ketiga *paper* tersebut yang nantinya akan dijadikan dasar untuk pengerjaan penelitian tugas akhir ini.

3.1.2. Pengumpulan Data

pada tahap ini dilakukan proses pengumpulan data dari platform media sosial *twitter*. Teknik pengambilan data yang dilakukan adalah *crawling* dengan menggunakan API yang disediakan oleh platform media sosial *twitter*. Untuk mendapatkan data sesuai studi kasus "politik" yang akan diteliti, proses pengambilan data menggunakan beberapa kata kunci yang berkaitan dengan politik. Untuk kata kunci yang digunakan dijelaskan pada Tabel 3.1.

Tabel 3.1 Daftar Kata Kunci untuk Topik Politik

Kategori	Entitas
Partai Politik	Partai Amanat Nasional
	Partai Berkarya
	PDI Perjuangan
	Partai Demokrat

	Partai Gerindra
	Partai Gerakan Perubahan Indonesia
	Partai Golkar
	Partai Hanura
	Partai Keadilan Sejahtera
	Partai Kebangkitan Bangsa
	Partai Nasional Demokrat
	Partai Persatuan Indonesia
	Partai Persatuan Pembangunan
	Partai Solidaritas Indonesia
Tokoh Politik	Jokowi
	Prabowo
	Gus Ipul
	Deddy Mizwar
	Ganjar Pranowo
	Khofifah Indar P
	Sudirman Said
	Setya Novanto
	Fahri Hamzah
	Fadli Zon
	Susilo Bambang Yudhoyono
	Ridwan Kamil
Isu Politik	Reuni Akbar 212
	Pilkada
	UU MD3
	Pilgub Jawa Timur
	Pilgub Jawa Barat
	Pilgub Jawa Tengah

Kemudian pengambilan data juga dilakukan tanpa kata kunci / filter yang akan digunakan untuk proses *word*

embeddings. Secara umum seluruh data yang dikumpulkan harus berbahasa Indonesia. Data yang didapatkan kemudian di simpan ke dalam bentuk file JSON.

3.1.3. Pra-Pemrosesan Data

Data yang diperoleh dari proses pengumpulan data akan dilakukan pra-pemrosesan data yang akan digunakan untuk proses *training* model. Visualisasi dari pra-pemrosesan data adalah pada Diagram 3.3.

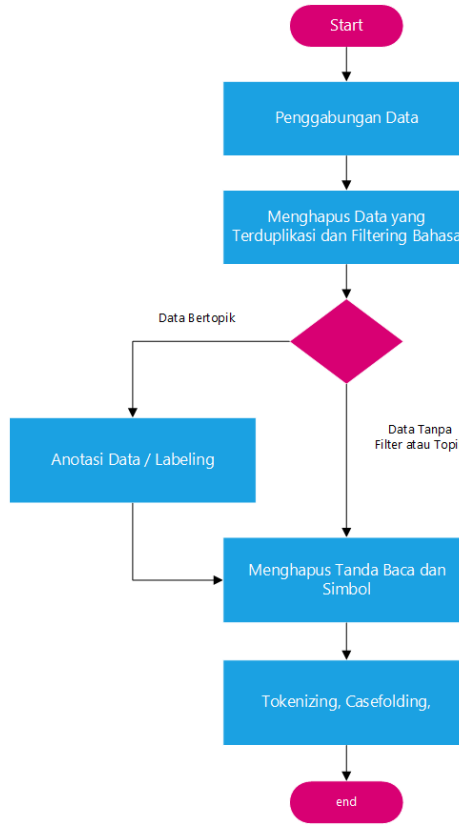


Diagram 3.3 Pra-Pemrosesan Data

3.1.3.1. Penggabungan Data

Data yang didapat dari proses pengumpulan data masih terpisah. Pada satu platform media sosial, data terbagi ke beberapa file sesuai dengan kata kunci yang digunakan pada proses *crawling*. Proses penggabungan data juga akan mengubah bentuk file JSON kemudian disimpan pada 1 basis data MySQL.

3.1.3.2. Menghapus Data yang Terduplikasi

Data yang sudah digabung ke dalam *database* akan dilakukan deteksi terhadap duplikasi atau kesamaan konten atau isi pada atribut teks. Proses deteksi duplikasi dilakukan dengan kombinasi *query* dan pemeriksaan secara manual pada proses akhirnya. Lalu untuk data yang tidak berbahasa Indonesia juga akan dihapus dari *database* agar diperoleh data teks yang lebih layak dianalisis pada tahap selanjutnya.

3.1.3.3. Anotasi Data / *Labeling*

Pada tahap ini akan dilakukan anotasi atau pelabelan data. Metode yang digunakan adalah *Human Intelligence Task* di mana proses ini harus memiliki tingkat persetujuan lebih besar dari 95%. Setiap tugas akan dilakukan oleh sedikitnya 3 orang. Masing-masing orang akan menentukan polaritas keseluruhan teks serta polaritas teks terhadap topik target dengan skala 5 poin.

Tabel 3.2 Contoh Pelabelan Data

Teks	Label Sentimen	Label Topik
Gerindra: Kebiasaan Pencitraan dan Bohongi Rakyat Hasilkan Istilah HoaxMembangun	Sangat Negatif	Gerindra: Netral
Prabowo hatinya lapang bahkan Jokowi ahok sudah beliau maafkan Jadi percuma berharap	Sangat Positif	Prabowo : Sangat Positif

La Nyalla bakal tinju-tinjuan sama Prabowo nyatanya mereka saling rangkulan, orang baik sulit diadu domba.. Cebong gagal girang		
Rakyat yg mna? Pliss @PDIP_Online @PDI_Perjuangan @budimandjatmiko masih bnyk kandidat lain yg lbih baik...tlg jgn buat Rakyat Ilfiel	Sangat Negatif	PDIP : Sangat Negatif

Label dibagi dalam skala 5 poin yaitu, sangat positif, positif, netral, negatif, dan sangat negatif. Nilai representasi dari setiap label adalah +2, +1, 0, -1, -2. Kemudian dalam pemilihan label dilakukan prosedur sebagai berikut :

a. Jika 2 dari 3 orang yang melakukan pelabelan menyetujui label yang sama, maka label tersebut akan diterima secara langsung.
b. Jika tidak, nilai representasi dari seluruh label akan dihitung nilai rata-ratanya lalu dibulatkan ke nilai integer terdekat. Untuk menyeimbangkan kecenderungan rata-rata untuk menjauh dari -2 dan 2, dan juga tidak memilih 0. Maka pembulatan dilakukan pada ± 0.4 dan ± 1.4.

3.1.3.4. Menghapus Tanda Baca dan Simbol

Pada tahap ini kumpulan data *twitter* akan di lakukan pembersihan terhadap data teksnya. Penghapusan menggunakan metode *replacing* pada pola karakter tertentu. Simbol yang memiliki makna seperti emoticon diterjemahkan ke dalam bentuk makna teksnya sehingga merepresentasikan perasaan yang diungkapkan oleh pembuat *tweet* terhadap suatu objek.

3.1.3.5. *Case Folding*

Pada tahap ini teks pada data akan diubah menjadi bentuk *lowercase*. Tujuan dari tahap ini adalah untuk menyeragamkan bentuk penulisan pada teks yang akan diproses ketika proses *training*.

Tabel 3.3 Proses *Casefolding*

Sebelum *Case Folding*	**Setelah *Case Folding***
Gerindra: Kebiasaan Pencitraan dan Bohongi Rakyat Hasilkan Istilah HoaxMembangun	gerindra: kebiasaan pencitraan dan bohongi rakyat hasilkan istilah hoaxmembangun
Prabowo hatinya lapang bahkan Jokowi ahok sudah beliau maafkan Jadi percuma berharap La Nyalla bakal tinju-tinjuan sama Prabowo nyatanya mereka saling rangkulan, orang baik sulit diadu domba.. Cebong gagal girang	prabowo hatinya lapang bahkan jokowi ahok sudah beliau maafkan jadi percuma berharap la nyalla bakal tinju-tinjuan sama prabowo nyatanya mereka saling rangkulan, orang baik sulit diadu domba.. cebong gagal girang

3.1.3.6. *Tokenizing*

Pada tahap ini akan dilakukan proses *tokenizing* atau pemisahan kata dari data teks. *Token* yang telah terbentuk digunakan untuk merepresentasikan sebuah kata ke dalam vektor menggunakan metode *word embedding*.

3.1.4. *Training* Model

pada tahap ini akan dilakukan proses pembuatan model dengan melakukan proses *training* pada model *word embeddings* dengan *Word2Vec* dan model *Convolutional Neural Network.*

3.1.4.1. Pembuatan Model *Word2Vec*

Proses pembuatan model *word2vec* memanfaatkan *library gensim* pada Python. Dalam proses pembuatan model, akan dibuat beberapa skenario dengan mengubah pengaturan di setiap skenarionya. Tujuannya adalah untuk mengukur kinerja dari pembuatan model sehingga dapat diperoleh model dengan akurasi tertinggi.

3.1.4.2. Pembuatan Model CNN

Proses pembuatan model CNN akan dibangun menggunakan *library pytorch* pada Python. CNN akan menggunakan proses konvolusi dalam setiap aktivitas urutan pembelajarannya. Sama seperti dalam proses pembuatan model sebelumnya, pada proses ini juga akan dibuat beberapa skenario dengan perbedaan pengaturan parameter, sehingga dapat diketahui performa dari proses *training* yang dilakukan.

3.1.5. Analisis Model

Dari hasil proses *training* dengan beberapa skenario yang dibuat, akan dilakukan proses analisis terhadap model yang dihasilkan. Pada proses ini juga akan dilakukan pengamatan dan perhitungan terhadap keseluruhan performa dari proses *training* yang dilakukan. Analisis tersebut termasuk di dalamnya untuk dilakukan pengujian terhadap kumpulan data baru, untuk mengukur sejauh mana model yang dihasilkan mampu melakukan klasifikasi teks dengan tepat.

3.1.6. Dokumentasi

Tahap akhir dari penelitian ini adalah penyusunan laporan tugas akhir yang bertujuan untuk mendokumentasikan langkah-langkah pembuatan tugas akhir secara rinci, mendokumentasikan hasil kajian dan analisis, serta mendokumentasikan hasil pembuatan tugas akhir dan kesimpulan dari pengerjaan penelitian tugas akhir ini. Selain itu,

penyusunan laporan tugas akhir juga bertujuan agar penelitian ini dapat dimanfaatkan baik di masa yang akan datang.

BAB IV
PERANCANGAN

Pada bab ini akan dijelaskan mengenai rancangan dari luaran penelitian tugas akhir ini. Perancangan yang dibuat berupa rancangan *crawler*, rancangan pra-pemrosesan data, dan rancangan model.

4.1. Akuisisi Data Media Sosial

perancangan dimulai dengan melakukan pengumpulan data dari media sosial yang dibutuhkan untuk proses pembuatan model *word embeddings* dan model CNN. *Crawler* akan dirancang untuk mengambil data secara otomatis dengan jumlah besar. Pengambilan data akan dilakukan secara periodik yaitu setiap 1 minggu sekali pada setiap kata kunci, hal tersebut dilakukan untuk mengurangi duplikasi dari data yang diambil. *Crawler* mengambil data dari media sosial *twitter* dengan menggunakan kata kunci yang telah dijelaskan pada sub bab 3. 1. 2 yaitu tokoh politik, isu politik dan partai politik. Kemudian untuk memperoleh data dengan jumlah banyak yang akan digunakan pada proses pembuatan model *word embeddings*, pengambilan data akan menggunakan kata kunci yang umum dalam Bahasa Indonesia.

4.2. Perancangan *Crawler*

Pengambilan data akan dilakukan secara periodik dengan menggunakan sebuah *crawler*. Oleh karena itu akan dirancang sebuah *crawler* yang dapat mengambil dan menyimpan data dari media sosial *twitter* dengan kata kunci yang telah ditentukan.

4.2.1. Desain *Crawler*

Kode program *crawler* dibuat menggunakan bahasa pemrograman Python. Kode program *crawler* menggunakan *library tweepy* untuk meningkatkan efisiensi dan efektivitas pengambilan data dari *twitter*. *Library* tersebut menyediakan berbagai kemudahan dalam menyediakan akses ke seluruh metode *API RESTful twitter*. Metode yang digunakan dalam

pengambilan data *twitter* pada penelitian ini adalah *search*, yaitu dengan mengambil data *twitter* dari periode sebelumnya, atau biasa dikenal dengan *backward crawling*. Karena *twitter* memiliki keterbatasan akses jumlah data yang dapat diambil pada metode *search*, maka kode program yang berasal dari *library tweepy* akan dilakukan kustomisasi.

4.2.2. Desain Basis Data

Untuk menyimpan kumpulan data yang diperoleh dari *crawling* data *twitter*, maka perlu menggunakan basis data dengan DBMS MySQL. Atribut dari data *twitter* yang disimpan serta tipe datanya dapat dilihat pada Tabel 4.1.

Tabel 4.1 Desain Tabel untuk Hasil Data *Crawling*

Atribut	Tipe Data	Penjelasan	Batasan
id_tweet	*bigint*	Berisi *id* unik dari setiap *tweet*	*Not Null*
topic	*varchar*	Berisi topik atau kata kunci dari *tweet* yang digunakan dalam *crawling*	*Not Null*
created_at	*date*	Berisi tanggal dari dibuatnya sebuah *tweet*	*Not Null*
text	*varchar*	Berisi pesan teks dari *tweet* yang di publikasikan	*Not Null*
username	*varchar*	Berisi *username* dari pemilik *tweet*	*Not Null*
location	*varchar*	Berisi lokasi di mana *tweet* dipublikasikan oleh pengguna	*Null*

4.3. Perancangan Pra-Pemrosesan Data

Pra-pemrosesan data akan melakukan transformasi pada kumpulan data hasil *crawling* yang diperlukan untuk proses pembuatan model *word embeddings* dan model CNN. Tahapan yang dilakukan dalam pra-pemrosesan data adalah penggabungan kumpulan data, penghapusan kumpulan data terduplikasi, *filtering* Bahasa Indonesia, anotasi data, penghapusan tanda baca dan simbol, *tokenizing* dan *casefolding*.

4.3.1. Perancangan Penggabungan Kumpulan Data

Proses pembuatan model *word embedding* selanjutnya akan membutuhkan kumpulan data *tweet* yang diperoleh dari beberapa sumber penelitian lain yang sama dengan studi kasus *e-commerce* dan telekomunikasi. Oleh karena itu akan dilakukan penggabungan data hasil *crawling twitter* dari seluruh studi kasus lain. Kumpulan data dari studi kasus lain akan diubah dalam bentuk *query* SQL. Kemudian akan digabung dalam 1 basis data yang sama.

4.3.2. Perancangan Penghapusan Kumpulan Data yang Terduplikasi

Untuk mendapatkan kumpulan data teks yang unik maka akan dilakukan penghapusan data yang terduplikasi menggunakan *query* SQL pada basis data. *Query* yang digunakan untuk mendeteksi dan menghapus data *tweet* yang terduplikasi dapat dilihat pada Kode 4.1.

```
INSERT INTO tweet_temp
SELECT * FROM tweet GROUP BY text
ORDER BY text;
```

Kode 4.1 Menghapus Kumpulan Data Terduplikasi

Kumpulan data teks yang unik akan disimpan pada tabel baru.

4.3.3. Perancangan *Filtering* Bahasa Indonesia

Kumpulan data *tweet* yang diperoleh terkadang mengandung bahasa asing. Kemudian pada proses pembuatan model *word embedding*, data yang dibutuhkan harus mengandung bahasa Indonesia seluruhnya. Sehingga harus dilakukan *filtering* bahasa pada kumpulan data *tweet*. Kode program Python akan dirancang untuk mendeteksi bahasa yang terkandung didalam kumpulan data *tweet*. Kode program tersebut menggunakan *library langdetetct* yang memiliki algoritma untuk mendeteksi bahasa yang terkandung dalam suatu teks.

4.3.4. Perancangan Anotasi Data

Anotasi dilakukan untuk memberikan label sentimen pada data *tweet*. Data *tweet* berlabel akan diperlukan pada pembuatan model klasifikasi teks CNN. Pemberian label dilakukan dengan menggunakan aplikasi berbasis web yang menggunakan *framework Code Igniter*. Data *tweet* yang diberi label hanya diambil sekitar 10.000 data dari kumpulan data *tweet* pada basis data.

Untuk menyimpan data pada aplikasi anotasi, akan digunakan basis data MySQL dengan desain tabel yang ditunjukkan pada Tabel 4.2.

Tabel 4.2 Desain Tabel Anotasi Data

Atribut	Tipe Data	Penjelasan	Batasan
id_tagging	*int*	Berisi *id* unik dari aktivitas pemberian label	*Not Null*
id_user	*int*	Berisi *id* dari pengguna yang melakukan pemberian label	*Not Null*
id_tweet	*bigint*	Berisi *id* dari *tweet* yang diberi label	*Not Null*

label_tweet	*int*	Berisi label sentimen dari *tweet*	*Not Null*
label_topic	*int*	Berisi label sentimen terhadap topik dari *tweet*	*Not Null*
sarkasme	*int*	Berisi label sarkasme dari *tweet*	*Null*

4.3.5. Perancangan Penghapusan Tanda Baca dan Simbol

Pada kumpulan data *tweet* masih terdapat beberapa karakter pada data teksnya yang tidak memiliki makna dan kurang berpengaruh pada pembuatan model *word embedding* dan CNN, sehingga harus dilakukan penghapusan karakter terutama tanda baca dan simbol. Namun penelitian ini tidak akan dilakukan penghapusan atau perubahan secara signifikan terhadap kumpulan data *tweet*.

Kode program Python dibuat untuk melakukan penghapusan tanda baca dan simbol. Untuk mendeteksi karakter pada teks data *tweet*, *regular expression* akan digunakan pada kode program tersebut. Daftar sampel tanda baca dan simbol yang akan dihapus adalah sebagai berikut :

1. Menghapus simbol "@" beserta teks yang mengikutinya dan diganti menjadi <mention>
2. Menghapus simbol "RT"
3. Mengganti simbol *tag* "#" kemudian diganti menjadi <hashtag>
4. Menghapus *url* yang terdapat pada *tweet*
5. Menghapus elemen HTML
6. Menghapus simbol titik berulang
7. Menghapus *emoticon* dan *emoji* kemudian diganti dengan makna dalam bentuk teks nya
8. Menghapus seluruh angka yang terdapat pada *tweet*

4.3.6. Perancangan *Tokenizing* dan *Casefolding*

Kumpulan data *tweet* luaran dari tahap sebelumnya akan dilakukan *tokenizing* dan *casefolding*. *Tokenizing* dilakukan hanya pada data yang digunakan untuk *training word embedding*. Hal tersebut berfungsi untuk mengubah bentuk *tweet* menjadi potongan-potongan kata. Sedangkan *casefolding* digunakan untuk mengubah seluruh teks kedalam bentuk *lowercase*. Untuk melakukan *tokenizing* dan *casefolding*, kode program dibuat dengan menggunakan bahasa pemograman Python.

4.4. Perancangan Pembuatan Model *Word2Vec*

Kumpulan data *tweet* yang telah dilakukan pra-pemrosesan data akan digunakan untuk membuat model *word embeddings.* Pada penelitian ini, pembuatan model *word embeddings* akan menggunakan algoritma *word2vec*. Kode program yang dibuat untuk melakukan *training* model *word2vec* menggunakan bahasa pemrograman Python dengan memanfaatkan *library gensim*.

Dalam penelitian ini *training* model word2vec akan dilakukan beberapa kali dengan konfigurasi parameter yang berbeda sesuai dengan skenario jumlah penggunaan model *word embeddings* pada pembuatan model CNN. Berikut ini adalah beberapa parameter yang diubah dalam proses *training Word2Vec*.

1. *Learning Algorithm* : merupakan parameter yang menentukan arsitektur model pada saat *training*. Terdapat dua jenis *learning algorithm*, yaitu *Skip-Gram* dan *Continuous Bag of Words*.
2. *Training Algorithm* : merupakan parameter yang menentukan jenis algoritma *training Word2Vec*. Terdapat 2 jenis *training algorithm* yaitu *Negative Sampling* dan *Heirarchical Softmax*.

3. *Layer Size / Dimention Size* : merupakan parameter yang menentukan jumlah *hidden layer* dan dimensionalitas dari vektor.
4. *Window Size* : merupakan parameter yang menentukan jumlah kata sesudah dan sebelum dari sebuah kata yang termasuk konteks dari kata itu.
5. *Minimum Word Frequency* : merupakan parameter yang menentukan nilai minimal sebuah kata muncul yang akan masuk didalam *training*.
6. *Iterations* : merupakan parameter untuk menentukan berapa banyak jumlah *neural net* dapat mengubah koefisiennya dalam sekali proses mini *training*.
7. *Epoch* : merupakan parameter untuk menentukan jumlah iterasi proses *training*

4.5. Perancangan Model *Convolutional Neural Network*

Model CNN dihasilkan dari proses *training* kumpulan data *tweet* berlabel. Data *tweet* berlabel merupakan luaran dari tahap anotasi data. Untuk mendapatkan model CNN dengan hasil terbaik pada penelitian ini, proses *training* model dilakukan beberapa kali dengan melakukan *tuning hyperparameter*. Parameter yang digunakan untuk percobaan *training* dijelaskan pada skenario *training* sub bab 4.5.1.

Kode program yang digunakan untuk menjalankan proses *training* model CNN menggunakan *library deep learning Pytorch*. Kemudian model *word2vec* yang dihasilkan pada tahap sebelumnya akan digunakan untuk mengubah *input* data teks menjadi *input* vektor. Selain menggunakan model *word2vec* yang dihasilkan dari tahap sebelumnya, *training* model CNN juga akan menggunakan model *word embedding* bojanowski yang menggunakan algoritma *fastext*.

4.5.1. Perancangan Skenario *Training*

Untuk melakukan percobaan *training* model CNN maka akan dibuat beberapa skenario. Skenario yang dibuat terdiri dari

kombinasi parameter yang dianggap memiliki pengaruh terhadap performa model. Skenario parameter yang digunakan dapat dilihat pada Tabel 4.3.

Tabel 4.3 Rancangan Skenario Parameter *Training* Model CNN

Parameter	Nilai	Keterangan
Model *Word Embeddings*	1. Model *word2vec* a. *Skip-gram* b. CBOW 2. Model Bojanowski	Model yang digunakan untuk mengubah *input* teks menjadi *input* vektor. Model *word2vec* adalah model yang dihasilkan dari tahap pembuatan model *word2vec* sebelumnya
Filter Region Size	1. *Single region size* 1 - 10 2. *Multiple region size*	Menentukan ukuran dari region *size* pada proses konvolusi input vektor. Untuk *multiple region size* menggunakan kombinasi lebih dari 1 nilai
Feature Maps	1. 100 2. 200 3. 300 4. 400 5. 500 6. 600	Menentukan ukuran dari *feature maps* pada model CNN
Embedding Mode	1. *Static* 2. *Non-static*	Menentukan cara belajar dari *word embeddings*
Subtask	1. *Subtask* A 2. *Subtask* B 3. *Subtask* C	Jenis *subtask* yang dilakukan *training*

Kemudian parameter lain yang tidak dilakukan perubahan pada proses *training* atau sebagai konfigurasi dasar model dapat dilihat pada Tabel 4.4.

Tabel 4.4 Konfigurasi Dasar Model CNN

Parameter	Nilai
Epoch	10
Batch Size	50
Fold	10
Activation Function	*ReLu*
Optimizer	adadelta
Pooling Strategy	1-Max pooling

4.5.2. Perancangan Evaluasi Model CNN

Performance metrics digunakan untuk mengevaluasi model yang dihasilkan dari proses *training* model CNN. Evaluasi dilakukan untuk mengetahui model terbaik yang dihasilkan dan menganalisis performa dari *training* yang berjalan sesuai skenario yang telah dibuat.

Jenis pengukuran dalam *performance metrics* yang digunakan pada setiap *subtask* adalah sebagai berikut:

1. *Precision*
   Untuk mengukur ketepatan prediksi label kelas, ditunjukkan pada persamaan 4.1.

$$Precision = \frac{T_p}{T_p + F_p} \quad (4.1)$$

   Dimana:
   T_p : Jumlah label yang diprediksi dengan benar
   F_p : Jumlah label yang diprediksi salah

2. *Recall*
   Untuk mengukur label kelas aktual yang diprediksi dengan tepat, ditunjukkan pada persamaan 4.2.

$$Recall = \frac{T_p}{T_p + F_n} \quad (4.2)$$

Dimana:
T_p : Jumlah label yang diprediksi dengan benar
F_n : Jumlah label yang tidak diprediksi dengan benar

3. *F-Measure.*
Untuk mengukur keseimbangan *precision* dan *recall* dari label kelas, yang ditunjukkan pada persamaan 4.3.

$$F\ Measure = 2 * \frac{(precison * recall)}{(precison + recall)} \quad (4.3)$$

4. *Accuracy*
Untuk mengukur akurasi dari model yang dihasilkan, ditunjukkan pada persamaan 4.4.

$$Accuracy = \frac{T_p + T_n}{T_p + F_p + T_n + F_n} \quad (4.4)$$

5. *Average Deviation Standard*
Untuk mengukur rata-rata standar deviasi dari distribusi label kelas, ditunjukkan pada persamaan 4.4.

$$s = \sqrt{\frac{n \sum_{i=1}^{n} x_i{}^2 - (\sum_{i=1}^{n} x_1)^2}{n(n-1)}} \quad (4.5)$$

Dimana:
S : Standar deviasi
n : Ukuran sampel
x_i : Nilai x ke i

6. *Macro Average Mean Absolute Error*
Untuk mengukur besarnya kesalahan rata-rata dalam satu set prediksi, ditunjukkan pada persamaan 4.5.

$$MAE^M(h, Te) = \frac{1}{|C|} \sum_{j=1}^{|C|} \frac{1}{|Te_j|} \sum_{x_i \in Te_j} |h(x_i) - y_i| \qquad (4.5)$$

Dimana:
C : Jumlah label di dalam class
Y_i : Label Actual pada item xi
$h(x_i)$: Predicted Label
Te_j : Class label data actual

Halaman ini sengaja dikosongkan

BAB V
IMPLEMENTASI

Pada bab ini akan dijelaskan mengenai proses implementasi penelitian dari perancangan yang telah dilakukan sesuai dengan metode pengembangan yang dibuat sebelumnya.

5.1. Persiapan Implementasi

Penelitian ini menggunakan perangkat keras dan lunak untuk membantu proses pengerjaannya. Untuk spesifikasi perangkat keras yang digunakan dapat dilihat pada Tabel 5.1dan Tabel 5.2.

Tabel 5.1 Spesifikasi Perangkat Keras Laptop

Nama Perangkat	Laptop
Processor	Intel® Core™ i5-7200 CPU @ 2.7GHz
Memory	8192 MB RAM
Sistem Operasi	Windows 10
Arsitektur Sistem	64-bit *Operating System, x64-based processor*

Tabel 5.2 Spesifikasi Perangkat Keras Komputer

Nama Perangkat	Komputer
Processor	Intel® Core™ i5-8400 CPU @ 2.8GHz
Memory	MB RAM
Graphic Card	Nvidia GTX 1070 8Gb
Sistem Operasi	Ubuntu 16.01
Arsitektur Sistem	64-bit *Operating System, x64-based processor*

Kemudian daftar perangkat lunak yang digunakan dalam penelitian ini dapat dilihat pada Tabel 5.3.

Tabel 5.3 Daftar Perangkat Lunak

Bahasa Pemrograman	PHP 5.6, Python 3.6, HTML, CSS, Javascript
Basis Data	MySQL
Editor	Visual Studio Code, Jupyter Notebook
Virtual Environment	Anaconda
Framework	CodeIgniter 3.1.6
Library	a. Tweepy b. Pandas c. Numpy d. Json e. Gensim f. LangDetect g. MySQL JDBC Connector h. Matplotlib i. Jsonpickle j. Sklearn

5.2. Pembuatan *Crawler*

Crawler berfungsi untuk melakukan pengambilan dan penyimpanan data yang diperoleh dari media sosial *twitter*. *Crawler* dalam penelitian ini menggunakan *library tweepy* dengan basis bahasa pemrograman Python. Bagian pertama pada kode program *crawler* adalah pengaturan pada otentikasi untuk mendapatkan akses API *twiter*. Otentikasi yang dilakukan oleh *crawler* membutuhkan API_KEY dan API_SECRET yang diperoleh dari *twitter apps*.

```
1. import tweepy
2.
3. auth = tweepy.AppAuthHandler('7TMHiqfBxC3uF8OEfV1N
   LIIPu','94w0jA6CHZe9WBm45FiHaQ1QPTdWJRBkTdzubEvezP
   ajUencJJ')
```

```
4.  api = tweepy.API(auth, wait_on_rate_limit=True,
    wait_on_rate_limit_notify=True)
```

Kode 5.1 Otentikasi API *Twitter*

Kode 5.1 mendeklarasikan variabel *auth* untuk menyimpan nilai API_KEY dan API_SECRET *twitter apps*. Kemudian objek baru *api* akan dibuat dengan memanfaatkan *function* API dari kelas *tweepy*. Setelah mendapatkan otentikasi API Twitter, program sudah mendapatkan akses untuk melakukan pengambilan data menggunakan API *twitter*.

Pengambilan data *twitter* yang dilakukan pada penelitian ini menggunakan metode *search* karena memiliki kelebihan yaitu lebih cepat dalam pengambilan data dalam jumlah besar dibandingkan dengan metode *stream*. Metode *search* pada *twitter* API akan mengambil data *tweet* hingga 10 hari kebelakang.

```
1.  searchQuery = input('keyword ')  # this is what we
    're searching for
2.  maxTweets = 50 # Some arbitrary large number
3.  tweetsPerQry = 100  # this is the max the API perm
    its
4.  fName = searchQuery+".txt" # We'll store the tweet
    s in a text file.
5.  sinceId = None
6.  max_id = 1008898212826976257
```

Kode 5.2 Variabel Crawler

Kode 5.2 mendeklarasikan beberapa variabel, terutama kata kunci yang digunakan sebagai acuan untuk pengambilan data *tweet*. *Crawler* juga mendeklarasikan variabel *max_id* yang berisi nilai dari *id tweet*. *Id tweet* tersebut harus *id* dari *tweet* terbaru pada saat *crawler* dijalankan. Hasil pengambilan data akan disimpan pada file dengan nama pada variabel *fName*.

```
1.  if (max_id <= 0):
2.    if (not sinceId):
3.      new_tweets = api.search(q=searchQuery, count=t
    weetsPerQry, tweet_mode='extended')
4.
5.    else:
6.      new_tweets = api.search(q=searchQuery, count=t
    weetsPerQry,
    since_id=sinceId, tweet_mode='extended')
```

Kode 5.3 Pengambilan *Tweet* Jika *Max_id* Memiliki Nilai 0

Pada Kode 5.3 program akan melakukan pengambilan data jika variabel *max_id* memiliki nilai kurang dari atau sama dengan 0. Hal tersebut dilakukan jika selama proses perulangan untuk pengambilan data, nilai *max_id* yang diperbarui memiliki nilai kurang dari atau sama dengan 0, sehingga *crawler* masih tetap terus berjalan. *Tweet* yang diambil akan dikembalikan dalam bentuk data JSON dan disimpan pada variabel *new_tweets*.

```
1.  else:
2.    if (not sinceId):
3.      new_tweets = api.search(q=searchQuery, count=t
    weetsPerQry, max_id=str(max_id - 1), tweet_mode='e
    xtended')
4.
5.    else:
6.     new_tweets = api.search(q=searchQuery, count=tw
    eetsPerQry, max_id=str(max_id - 1), since_id=since
    Id, tweet_mode='extended')
```

Kode 5.4 Pengambilan *Tweet* Jika *Max_id* Tidak Sama dengan 0

Jika nilai *max_id* tidak kurang dari atau sama dengan 0, *crawler* juga akan memulai untuk melakukan pengambilan data. Karena *twitter* API memiliki *limit* untuk jumlah data *tweet* yang diperbolehkan untuk diambil, maka kode program tersebut

menggunakan logika perulangan dengan selalu melakukan pembaruan pada variabel *max_id*, sehingga *limit* pengambilan data pada *twitter* API tidak akan membuat program untuk berhenti melakukan *crawling*. Dalam 1 perulangan, data yang diambil dapat mencapai 100 *tweet*. *Crawler* hanya akan berhenti jika sudah tidak ada *tweet* yang mengandung kata kunci. Kumpulan *tweet* hasil *crawling* akan disimpan ke dalam file JSON.

Seluruh file JSON kumpulan data *twitter* akan disimpan ke dalam basis data dengan menggunakan program. Program yang dibuat berfungsi untuk membaca seluruh file JSON, kemudian diubah ke dalam bentuk *query* SQL.

```
1.  with open(filename,'r',encoding='utf-
    8') as data:
2.    for line in data:
3.      tweet = json.loads(line)
4.
5.      if 'retweeted_status' in tweet:
6.        created_at = tweet['created_at']
7.        id = tweet['id']
8.        text = tweet['retweeted_status']['full_text'
    ]
9.        user = tweet['user']['name']
10.       location = tweet['user']['location']
11.
12.     elif 'retweeted_status' not in tweet:
13.       created_at = tweet['created_at']
14.       id = tweet['id']
15.       text = tweet['full_text']
16.       user = tweet['user']['name']
17.       location = tweet['user']['location']
```

Kode 5.5 Membaca File JSON Kumpulan Data *Twitter*

Pada Kode 5.5 program berfungsi untuk membaca file JSON luaran dari proses *crawling twitter* sebelumnya. Setiap *line* data *twitter* yang dibaca akan disimpan pada variabel *tweet*. Terdapat 2 tipe data *twitter* berdasarkan objek *retweeted_status*. Untuk data *twitter* yang merupakan *retweet*, maka teks yang diambil

adalah teks asli yang di-*retweet*. Sedangkan untuk data *twitter* yang bukan *retweet* akan langsung diambil teksnya. Setelah nilai dari objek JSON selesai diambil, nilai tersebut akan diubah ke dalam bentuk *query*.

```
values = id, topic, created_at, text, user, location
query = 'INSERT INTO tweet VALUES {};'.format(str(values))

open file .sql
output = open("insert_tweet_to_database.sql","a",encoding="utf-8")
output.write(query+"\n")
output.close()
```

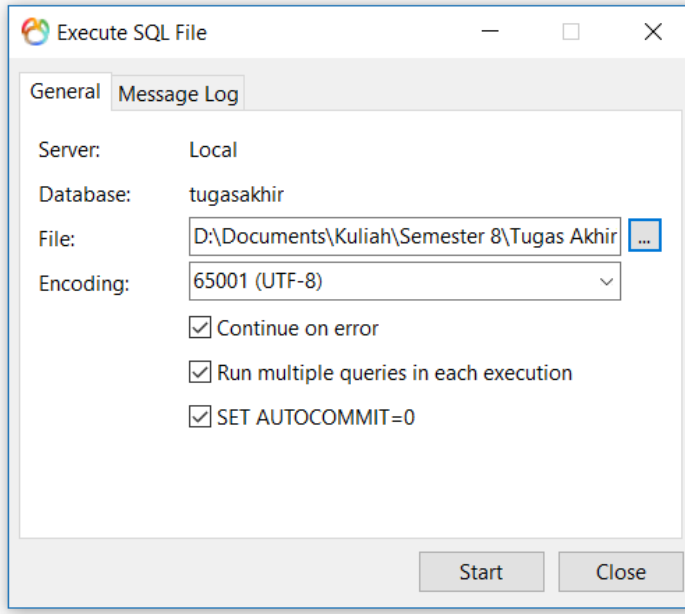
Kode 5.6 Menyimpan Data dengan Bentuk *Query*

Query yang dihasilkan dari proses pengubahan data JSON akan disimpan pada file dengan ekstensi SQL.

5.3. Pra-pemrosesan Data

5.3.1. Penggabungan Kumpulan Data

Penggabungan kumpulan data *twitter* dilakukan dengan menggunakan aplikasi Navicat. Kumpulan data *twitter* dari penelitian lain dengan studi kasus *e-commerce* dan telekomunikasi akan diubah terlebih dahulu ke dalam bentuk *query* SQL dengan menyesuaikan desain basis data yang akan digunakan untuk menyimpan seluruh kumpulan data *twitter*. seluruh file *query* SQL yang dihasilkan akan diimpor ke basis data menggunakan aplikasi Navicat.



Gambar 5.1 Impor *Query* SQL Menggunakan Navicat

5.3.2. Penghapusan Kumpulan Data Terduplikasi

Penghapusan dilakukan pada kumpulan data *twitter* yang memiliki konten teks yang sama persis. Hal ini dilakukan dengan menggunakan aplikasi Navicat dan memanfaatkan fungsi *query* SQL.

```
1.  CREATE TABLE tweet_temp LIKE tweet;
2.
3.  INSERT INTO tweet_temp
4.  SELECT * FROM tweet GROUP BY text
5.  ORDER BY topic;
6.
7.  DROP TABLE tweet;
8.  ALTER TABLE tweet_temp RENAME TO tweet
```

Kode 5.7 *Query* Menghapus Data Terduplikasi

Pada Kode 5.7 bagian pertama merupakan *query* pembuatan tabel baru dengan struktur atribut yang sama dengan tabel *tweet*. Hal tersebut berfungsi untuk menyimpan data baru yang telah bersih dari duplikasi. Kemudian pada bagian kedua, data pada tabel *tweet* akan dilakukan penghapusan duplikasi dengan menyeleksi data yang memiliki konten teks yang sama persis dan dimasukkan ke dalam tabel baru *tweet_temp*. Yang terakhir adalah menghapus tabel lama *tweet* dan mengubah nama tabel *tweet_temp* menjadi tabel *tweet*.

5.3.3. *Filtering* Bahasa Indonesia

Kumpulan data *tweet* yang telah bersih dari duplikasi akan dilakukan pemrosesan kembali untuk menghilangkan data yang tidak mengandung bahasa Indonesia. Kode program yang dibuat untuk melakukan *filtering* bahasa Indonesia menggunakan *library LangDetect*. *Library* tersebut memiliki kemampuan untuk mendeteksi jenis bahasa yang terkandung di dalam teks.

```
1.  import mysql.connector
2.
3.  db = mysql.connector.connect(host="localhost", use
    r="root", passwd="podvertible19960504", db="tugasa
    khir")
4.  cur = db.cursor()
5.  cur.execute("SELECT * FROM tweet")
6.
7.  result = cur.fetchall()
```

Kode 5.8 Membuat Koneksi Dengan *Database*

Kode 5.8 menujukan *syntax* pembuatan koneksi pada basis data dengan menggunakan *library* MySQL JDBC *connector*. *Query* akan dieksekusi menggunakan fungsi *cursor*. Luaran dari *query* yang dieksekusi akan disimpan pada variabel *result*.

```
from langdetect import detect

for row in result:
  try:
    # Detect Language
    if detect(row[3]) == 'id':
      text = row[3]
      output = write.writetoSQL(row[0], row[1], ro
w[2], text, row[4], row[5], 'Word_Embedd_1')
```

Kode 5.9 Deteksi Teks Berbahasa Indonesia

Pada Kode 5.9 luaran yang disimpan pada variabel *result* akan dilakukan perulangan per baris (*tweet*) untuk mendeteksi bahasa yang terkandung di dalam teks. Deteksi kandungan bahasa menggunakan fungsi *detect*. Nilai yang dikembalikan oleh fungsi *detect* adalah kode dari bahasa yang terkandung di dalam teks. Kode dari bahasa Indonesia sendiri adalah 'id'. Jika baris mengandung bahasa Indonesia maka akan di simpan dalam bentuk *query* SQL.

5.3.4. Anotasi Data

5.3.4.1. Pemilihan Data

Data yang akan diberi label diambil dari kumpulan data *twitter* hasil dari pra-pemrosesan sebelumnya. Data diambil secara acak berdasarkan topik yang telah ditentukan.

```
INSERT INTO tweet_tagging
SELECT * FROM tweet_non_duplicate WHERE topic='jok
owi' ORDER BY RAND() LIMIT 1500;
```

Kode 5.10 Pengambilan Data Twitter Secara Acak Berdasarkan Topik

Pada Kode 5.10 pengambilan data dilakukan berdasarkan pada topik, kemudian data diambil secara acak dengan batas maksimal 1500 data per topik. Daftar topik yang digunakan untuk anotasi data dapat dilihat pada Tabel 5.4.

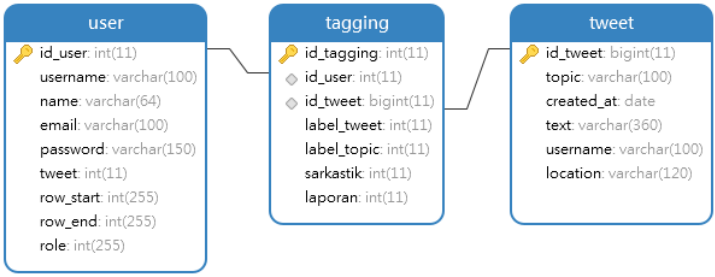
Tabel 5.4 Daftar Topik Anotasi Data

Topik / Kata Kunci	Jumlah *Tweet*
Reuni Akbar	1500
Jokowi	1500
Prabowo	1500
PDI Perjuangan	1500
Gerindra	1500
UU MD3	1500
Pilgub	1500

Pemilihan topik yang diambil untuk diberi label merupakan representasi dari isu politik, partai politik, dan tokoh politik.

5.3.4.2. Pembuatan Basis Data Anotasi

Langkah selanjutnya adalah membuat basis data untuk aplikasi anotasi. Basis data tersebut terdiri dari 3 tabel, yaitu *user*, *tweet*, dan *tagging*. Skema basis data yang dibuat dapat dilihat pada Gambar 5.2.

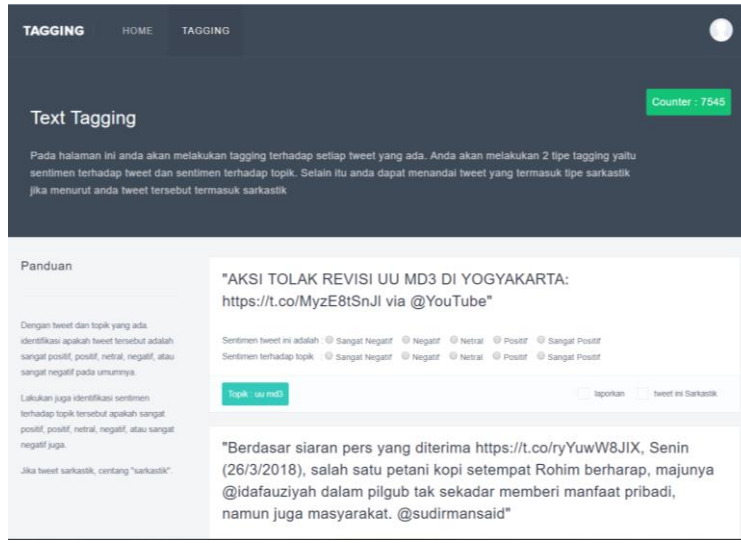


Gambar 5.2 Skema Basis Data Aplikasi Anotasi

Tabel *tweet* adalah tabel yang berisi kumpulan data *twitter* yang siap untuk diberi label hasil dari pemilihan data sebelumnya. Tabel *user* berisi identitas dan otentikasi anotator yang digunakan untuk *login* pada aplikasi. Hasil dari pemberian label data *twitter* oleh anotator akan disimpan pada tabel *tagging*.

5.3.4.3. Pembuatan Aplikasi Anotasi

Aplikasi berbasis web untuk anotasi data dibangun menggunakan *framework CodeIgniter*. *CodeIgniter* dipilih pada penelitian ini karena kemudahan dalam pengembangan dan pemeliharaannya. Aplikasi anotasi hanya memiliki 1 tampilan antar muka untuk pemberian label. Terdapat bantuan berupa panduan dalam pemberian label terhadap *tweet* yang ditampilkan untuk memudahkan anotator. Desain antar muka dari aplikasi dapat dilihat pada Gambar 5.3.



Gambar 5.3 Desain Antar Muka Aplikasi Anotasi

Bagian *card* pada aplikasi akan memuat *tweet* dan topik yang akan diberi label. terdapat 2 isian yang wajib di isi oleh anotator yaitu label *tweet* dan label terhadap topik. Untuk isian sarkasme bersifat opsional.

Gambar 5.4 Bagian *Card* pada Aplikasi Anotasi

Aplikasi dan basis data anotasi akan diunggah ke VPS untuk dapat diakses secara *online* oleh anotator.

5.3.4.4. Skenario Anotasi Data

Penelitian ini membutuhkan sekitar 15 anotator dengan masing-masing melakukan pemberian label pada 2000 data *twitter*. Setiap data *twitter* akan diberi label oleh 3 anotator untuk memperoleh label sentimen yang lebih objektif. Pembagian data *tweet* yang di anotasi oleh anotator dapat dilihat pada Tabel 5.5.

Tabel 5.5 Skenario Anotasi Data

Anotator	Indeks *tweet*
Anotator 1,2,3	0 – 2.000
Anotator 4,5,6	2.001 – 4.000
Anotator 7,8,9	4.001 – 6.000
Anotator 10,11,12	6.001 – 8.000
Anotator 13,14,15	8.001 – 10.000

5.3.4.5. Konsolidasi Hasil Anotasi Data

Hasil dari pemberian label oleh anotator akan dilakukan konsolidasi untuk mendapatkan label akhir dari data *twitter*. Sesuai dengan metodologi yang digunakan pada anotasi data, konsolidasi data dilakukan dengan skenario seperti berikut:

a. Jika 2 dari 3 orang yang melakukan pelabelan menyetujui label yang sama, maka label tersebut akan diterima secara langsung.
b. Jika tidak, nilai representasi dari seluruh label akan dihitung nilai rata-ratanya lalu dibulatkan ke nilai integer terdekat. Untuk menyeimbangkan kecenderungan rata-rata untuk menjauh dari -2 dan 2, dan juga tidak memilih 0. Maka pembulatan dilakukan pada ± 0.4 dan ± 1.4.

5.3.5. Penghapusan Tanda Baca dan Simbol

Kumpulan data *twitter* hasil dari pra-pemrosesan *filtering* bahasa Indonesia diekspor ke dalam bentuk file JSON. Kemudian file JSON akan dibaca oleh kode program dengan menggunakan *library* Pandas untuk diubah tipe datanya menjadi *dataframe*.

```
1.  import pandas as pd
2.  import json
3.  from pandas.io.json import json_normalize
4.
5.  with open('tweet.json','r',encoding='utf-
    8') as json_data:
6.      tweet = json.load(json_data)
7.
8.  raw_df = pd.io.json.json_normalize(data=tweet, rec
    ord_path=['RECORDS'])
```

Kode 5.11 Mengubah Tipe Data Menjadi *Dataframe*

Kumpulan data *twitter* dalam bentuk *dataframe* kemudian diubah kembali ke dalam bentuk *array* untuk dilakukan proses penghapusan tanda baca dan simbol yang terkandung di dalam data teksnya.

```
1.  import re
2.  import emoji
3.
4.  # Senyum
```

```
5.  string = string.replace(':))', ' <senyum_senyum> '
    )
6.  string = string.replace(':)', ' <senyum> ')
7.  string = string.replace(':-
    ))', ' <senyum_senyum> ')
8.  string = string.replace(':-)', ' <senyum> ')
9.  string = string.replace('((:', ' <senyum_senyum> '
    )
10. string = string.replace('(:', ' <senyum> ')
11. string = string.replace('((-
    :', ' <senyum_senyum> ')
12. string = string.replace('(-:', ' <senyum> ')
13. string = string.replace('=))', ' <senyum_senyum> '
    )
14. string = string.replace('=)', ' <senyum> ')
15. string = string.replace('^_^', ' <senyum> ')
16.
17. # Sedih
18. string = string.replace(':((', ' <sedih_sedih> ')

19. string = string.replace(':(', ' <sedih> ')
20. string = string.replace(':-
    ((', ' <sedih_sedih> ')
21. string = string.replace(':-(', ' <sedih> ')
22. string = string.replace(')):', ' <sedih_sedih> ')

23. string = string.replace('):', ' <sedih> ')
24. string = string.replace('))-
    :', ' <sedih_sedih> ')
25. string = string.replace(')-:', ' <sedih> ')
26.
27. # Berkedip
28. string = string.replace(';))', ' <senyum_berkedip>
     ')
29. string = string.replace(';)', ' <senyum_berkedip>
    ')
30.
31. # Tears
32. string = string.replace(":'))", ' <menangis_bahagi
    a> ')
33. string = string.replace(":')", ' <menangis_bahagia
    > ')
34. string = string.replace(":'((", ' <menangis_sedih>
     ')
```

```
35. string = string.replace(":'(", ' <menangis_sedih>
    ')
36. string = string.replace("((':", ' <menangis_bahagi
    a> ')
37. string = string.replace("(':", ' <menangis_bahagia
    > ')
38.
39. # Some annoyed
40. string = string.replace(':/', ' <terganggu> ')
41. string = string.replace(':\\', ' <terganggu> ')
42.
43. # Straight face
44. string = string.replace(':|', ' <muka_datar> ')
45. string = string.replace(':-|', ' <muka_datar> ')
```

Kode 5.12 Penggantian Simbol *Emoticon*

Kode 5.12 merupakan potongan kode program di dalam fungsi *cleansing* yang berfungsi untuk menerjemahkan ekspresi *emoticon* ke dalam bentuk teks. Fungsi tersebut menggunakan *library regular expression* untuk mendeteksi pola karakter *emoticon* pada teks dan *library emoji* untuk mengubah *unicode emoji* ke dalam bentuk teks.

Tabel 5.6 Daftar *Mapping Emoticon*

Emoticon	Teks	Emoticon	Teks
<3	Hati	:((	Sedih
:D	Tertawa	:(	Sedih
:P	Menjulurkan Lidah	:-((	Sedih
:O	Terkejut	:-(	Sedih
:X	Cium	(:	Sedih
:*	Cium	((:	Sedih
:3	Malu Malu Kucing	;)	Sedih

XD	Tertawa	:'	Menangis
:))	Senyum	:'))	Menangis
:)	Senyum	:')	Menangis
:-))	Senyum	:'((	Menangis
:-)	Senyum	:'(	Menangis
(:	Senyum	(':	Menangis
((:	Senyum	:/	Terganggu
^_^	Senyum	:|	Muka

Daftar *mapping emoticon* ke dalam bentuk teks dapat dilihat pada Tabel 5.6. Penggantian *emoticon* menggunakan bahasa Indonesia bertujuan untuk mendapatkan makna dari *emoticon* tersebut.

```
1.  text = text = text.replace("\n"," ")
2.  text = re.sub(r"\s–\s", "<url>", text)
3.  text = re.sub(r"http\S+", "<url>", text)
4.  text = re.sub(r'(.)\1{2,}', r'\1\1', text)
5.  text = re.sub(r'\.{2,}', ' <ellipsis> ', text)

6.  text = re.sub(r"\W@\w+", "<mention>", text)
7.  text = re.sub(r"xa0"," ", text)
8.  text = re.sub(r"amp;"," ", text)
9.  text = re.sub(r"quot;"," ",text)
10. text = re.sub(r"\b'\b","", text)
11. text = re.sub(r"\.", " . ", text)
12. text = re.sub(r",", " , ", text)
13. text = re.sub(r":", " : ", text)
14. text = re.sub(r";", " ; ", text)
15. text = re.sub(r"!", " ! ", text)
16. text = re.sub(r"\?", " ? ", text)
17. text = re.sub(r"\(", " ( ", text)
18. text = re.sub(r"\)", " ) ", text)
19. text = re.sub(r"#", " <hash_tag> ", text)
20. text = re.sub(r"\[", " [ ", text)
21. text = re.sub(r"\]", " ] ", text)
22. text = re.sub(r"\'m ", " \'m ", text)
23. text = re.sub(r"\'s ", " \'s ", text)
24. text = re.sub(r"\'re ", " \'re ", text)
25. text = re.sub(r"\'ll ", " \'ll ", text)
```

```
26. text = re.sub(r"\'d ", " \'d ", text)
27. text = re.sub(r"\'ve ", " \'ve ", text)
28. text = re.sub(r"n\'t ", " n\'t ", text)
29. text = re.sub(r"[^A-Za-z0-
    9().,<_>!?\'`]", " ", text)
30. text = re.sub(r"\s{2,}", " ", text)
```

Kode 5.13 Penghapusan Tanda Baca dan Simbol

Kode 5.13 merupakan potongan kode program selanjutnya yang berfungsi untuk mengganti seluruh tanda baca dan simbol yang tidak memiliki makna di dalam teks. Penghapusan yang dilakukan menggunakan *regular expression* dengan fungsi *sub* yang berfungsi untuk melakukan substitusi karakter. Penggantian hanya dilakukan pada simbol *mention*, *hashtag*, *url*, dan *ellipsis*. Simbol-simbol tersebut diubah ke dalam bentuk *tag* <simbol>.

5.3.6. *Tokenizing* dan *Casefolding*

tokenizing dilakukan pada kumpulan data yang akan digunakan untuk *training* model *word embeddings*. sedangkan *casefolding* dilakukan untuk semua kumpulan data untuk *training word embedding* dan CNN.

```
1.  text = text.lower() # lowercase text
2.  return text.strip().split()
```

Kode 5.14 *Tokenizing* dan *Casefolding*

Kode 5.14 merupakan potongan kode program yang berfungsi untuk melakukan *casefolding* dan *tokenizing*. *Casefolding* menggunakan fungsi *lower* pada Python. Sedangkan untuk melakukan *tokenizing* menggunakan fungsi *split*.

5.4. Pembuatan Model Word2Vec

Model *word embeddings* akan dihasilkan dari proses *training* data dengan menggunakan algoritma *word2vec*. Data yang digunakan untuk *training* model merupakan gabungan kumpulan data dari seluruh penelitian dengan studi kasus *e-commerce* dan telekomunikasi yang sudah dilakukan pra-pemrosesan pada tahap sebelumnya. Penelitian ini akan membuat 2 model *word2vec* dengan perbedaan pada parameter *learning algortihm*, yaitu *skip-gram* dan CBOW. Skenario pembuatan model dapat dilihat pada Tabel 5.7 dan Tabel 5.8.

Tabel 5.7 Konfigurasi Model *Word2vec* 1

Parameter	Nilai
Learning Algortithm	*Skip-gram*
Iteration	10
Minimum Word Frequency	3
Context Window	5
Layer Size / Vector Dimention	300

Tabel 5.8 Konfigurasi Model *Word2vec* 2

Parameter	Nilai
Learning Algortithm	CBOW
Iteration	10
Minimum Word Frequency	3
Context Window	5
Layer Size / Vector Dimention	300

Training model *word embeddings* akan dilakukan dengan menggunakan kode program Python yang memanfaatkan *library* gensim.

```
1.  import multiprocessing
2.  from gensim.models import Word2Vec
```

```
3.
4.  # Dimentionality of the resulting words vector
5.  num_features = 300
6.  # Minimum word count treshold
7.  min_word_count = 3
8.  # Number of thread to run parallel
9.  num_workers =  multiprocessing.cpu_count()
10. # Context windows length
11. context_size = 5
12. # Seed for RNG, to make the result reproducible
13. seed = 1
14. # Iteration
15. iteration = 10
```

Kode 5.15 Inisiasi Variabel Model *Word2vec*

Error! Reference source not found. merupakan potongan kode program yang melakukan inisiasi variabel. Variabel berisi nilai konfigurasi parameter model *word2vec* yang akan dibuat. Nilai dari konfigurasi parameter yang digunakan sesuai dengan skenario *training* model *word2vec* yang telah dibuat sebelumnya. Variabel *num_features* merupakan dimensi vektor kata yang dihasilkan, *min_word_count* adalah jumlah minimal kemunculan kata yang digunakan *training*, *context_size* adalah jumlah kata sebelum dan sesudah, dan *iteration* merupakan jumlah iterasi *training* model.

```
word2vec_model = Word2Vec(
    sg = 1, # 1=skipgram, 0=CBOW
    seed = seed,
    workers = num_workers,
    size = num_features,
    min_count = min_word_count,
    window = context_size,
    iter = iteration
)
```

Kode 5.16 Inisiasi Model *Word2vec*

Kode 5.16 merupakan inisiasi model *word2vec* dengan beberapa konfigurasi parameter yang berasal dari variabel sebelumnya. Parameter yang menentukan jenis *learning algorithm* adalah variabel *sg*, nilai 1 untuk *learning algorithm skip-gram*, dan 2 untuk CBOW. *Training word2vec* akan memanfaatkan seluruh *resource* CPU.

```
word2vec_model.build_vocab(sentences=sentences)
word2vec_model.train(sentences, total_examples = t
oken_count, epochs = 10)
word2vec_model.save("Word2Vec_trained_1.bin")
```

Kode 5.17 *Training Word2vec*

Sebelum memulai *training* model, *gensim* akan membuat *corpus* kosakata yang berisi daftar seluruh kata dengan indeksnya. Pembuatan *corpus* kosakata menggunakan fungsi *build_vocab*. Setelah *corpus* kosakata terbentuk, kemudian *training* akan berjalan menggunakan fungsi *train* sesuai dengan parameter yang telah ditentukan sebelumnya. Model *word2vec* hasil *training* akan disimpan ke dalam bentuk file BIN. *Training* model *word2vec* dilakukan 2 kali hanya dengan mengganti parameter *sg*.

5.5. Pembuatan Model CNN

Kumpulan data yang digunakan untuk *training* model CNN adalah data *twitter* berlabel luaran tahap anotasi data. Untuk menyiapkan data pembuatan model, kumpulan data *tweet* berlabel pada basis data aplikasi anotasi, diekspor ke dalam bentuk file JSON. Sampel data dalam file JSON dapat dilihat pada Kode 5.18.

```
{
    "RECORDS": [
        {
            "id_tweet": "936833665287340034",
            "label": "1",
            "label_topic": "1",
            "sarkastik": "0",
            "tweet": "Kau sebut mereka RADIKAL\n\n
Kalau Kami menyebutnya BIDADARI !\n\n( tapi gada s
ayapnya ? )\n\nRT         = Setuju \nRT&amp;♡ = S
etuju pake BANGET !\n\n#ReuniAkbar212"
        }
```

Kode 5.18 Kumpulan data *Training* Model CNN

Kode program yang dibuat untuk melakukan *training* model CNN menggunakan *library* Pytorch. *Library* tersebut akan memanfaatkan kemampuan komputasi arsitektur *cuda* pada GPU untuk melakukan komputasi secara pararel. Tahapan dalam *training* model CNN meliputi mempersiapkan data *training*, memuat model *word embeddings*, pembagian data *training* dan *testing*, *training* model, dan evaluasi performa model yang dihasilkan.

5.5.1. Persiapan Data *Training*

Pada tahap ini kumpulan data *twitter* akan dibaca dan dibersihkan. Proses pembersihan data yang dilakukan sama dengan tahap pra-pemrosesan data pada sub bab 5.3.5 dan 5.3.6.

```
1.  in_data = prepare_data.DataPreparation()
2.  in_data.read_dataset(args.subtask, 'json')
```

Kode 5.19 Potongan Kode Program pada Kelas *Main* yang Menginisiasi Objek dari Kelas *Data Preparation*

Persiapan data *training* diawali dengan inisiasi objek *in_data* dari kelas *Data Preparation*. Hal tersebut dijelaskan pada Kode 5.19. *Method read_dataset* akan dieksekusi untuk membuat data *training* model CNN dengan menggunakan *library torch text*,

```
1.  def read_dataset(self, substask, type_file):
2.
3.      # Read in data
4.      file_name = 'dataset/sentimen_own_full_repaire
    d.json'
5.
6.      # politic_data is an iterator object returning
     the id, label, and text of data
7.      politic_data = read_sentences.ReadingIndonesia
    nPolitic(file_name, type_file)
8.
9.      # Create instructions for processing the text
10.     id_field = torchtext.data.Field(use_vocab=Fals
    e, sequential=False)
11.     label_field = torchtext.data.Field(sequential=
    False)
12.     text_field = torchtext.data.Field()
13.
14.     fields = [('twt_id',id_field),
15.               ('label',label_field),
16.               ('text', text_field)]
17.
18.     self.fields = fields
19.
20.     # Populate the list of training and test examp
    les
21.     if substask == 'subtaskA':
22.       examples = [torchtext.data.Example.fromlist(
    [twt_id, self.polarity_to_label_substask_a(polarit
    y), text], fields)for twt_id, polarity, text in po
    litic_data]
```

```
23.     elif substask == 'subtaskB':
24.       examples = [torchtext.data.Example.fromlist(
    [twt_id, self.polarity_to_label_substask_b(polarit
    y), text], fields)for twt_id, polarity, text in po
    litic_data]
25.     elif substask == 'subtaskC':
26.       examples = [torchtext.data.Example.fromlist(
    [twt_id, self.polarity_to_label_substask_c(polarit
    y), text], fields)for twt_id, polarity, text in po
    litic_data]
27.
28.     self.examples = examples
```

Kode 5.20 ***Method Read Dataset***

Data akan dibaca berdasarkan *directory path* dari file kumpulan data *twitter* berlabel. Nama file yang tersimpan pada variabel *file_name* akan di-*parsing* pada kelas *ReadingIndonesianPolitic* untuk dilakukan proses pembacaan dan pembersihan data.

```
1.  def __iter__(self):
2.
3.    if self.tipe == 'json':
4.      data = self.read_json()
5.
6.    # Start Processing
7.    for line in data:
8.      if self.tipe =='json':
9.        twt_id, polarity, text = self.divide_line(li
    ne)
10.     elif self.tipe =='txt':
11.       twt_id, polarity, text = self.divide_line_te
    xt(line)
12.
13.     text = self.replace_URL(text)
14.     text = html.unescape(text)
15.     text = text.replace('""'"", ' <kutip> ') # t
    ext.replace('"', ' <kutip> ')
16.     text = self.replace_mention(text)
17.     text = self.replace_mult_occurences(text)
18.     text = text.replace('..', ' <elipsis> ')
19.     text = self.replace_emoticons(text)
```

```
20.     text = self.clean_str(text)
21.     text = text.lower()
22.
23.     yield twt_id, polarity, text
```

Kode 5.21 Kelas *ReadIndonesianPoltic*

Kelas *ReadIndonesianPolitic* merupakan *subclass* dari kelas *Read Sentences* yang berisi proses pembersihan data. File JSON kumpulan data *twitter* berlabel akan dibaca per *tweet* dan dibersihkan menggunakan *method* pada kelas *Read Sentences*.

```
1.  def divide_line(self, line):
2.    twt_id = line[0]
3.    polarity = line[1]
4.    text = line[4]
5.
6.    return twt_id, polarity, text
```

Kode 5.22 *Method Divide Line* pada Kelas *Read Sentences*

Data yang dibaca akan dibagi berdasarkan kolom pada setiap baris *tweet*. Dari kumpulan data *twitter* berlabel hanya data *id*, label sentimen, dan teksnya yang digunakan untuk *training*.

```
1.  def replace_URL(self, string):
2.    tokens  = ['<url>' if '://' in token else token

3.               for token in string.split()]
4.    return ' '.join(tokens)
5.
6.  def replace_mention(self, string):
7.    tokens  = ['<mention>' if token.startswith('@')
    else token
8.                 for token in string.split()]
9.    return ' '.join(tokens)
10.
11. def replace_mult_occurences(self, string):
12.   return re.sub(r'(.)\1{2,}', r'\1\1', string)
13.
```

```
14. def replace_token_emoticon(self, token):
15.   if token.startswith('<3'):
16.     return ' <hati> '
17.   if token.startswith(':'):
18.     if token == ':D':
19.       return ' <tertawa> '
20.     if (token == ':P') | (token == ':p'):
21.       return ' <menjulurkan_lidah> '
22.     if (token == ':O') | (token == ':o'):
23.       return ' <terkejut> '
24.     if (token == ':x') | (token == ':*'):
25.       return ' <cium> '
26.     if token == ':3':
27.       return ' <malu-malu_kucing> '
28.   if token.startswith('='):
29.     if (token == '=/') | (token == '=\\'):
30.       return ' <terganggu> '
31.   if token == 'XD':
32.     return ' <tertawa_terbahak-bahak> '
33.       return token
34.
35. def replace_emoticons(self, string):
36.   # Senyum
37.   string = string.replace(':))', ' <senyum_senyum>
    ')
38.   string = string.replace(':)', ' <senyum> ')
39.   string = string.replace(':-
    ))', ' <senyum_senyum> ')
40.   string = string.replace(':-)', ' <senyum> ')
41.   string = string.replace('((:', ' <senyum_senyum>
    ')
42.   string = string.replace('(:', ' <senyum> ')
43.   string = string.replace('((-
    :', ' <senyum_senyum> ')
44.   string = string.replace('(-:', ' <senyum> ')
45.   string = string.replace('=))', ' <senyum_senyum>
    ')
46.   string = string.replace('=)', ' <senyum> ')
47.   string = string.replace('^_^', ' <senyum> ')
48.
49.   # Sedih
50.   string = string.replace(':((', ' <sedih_sedih> '
    )
51.   string = string.replace(':(', ' <sedih> ')
```

```
52.   string = string.replace(':-
    ((', ' <sedih_sedih> ')
53.   string = string.replace(':-(', ' <sedih> ')
54.   string = string.replace(')):', ' <sedih_sedih> '
    )
55.   string = string.replace('):', ' <sedih> ')
56.   string = string.replace('))-
    :', ' <sedih_sedih> ')
57.   string = string.replace(')-:', ' <sedih> ')
58.
59.   # Berkedip
60.   string = string.replace(';))', ' <senyum_berkedi
    p> ')
61.   string = string.replace(';)', ' <senyum_berkedip
    > ')
62.
63.   # Tears
64.   string = string.replace(":'))", ' <menangis_baha
    gia> ')
65.   string = string.replace(":')", ' <menangis_bahag
    ia> ')
66.   string = string.replace(":'((", ' <menangis_sedi
    h> ')
67.   string = string.replace(":'(", ' <menangis_sedih
    > ')
68.   string = string.replace("((':", ' <menangis_baha
    gia> ')
69.   string = string.replace("(':", ' <menangis_bahag
    ia> ')
70.
71.   # Some annoyed
72.   string = string.replace(':/', ' <terganggu> ')
73.   string = string.replace(':\\', ' <terganggu> ')

74.
75.   # Straight face
76.   string = string.replace(':|', ' <muka_datar> ')

77.   string = string.replace(':-
    |', ' <muka_datar> ')
78.
79.   string = ' '.join([self.replace_token_emoticon(t
    oken)
80.                      for token in string.split()]
    )
```

```
81.   return string
```

Kode 5.23 ***Method Replace URL, Mention, Mult Occurences, Token Emoticon*** **dan** ***Emoticons***

Setelah data dibagi menjadi beberapa *field*, data teksnya akan dibersihkan terlebih dahulu dari *url* yang kemudian diganti dengan *tag* <url>. Setelah itu data teks melalui proses pembersihan dari simbol @ yang kemudian diganti dengan <mention>. Pada *method* selanjutnya data teks yang memiliki pola karakter *emoticon* akan diubah artinya ke dalam bentuk teks. Data *twitter* yang sudah dibersihkan akan dikembalikan dalam bentuk 3 *field* yaitu *id tweet*, *label*, dan teks.

Objek baru dibuat menggunakan *libary torchtext* untuk menyimpan data yang siap digunakan untuk *training* model CNN. Kumpulan data yang dibuat akan menyesuaikan dengan *subtask* yang digunakan. Polaritas label data akan dibuat pada *method polarity to label*.

```
1.  def polarity_to_label_substask_a(self, polarity):

2.    # Positive, Negative
3.    if int(polarity) < 0 :
4.      return 'Negative'
5.    elif int(polarity) > 0 :
6.      return 'Positive'
7.
8.  def polarity_to_label_substask_b(self, polarity):

9.    # Positive, Netral, Positive
10.   if int(polarity) < 0 :
11.     return 'Negative'
12.   elif int(polarity) > 0 :
13.     return 'Positive'
14.   elif int(polarity) == 0 :
15.     return 'Neutral'
16.
17. def polarity_to_label_substask_c(self, polarity):
```

```
18.   # Highly Negative, Negative, Neutral, Positive,
    Highly Positive
19.   if int(polarity) == -2 :
20.     return 'Highly_Negative'
21.   elif int(polarity) == -1 :
22.     return 'Negative'
23.   elif int(polarity) == 0 :
24.     return 'Neutral'
25.   elif int(polarity) == 1 :
26.     return 'Positive'
27.   elif int(polarity) == 2 :
28.     return 'Highly_Positive'
```

Kode 5.24 *Method Polarity to* Label Setiap *Subtask*

Seluruh angka pada *field* label akan diubah ke dalam bentuk teks. Jumlah label yang diubah sesuai dengan jenis *subtask* pada saat *training* model. *subtask* A berjumlah 2, B berjumlah 3, dan C berjumlah 5.

5.5.2. Memuat Model *Word Embeddings*

Model *word embeddings* dibutuhkan sebagai input *training* model CNN. Model tersebut akan dibaca dengan kelas *Embeddings* dan disimpan nilai vektornya sebagai objek.

```
1.  embeddings = read_embeddings.Embeddings()
2.  embeddings.read_model(args.embeddings_source)
```

Kode 5.25 Inisiasi Objek *Embeddings*

Model *word embeddings* yang dibaca tidak hanya model *word2vec* luaran dari tahap sebelumnya, namun juga model *word embeddings* dari bojanowski (Wikipedia) yang menggunakan algoritma *fastext* dalam *training* modelnya. Model *word embeddings* dimuat menggunakan *method read_model*.

```
1.  def read_model(self, tipe):
2.    if tipe == 'own-model2':
3.      print('Loading {} '.format(str(tipe)))
4.      embeddings_path = '/home/pras/Embeddings/model
    _arif'
5.      word2vec_model = KeyedVectors.load_word2vec_fo
    rmat(embeddings_path, binary=False, unicode_errors
    ="ignore")
6.    elif tipe == 'own-model3':
7.      print('Loading {} '.format(str(tipe)))
8.      embeddings_path = '/home/adrian/new_cnn/modela
    pik_cbows.bin'
9.      word2vec_model = KeyedVectors.load_word2vec_fo
    rmat(embeddings_path, binary=False, unicode_errors
    ="ignore")
10.   elif tipe == 'bojanowski':
11.     print('Loading {} model'.format(str(tipe)))
12.     embeddings_path = '/home/pras/Embeddings/wiki.
    bin'
13.     word2vec_model = FastText.load_fasttext_format
    (embeddings_path)
14.   else:
15.     print('Error Embeddings')
16.
17.   self.word2vec = word2vec_model
18.   self.embed_dim = 300
19.
20.   return word2vec_model
```

Kode 5.26 Membaca Model Word Embeddings

Pembacaan model *word embeddings* menggunakan *library gensim*. Ada 3 model yang digunakan, yaitu *word2vec skip-gram*, *word2vec* CBOW, dan *fasttext* bojanowski. *Method KeyedVetors.load* akan membaca vektor kata pada model *word embeddings* dan disimpan pada variabel *word2vec*.

5.5.3. *Training* Model CNN

Training model dimulai dengan menjalankan *method cross_validate*, di mana data *training* dan data *testing* akan dibagi. Perulangan dilakukan sejumlah 10 kali dengan kombinasi data *training* dan *testing* yang berbeda beda setiap perulangannya.

```
1.  cross_validate(args.fold_num, in_data, embeddings,
     args)
```

Kode 5.27 *Method Cross Validation* Dieksekusi

Input parameter dari *method cross validation* adalah jumlah *fold*, data *training*, vektor kata, dan beberapa variabel yang disimpan pada *args*.

```
1.  split_width = int(ceil(len(data.examples)/fold))
```

Kode 5.28 Menentukan Ukuran Data Setiap *Fold* pada *Method Cross_validate*

Ukuran data *training* pada setiap *fold* akan dihitung dan disimpan pada variabel *split_width*.

```
1.  train_examples = data.examples[:]
2.    del train_examples[i*split_width:min(len(data.ex
    amples), (i+1)*split_width)]
3.    test_examples = data.examples[i*split_widthmin(l
    en(data.examples), (i+1)*split_width)]
```

Kode 5.29 Pembagian Data *Training* dan Data *Testing* pada *Method Cross_validate*

Pada bagian awal, data *train_examples* mendefinisikan seluruh kumpulan data terlebih dahulu. Kemudian dihapus pada indeks data yang digunakan sebagai data *testing* dengan ukuran yang

telah ditentukan pada variabel *split_width*. Indeks data yang dihapus tersebut akan digunakan sebagai data *testing* yang disimpan pada variabel *test_examples*.

```
# Building the vocabs
  text_field  = None
  label_field = None

  for field_name, field_object in fields:
    if field_name == 'text':
      text_field = field_object
    elif field_name == 'label':
      label_field = field_object

  text_field.build_vocab(train_set)
  label_field.build_vocab(train_set)

  data.vocab_to_idx = dict(text_field.vocab.stoi)

  data.idx_to_vocab = {v: k for k, v in data.vocab
_to_idx.items()}

  data.label_to_idx = dict(label_field.vocab.stoi)

  data.idx_to_label = {v: k for k, v in data.label
_to_idx.items()}

  embed_num = len(text_field.vocab)
  label_num = len(label_field.vocab)
```

Kode 5.30 Membuat *Corpus* Kosakata pada *Method Cross_validate*

Sebuah *corpus* kosakata pada *field* teks akan dibuat dengan indeks kata-katanya. Kemudian *corpus* label dan indeksnya juga akan dibuat. Seluruh *corpus* yang telah dibuat disimpan pada *dictionary*.

```
# Loading pre-trained embeddings
  emb_init_values = np.array(data.create_folds_embeddings(embeddings, args))
```

Kode 5.31 Memuat Vektor Kata pada *Method Cross_validate*

Langkah selanjutnya adalah memuat vektor kata menggunakan *method create fold embeddings*. Nilai yang disimpan pada variabel *emb_init_values* adalah nilai *initial* vektor kata yang digunakan untuk *training*.

```
args.embeddings_dim = embeddings.embed_dim
for i in range(self.idx_to_vocab.__len__()): #untuk memastikan urut
word = self.idx_to_vocab.get(i)
if word == '<unk>':
  emb_init_values.append(np.random.uniform(-0.25, 0.25, args.embeddings_dim).astype('float32'))
elif word == '<pad>':
  emb_init_values.append(np.zeros(args.embeddings_dim).astype('float32'))
elif word in embeddings.word2vec.wv.vocab:
  emb_init_values.append(embeddings.word2vec.wv.word_vec(word))
else:
  emb_init_values.append(np.random.uniform(-0.25, 0.25, args.embeddings_dim).astype('float32'))
```

Kode 5.32 Potongan Kode Program *Method Create Fold Embeddings*

Kata yang tidak memiliki representasi vektor pada model *word embeddings* maka nilai vektornya akan diacak menggunakan *random uniform* -0.25 – 0.25. Kata yang memiliki representasi vektor pada model *word embeddings* akan diambil nilai vektor katanya dari model sesuai dengan dimensi vektor yang ditentukan.

```
1.   kim2014 = model.CNN(embed_num, label_num -
    1, args.embeddings_dim, args.embeddings_mode, emb_
    init_values, args)
2.
3.    if args.cuda:
4.      kim2014.cuda()
5.
6.    trained_model = train(kim2014, train_iter, test_
    iter, data.label_to_idx, data.idx_to_label, train_
    bulk_iter, args, i)
```

Kode 5.33 Inisiasi Model dan Mulai Training Model CNN

Model *neural network* CNN diinisiasi menggunakan kelas CNN. setelah itu model akan mulai melakukan *training* dengan menggunakan *method train*. *Training* akan dilakukan dengan menggunakan komputasi GPU jika pada parameter *cuda* adalah *True*.

```
1.  parameters = filter(lambda p: p.requires_grad, mod
    el.parameters())
2.
3.  # Optimizer
4.  optimizer = torch.optim.Adadelta(parameters) # Ada
    m, Adadelta
5.
6.  # using GPU
7.  if args.cuda:
8.      model.cuda()
9.
10. model.train()
11.
12. for epoch in range(1, args.epoch_num+1):
13.   print("FOLD/EPOCH [{}/{}]".format(fold+1, epoch)
    )
14.   step = 0
15.   corrects_sum = 0
16.
17.   for batch in train_iter:
18.     text_numerical, target = batch.text, batch.lab
    el
19.     if args.cuda:
```

```
20.       text_numerical, target = text_numerical.cuda
    (), target.cuda()
21.
22.     text_numerical.data.t_()
23.     target.data.sub_(1)
24.
25.     optimizer.zero_grad()
26.
27.     forward = model(text_numerical)
28.
29.     loss = F.cross_entropy(forward, target)
30.     loss.backward()
31.
32.     optimizer.step()
33.     step += 1
34.
35.     corrects = (torch.max(forward, 1)[1].view(targ
    et.size()).data == target.data).sum()
36.
37.     accuracy = 100.0 * corrects / batch.batch_size
```

Kode 5.34 *Method Train*

Training model diawali dengan inisiasi parameter *optimizer*. Pada penelitian ini *optimizer* yang digunakan adalah Adadelta. Perulangan *training* model dilakukan pada setiap *fold*, pada penelitian ini jumlah perulangan (*epoch*) dalam 1 *fold* adalah 10 *epoch*. Pada 1 *epoch*, data *training* akan dipecah kembali ke dalam *batch*. *Training* akan berjalan dengan menggunakan *method forward*.

```
1.  batch_width = input.size()[1]
2.  x = self.embeddings(input).view(-
    1, 1, self.embed_dim*batch_width)
3.  if self.embed_mode == 'multichannel' :
4.    x2 = self.embeddings2(input).view(-
    1, 1, self.embed_dim*batch_width)
5.    x = torch.cat((x, x2), 1)
6.
7.  conv_results = [
8.    F.max_pool1d(
```

```
9.      F.relu(self.convs[i](x)), batch_width - self.k
    ernel_width[i] + 1 )
10.     .view(-1, self.feature_num[i])
11.   for i in range(len(self.feature_num))
12. ]
13.
14. x = torch.cat(conv_results, 1)
15. x = F.dropout(x, p=self.dropout_rate, training=sel
    f.training)
16. x = self.linear(x)
17.
18. return x
```

Kode 5.35 ***Method Foward***

Method forward merupakan fungsi yang melakukan proses konvolusi hingga *backpropagation* pada model CNN. Diawali dengan membentuk matriks dimensi vektor dan data *batch*. Kemudian dilakukan konvolusi 1D. Hasil konvolusi dilakukan fungsi aktivasi menggunakan *relu*. Setelah dilakukan fungsi aktivasi hasilnya adalah berupa *feature maps*. *Pooling strategy* yang digunakan adalah 1-*max pooling*, yaitu nilai paling maksimal pada setiap *feature maps*. Seluruh hasil dari *pooling strategy* akan digabung dan selanjutnya dilakukan *back propagation* untuk menentukan prediksi label kelas.

5.5.4. Evaluasi dan Analisis Model CNN

Evaluasi *training* model CNN menggunakan perhitungan dari pengukuran *evaluation metrics* pada setiap *epoch* dan *fold training*.

```
1.  model.eval()
2.  corrects, avg_loss = 0, 0
3.
4.  data_iter.sort_key = lambda x: len(x.text)
5.
6.  for batch in data_iter:
7.
```

```
8.    text_numerical, target = batch.text, batch.label

9.
10.   if args.cuda:
11.     text_numerical, target = text_numerical.cuda()
    , target.cuda()
12.
13.   text_numerical.data.t_()
14.   target.data.sub_(1)
15.
16.   forward = model(text_numerical)
17.   loss = F.cross_entropy(forward, target, size_ave
    rage=False)
18.
19.   avg_loss += loss.data[0]
20.   corrects += (torch.max(forward, 1)[1].view(targe
    t.size()).data == target.data).sum()
21.
22. size = len(data_iter.dataset)
23. avg_loss = avg_loss/size
24. accuracy = 100.0 * corrects/size
25.
26. if tipe == 'testing':
27.   if args.cuda:
28.     data.append(accuracy.item())
29.   else:
30.     data.append(accuracy)
31.
32. return target.data, torch.max(forward, 1)[1].view(
    target.size()).data
```

Kode 5.36 ***Method Evaluate***

Kode 5.36 merupakan *method evaluate* yang berfungsi untuk mengembalikan nilai label aktual dan label yang diprediksi oleh model hasil *training*. Model *evaluate* akan melakukan prediksi terhadap data *training* dan *testing*. Kode program yang dilakukan untuk memprediksi label sama dengan *method train* hanya saja pada *method evaluate* model yang digunakan untuk memprediksi adalah model yang sudah dilakukan *training*. Prediksi label menggunakan fungsi *torch.max*.

Berdasarkan hasil *output* label prediksi yang dihasilkan, langkah selanjutnya adalah melakukan perhitungan performa. Perhitungan performa dilakukan berdasarkan label aktual dan label prediksi yang dihasilkan.

```
1.  for i in range(len(actual)):
2.    idx = actual[i]+1
3.    idx_predicted = predicted[i]+1
4.
5.    diff_label = abs((idx_predicted.item())-
    idx.item())
6.    diff_mean = (idx.item()-mean)**2
7.
8.    label = idx_to_label[idx.item()]
9.    fold_actual_counts[label] += 1
10.   mae_calculate_label[label] += diff_label
11.   std_calculate_label[label] += diff_mean
12.
13.   if actual[i] == predicted[i]:
14.     fold_match_counts[label] += 1
15.
16. for i in range(len(predicted)):
17.   idx = predicted[i] + 1
18.   label = idx_to_label[idx.item()]
19.   fold_predicted_counts[label] += 1
```

Kode 5.37 Potongan Kode Program pada *Method Calculate Fold Count*

Kode 5.37 menjelaskan algoritma yang digunakan untuk menghitung performa dari model yang dihasilkan. Pada jenis pengukuran MAE, perhitungan dilakukan dengan menjumlahkan selisih label aktual dan prediksi seluruhnya. Kemudian untuk menghitung standar deviasi adalah nilai label dikurangi dengan nilai rata-rata dari seluruh label dan dipangkat 2. Di dalam *method* tersebut juga akan menyimpan jumlah label aktual yang cocok dengan label prediksinya. *Dictionary label to idx* dan *idx to label* digunakan untuk memastikan indeks label dengan label aslinya sesuai.

```
1.  macro_mae_avg = mae_calculate_label[label] / actua
    l_counts[label] if actual_counts[label] > 0 else 0

2.  standard_mae  += mae_calculate_label[label]
3.
4.  std_avg = std_calculate_label[label] / (actual_cou
    nts[label]-1) if (actual_counts[label]-
    1) > 0 else 0
5.
6.  kld_actual    = actual_counts[label]/test_size
7.  kld_predicted = predicted_counts[label]/test_size

8.  diff = kld_actual/kld_predicted if kld_predicted >
     0 else 0
9.  kld_calculate = kld_actual * math.log(diff) if dif
    f > 0 else 0
10.
11. precision = match_counts[label] / predicted_counts
    [label] if predicted_counts[label] > 0 else 0
12. recall    = match_counts[label] / actual_counts[la
    bel] if actual_counts[label] > 0 else 0
13. f_measure = 2 * precision * recall / (precision +
    recall) if (precision + recall) > 0 else 0
```

Kode 5.38 Potongan Kode Program pada *Method Display Semeval Metrics*

Hasil akhir pengukuran akan dihitung menggunakan *method display semeval metrics*. *Macro average* MAE dihitung dari rata-rata seluruh MAE per label. Presisi per label dihitung dari jumlah label diprediksi yang tepat sesuai dengan label aktualnya. *Recall* per label dihitung dari jumlah label aktual yang diprediksi dengan benar. Nilai *F Measure* dihitung dari nilai *recall* dan presisi.

BAB VI
HASIL DAN PEMBAHASAN

Pada bab ini akan dijelaskan mengenai hasil dan analisis terhadap hasil yang diperoleh dari proses implementasi penelitian.

6.1 Hasil Data Crawling

Terdapat 2 kelompok data yang diperoleh dari proses *crawling twitter* pada penelitian ini. Yang pertama adalah kelompok data politik. Proses *crawling* data politik yang dilakukan sejak tanggal 1 Desember 2017 hingga 30 Maret 2018 menghasilkan 161,399 total data. Kemudian kelompok data yang kedua adalah data tidak bertopik. Proses *crawling* data tidak bertopik yang dilakukan sejak tanggal 3 Maret 2018 hingga 30 Mei 2018 menghasilkan 20,196,122 total data. Sehingga jumlah keseluruhan data yang diperoleh pada penelitian ini adalah 20,357,521 data *twitter*.

6.2 Hasil Kumpulan Data *Twitter*

6.2.1 Hasil Penggabungan Kumpulan Data

Jumlah data *twitter* yang digabung dengan data dari penelitian dengan studi kasus *e-commerce* dan studi kasus telekomunikasi adalah 64,245,029 data. Data tersebut disimpan di dalam 1 basis data MySQL yang sama.

6.2.2 Hasil Penghapusan Kumpulan Data Terduplikasi

Kumpulan data hasil penggabungan yang telah dilakukan proses penghapusan data terduplikasi, jumlahnya menjadi 32,399,220 data unik. Hal tersebut terjadi karena terdapat beberapa data *twitter* yang memiliki kesamaan di dalam teksnya.

6.2.3 Hasil Filtering Bahasa Indonesia

Kumpulan data *twitter* hasil dari proses *filtering* bahasa Indonesia menjadi 26,004,595 data. Presisi dari *filter* Bahasa Indonesia yang menggunakan *library langdetect* mencapai 99.5 %. Nilai presisi tersebut didapatkan dari pengecekan secara manual pada 1000 sampel data hasil *filtering* yang di ambil secara acak, masih terdapat 5 data yang tidak mengandung bahasa Indonesia, melainkan bahasa Melayu.

6.2.4 Hasil Anotasi Data

Jumlah kumpulan data hasil dari anotasi atau pemberian label adalah 9,948 data. Distribusi label kelas pada data hasil anotasi dapat dilihat pada Tabel 6.1.

Tabel 6.1 Distribusi Label Kelas Data Anotasi

Subtask	**Positif**		**Netral**	**Negatif**		**Total**
	2	**1**	**0**	**-1**	**-2**	
A	2,510		-	2,502		5,015
B	2,510		4,936	2,502		9,948
C	232	2278	4936	2283	219	9,948

Dari persebaran data di atas label netral memiliki distribusi data paling banyak. Distribusi data kelas positif dan negatif hampir seimbang dengan selisih 8 data.

Topic	**-2**	**-1**	**0**	**1**	**2**	**Total**
Reuni Akbar	2	284	643	495	58	**1482**
Jokowi	30	242	539	230	14	**1055**
Prabowo	47	629	472	311	33	**1492**
PDI Perjuangan	22	239	906	304	16	**1487**
Gerindra	65	330	670	390	42	**1497**
UU MD3	31	136	354	33	2	**556**
Pilgub	16	117	547	182	20	**882**

Topic	-2	-1	0	1	2	Total
Ganjar Pranowo	6	306	805	333	47	**1497**
Total	**219**	**2283**	**4936**	**2278**	**232**	**9948**

Tabel 6.2 Distribusi Label Kelas Per Topik

Tabel 6.2 menjelaskan tentang persebaran data label per topik. Dapat dilihat jika seluruh topik didominasi oleh label netral, namun hanya topik Prabowo yang label negatifnya paling tinggi.

Kemudian visualisasi fitur kata yang didapatkan dari setiap label kelas pada *subtask* B dapat dilihat pada Gambar 6.1, Gambar 6.2, dan Gambar 6.3.

Gambar 6.1 *Wordcloud* Label Kelas Positif

Dari label kelas positif kata yang memiliki frekuensi kemunculan paling tinggi adalah kata reuni akbar dan Allah. Kemudian kata-kata yang bersifat doa juga masuk ke dalam data label kelas positif. Kata reuni akbar, menolak UU MD3, dan pilgub memiliki frekuensi kemunculan yang cukup tinggi dalam label kelas positif daripada kata atau topik lain. Sampel data dengan topik reuni akbar pada label kelas positif dapat dilihat pada Tabel 6.3.

Tabel 6.3 Sampel Data Topik Reuni Akbar pada Label Kelas Positif

Id Tweet	**Teks**
936833669704007688	Elok nian akhlak Muslim Indonesia.... #ReuniAkbar212 #ReuniAkbarAlumni212
936833700435673088	Masya Allah... Pak Polisi Pimpin Sholawatan di Acara #ReuniAkbar212
936833708358737920	Insya Allah, inilah tanda2 yang selalu ditakuti musuh Islam. Tiada rencana terbaik selain rencana Allah kepada umatnya
936833711588253698	Alhamdulillah yg di Monas juga mendoakan dan galang Donasi utk korban bencana yg tengah melanda di berbagai wilayah negeri ini.

Gambar 6.2 *Wordcloud* Label Kelas Negatif

Dari label kelas negatif, kata yang memiliki frekuensi kemunculan paling tinggi adalah UU MD3. Topik UU MD3 memiliki frekuensi kemunculan paling tinggi pada kelas negatif daripada topik lain. Namun topik Reuni Akbar tidak hanya muncul pada label kelas positif saja, pada label kelas negatif juga memiliki frekuensi kemunculan yang cukup besar. Kata-kata umpatan juga masuk ke dalam label kelas negatif. Sampel data dengan topik UU MD3 pada label kelas negatif dapat dilihat pada Tabel 6.4.

Tabel 6.4 Sampel Data Topik UU MD3 pada Kelas Label Negatif

Id Tweet	Teks
977164051414315008	@detikcom Terlambat uu md3 sdh berlaku. Knp skrang mau demo, kmrin kmna sj
977172571954098178	UU MD3 berlaku utk DPRD nggak sih? Pengen ngasu-asuke Wong.... https://t.co/lbW7KfMhNd
977179983306092544	Dusta Menkumham Yosanna & Presiden Jokowi dalam UU MD3 Netizen: Apakah syarat jadi penguasa harus pandai BERDUSTA? #keripikpedas_ Folow: @keripikpedas_ Twitter: https://t.co/oD69EJKfeT Telegram:?https://t.co/vetaUfaQzv
977184698110697473	UU MD3 yang membungkam hak-hak kita sebagai warga negara Indonesia. Satu kata, TOLAK

Gambar 6.3 *Wordcloud* Label Kelas Netral

Dari label kelas netral, kata yang memiliki frekuensi kemunculan paling tinggi adalah pilgub. Hal ini disebabkan karena data dengan topik pilgub, memiliki banyak data *tweet* dengan bentuk promosi atau kampanye, sehingga anotator cenderung memberikan label netral terhadap data tersebut. Topik reuni akbar juga memiliki frekuensi yang cukup besar pada label kelas netral. Sampel data dengan topik pilgub pada label kelas netral dapat dilihat pada Tabel Tabel 6.5.

Tabel 6.5 Sampel Data *Tweet* Topik Pilgub pada Label Kelas Netral

Id Tweet	Teks
978496829531680768	Jangan lewatkan debat publik pertama Pilgub Sulawesi Selatan 2018 di @KompasTV, Rabu, 28 Maret 2018, pukul 19.30 WITA bersama para kandidat #DebatPilgubSulsel https://t.co/K78FHa1EVF @KompasTV_Mks #DebatPilgubSulsel #Pilkadaserentak2018"
978436198237593600	Debat Pilgub Sulsel pertama dan Eksklusif di Kompastv https://t.co/Kz6AUtxnrc
978512252503543808	Mendongkrak Partisipasi Pemilih di Pilgub Jateng 2018 https://t.co/vE1iH4d3UX
978517955330715649	PILGUB JATENG 2018: Jelang Pilgub, Baru 10.662 Warga Klaten Rekam Data E-KTP https://t.co/wOAfpR4US

6.2.5 Hasil Penghapusan Tanda Baca dan Simbol, *Tokenizing* dan *Casefolding*

Hasil pra-pemrosesan pada tahap ini tidak akan mengubah jumlah kumpulan data. Sampel data hasil dari penghapusan tanda baca dan simbol, *tokenizing* dan *casefolding* dapat dilihat pada Tabel 6.6.

Tabel 6.6 Sampel Data Hasil Penghapusan Tanda Baca dan Simbol, *Tokenizing*, dan *Casefolding*

Data Sebelum	Data Sesudah
Kau sebut mereka RADIKAL Kalau Kami menyebutnya BIDADARI (tapi gada sayapnya ?) RT = Setuju RT Setuju pake BANGET ! #ReuniAkbar212	'kau', 'sebut', 'mereka', 'radikal', 'kalau', 'kami', 'menyebutnya', 'bidadari', '!', '(', 'tapi', 'gada', 'sayapnya', '?', ')', 'rt', 'setuju', 'rt', 'setuju', 'pake', 'banget', '!', '<hash_tag>', 'reuniakbar212'
#MenolakLupa Kedubes Palestina di Jakarta: Kami Tidak Mau Bendera Kami (Palestina) dikibarkan pihak anti pemerintah (Indonesia) untuk memicu konflik di Indonesia. Cc Anti Pemerintah @RizieqSyihabFPI @DPP_LPI @bachtiarnasir @GNPF_MUI #ReuniAkbar212	'<hash_tag>', 'menolaklupa', 'kedubes', 'palestina', 'di', 'jakarta', 'kami', 'tidak', 'mau', 'bendera', 'kami', '(', 'palestina', ')', 'dikibarkan', 'pihak', 'anti', 'pemerintah', '(', 'indonesia', ')', 'untuk', 'memicu', 'konflik', 'di', 'indonesia', '.', 'cc', 'anti', 'pemerintah<mention>', '<mention>', '<mention>', '<mention>', '<hash_tag>', 'reuniakbar212'

6.3 Model *Word2Vec*

Jumlah data yang digunakan untuk *training* model *word2vec* adalah 26,004,595 data yang merupakan luaran pada tahap pra-pemrosesan sebelumnya. Penelitian ini menghasilkan 2 model *word2vec* dengan *learning algorithm skip-gram* dan CBOW yang digunakan untuk *input word vector* pada *training* model CNN. Model *word2vec* yang dihasilkan memiliki jumlah *vocabulary* sebanyak 600,978 kosakata dengan 300 dimensi vektor.

6.4 Model CNN

Untuk memberikan referensi terhadap performa hasil *training* model CNN, penelitian ini juga melakukan percobaan *training* model klasifikasi teks dengan menggunakan algoritma lain yaitu SVM dan *naive bayes*. Percobaan *training* tersebut menggunakan kumpulan data yang sama dengan yang digunakan pada *training* model CNN. *Training* akan dilakukan di setiap *subtask* dengan menggunakan parameter deterministik. Pada *training* SVM, kernel yang digunakan adalah *linear* SVM, kemudian untuk *naive bayes* menggunakan *multinomial*. Metode *cross validation* juga digunakan pada *training* model SVM dan *naive bayes* dengan 10 *fold*.

Konfigurasi awal yang digunakan untuk melakukan *training* model CNN pertama kali dapat dilihat pada Tabel 6.7.

Tabel 6.7 Konfigurasi Awal *Training* Model CNN

Parameter	Nilai
Input word vector	*Word2vec*
Filter region size	(3,4,5)
Feature maps	100
Activation function	ReLU
Pooling	1-*max pooling*
Dropout rate	0.5
Norm contstraint	3

6.4.1 *Subtask* A

Subtask A merupakan *task* klasifikasi yang terdiri dari 2 label kelas yaitu positif dan negatif. Referensi evaluasi pengukuran performa model CNN pada *subtask* A dengan menggunakan algoritma SVM dan *naive bayes* dapat dilihat pada Tabel 6.8.

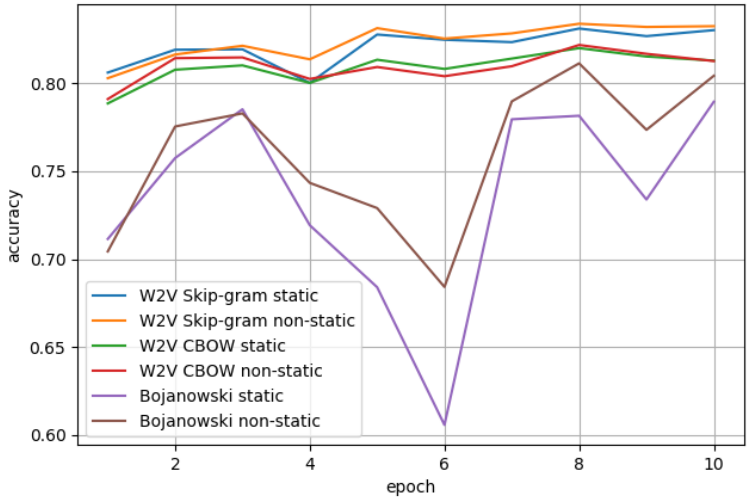
Tabel 6.8 Performa *Training* Model SVM dan *Naive Bayes Subtask* A

Alogitma	***Accuracy***	***Avg Recall***	***Avg Precision***
Naive bayes	0.800_1	0.780_1	0.800_1
SVM	0.792_2	0.777_2	0.787_2

Pengukuran utama yang digunakan pada *subtask* A adalah akurasi. Akurasi digunakan sebagai pengukuran utama karena *subtask* A memiliki distribusi kelas data positif dan negatif yang seimbang. Berdasarkan referensi evaluasi pengukuran pada Tabel 6.8 algoritma *naive bayes* memiliki nilai akurasi paling tinggi dibandingkan dengan SVM. Nilai akurasi yang berhasil dicapai oleh *naive bayes* adalah 0.800, maka setidaknya akurasi yang dihasilkan oleh *training* CNN pada *subtask* A dapat mencapai nilai 0.800 atau lebih.

6.4.1.1 Pengaruh *Input Word Vectors*

Model klasifikasi teks CNN dimulai dengan distribusi dari representasi kata atau vektor kata sebagai *input*. Model *word embeddings* yang digunakan sebagai parameter *input word vectors* pada *subtask* A meliputi model *word2vec skip-gram* dan CBOW hasil *training* model pada tahap sebelumnya, dan model *fastext* dari Bojanowski.



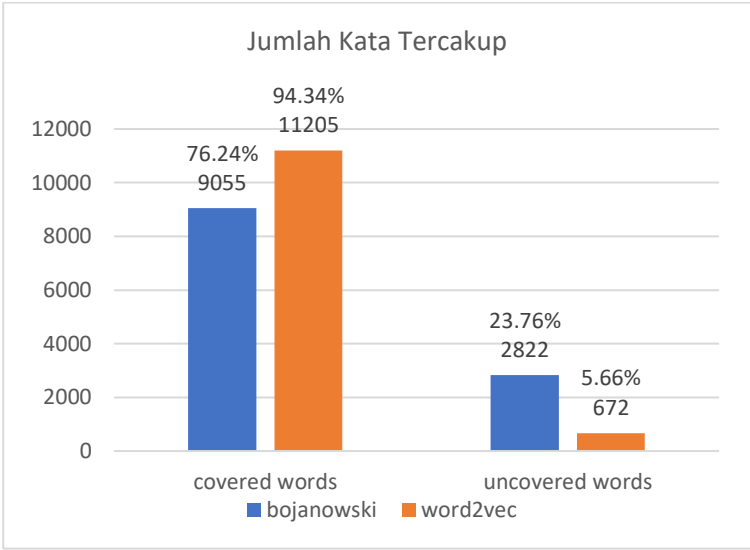
Gambar 6.4 Grafik Pengaruh *Input Word Vector Subtask* A

Gambar 6.4 menjelaskan performa dari setiap parameter *input word vectors* yang digunakan. Model *word2vec skip-gram non-static* memiliki performa paling optimal dibandingkan dengan model lain, karena model *word2vec skip-gram non-static* memiliki grafik performa paling tinggi pada setiap *epoch*. Model *word2vec* CBOW memiliki performa dengan pola yang hampir sama dengan model *word2vec skip-gram* namun setelah *epoch* ke-4 perlahan bergerak menjauh lebih rendah. Perbedaan pola terlihat jelas antara performa model bojanowski dan *word2vec*. Performa yang dihasilkan oleh model bojanowski *static* maupun *non-static* lebih rendah dibandingkan model *word2vec*, kemudian terjadi penurunan performa secara drastis pada *epoch* ke-6. Detail nilai performa setiap parameter *input word vectors* yang digunakan pada *subtask* A dapat dilihat pada Tabel 6.9.

Tabel 6.9 Pengaruh *Input Word Vector Subtask* A

Model	Embed Mode	Acc	Avg Recall	F1
Word2Vec Skip Gram	non-static	0.832_1	0.822_1	0.822_1
Word2Vec Skip Gram	static	0.830_2	0.815_2	0.818_2
Word2vec CBOW	static	0.813_3	0.802_4	0.802_4
Word2vec CBOW	non-static	0.813_4	0.806_3	0.804_3
Bojanowski	non-static	0.804_5	0.793_5	0.793_5
Bojanowski	static	0.789_6	0.782_6	0.779_6

Berdasarkan Tabel 6.9 nilai akurasi model *word2vec skip-gram non-static* mencapai 0.832. Nilai *average recall* dan F_1 dari model *word2vec skip-gram non-static* juga merupakan yang paling tinggi dibandingkan *parameter word input vectors* lain. Secara konsisten peringkat nilai *average recall* dan F_1 dari seluruh parameter *input word vectors* mengikuti peringkat nilai akurasinya, namun hanya model *word2vec* CBOW yang memiliki perbedaan peringkat dengan akurasinya. Nilai akurasi model *word2vec skip-gram* pada *subtask* A lebih baik dibandingkan dengan model *word2vec* CBOW. Model *word2vec* cenderung memiliki nilai akurasi lebih tinggi daripada model *fastext* Bojanowski karena jumlah kata yang tercakup oleh model *word2vec* pada saat proses *training* lebih banyak dan jumlah kata yang tidak tercakup lebih sedikit.

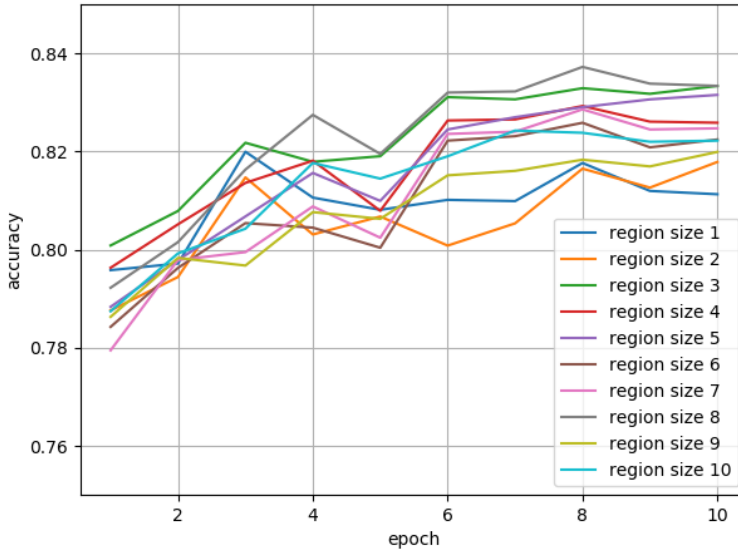


Gambar 6.5 Jumlah Kata *Subtask* A yang Tercakup Oleh Model *Word Embeddings*

Tahap percobaan selanjutnya akan menggunakan model *word2vec skip-gram non-static* sebagai parameter *input word vectors*, karena memiliki performa paling optimal pada percobaan *input word vectors*.

6.4.1.2 Pengaruh Filter Region Size

Percobaan *training* model CNN dilakukan kembali dengan mengubah parameter *filter region size*. Percobaan pertama akan menggunakan parameter *single filter region size* 1-10.



Gambar 6.6 Grafik Pengaruh *Single Filter Region Size Subtask* A

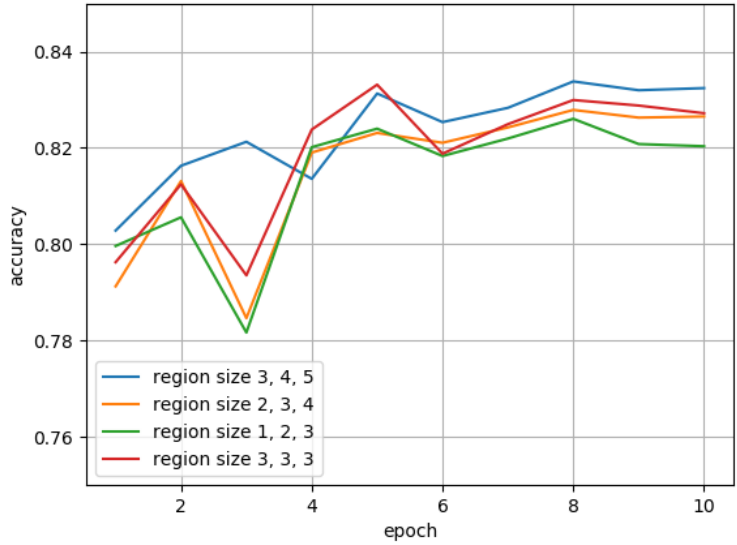
Berdasarkan Gambar 6.6 seluruh *region size* memiliki performa dengan pola yang hampir sama dengan tren yang meningkat. *Region size* 1 dan 2 tidak mengalami peningkatan performa yang cukup besar seperti *region size* lain setelah *epoch* ke-5. Performa paling baik pada saat *epoch* ke-10 dimiliki *region size* 3 dan 8. Detail nilai performa dengan parameter *single filter region size* dapat dilihat pada Tabel 6.10.

Tabel 6.10 Pengaruh *Single Filter Region Size Subtask* A

Region Size	***Acc***	***Avg Recall***	***F1***
8	0.833_1	0.817_3	0.821_2
3	0.833_1	0.820_2	0.822_1
5	0.832_2	0.822_1	0.822_1
4	0.826_3	0.814_5	0.815_3
7	0.825_4	0.813_6	0.814_4
6	0.822_5	0.816_4	0.815_3

Region Size	*Acc*	*Avg Recall*	F_1
10	0.822_6	0.812_7	0.812_5
9	0.820_7	0.812_7	0.810_6
2	0.818_8	0.809_8	0.808_7
1	0.811_9	0.803_9	0.800_8

Nilai akurasi tertinggi dicapai oleh *region size* 8 dan 3. Akurasi yang berhasil dicapai *region size* 8 dan 3 yaitu sama 0.833. Peringkat nilai *average recall* dan F_1 *region size* 8 dan 3 berbeda dengan peringkat nilai akurasinya. *Average recall* tertinggi dimiliki oleh *region size* 5 sedangkan nilai F_1 tertinggi dimiliki *region size* 3. *Region size* 8 dan 3 akan digunakan untuk percobaan selanjutnya karena memiliki nilai akurasi lebih optimal dibandingkan *region size* lain . Percobaan *training* selanjutnya menggunakan parameter *multiple filter region size* dengan kombinasi 3 angka yang mengandung angka 8 dan 3 sebagai *single filter region size* terbaik.



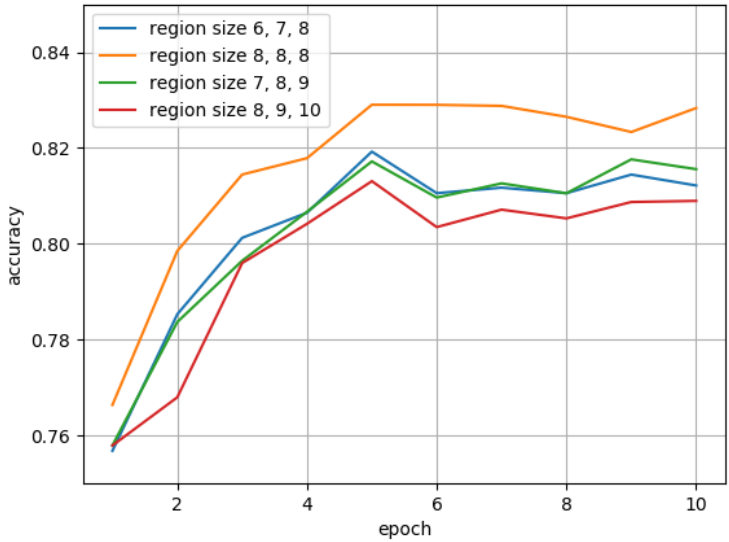
Gambar 6.7 Grafik Pengaruh *Multiple Filter Region Size 3 Subtask* A

Berdasarkan grafik hasil percobaan *multiple filter region size* 3 pada Gambar 6.7 seluruh *region size* memiliki performa dengan pola yang hampir sama pada setiap *epoch* dengan tren meningkat. Pada *epoch* ke-3 seluruh *region size* mengalami penurunan performa kecuali *region size* (3, 4, 5). *Multiple filter region size* (3, 4, 5) memiliki performa paling tinggi pada *epoch* ke-10. Detail nilai performa dari percobaan *training multiple filter region size* dapat dilihat pada Tabel 6.11.

Tabel 6.11 Pengaruh *Multiple Filter Region Size* 3 *Subtask A*

Region Size	***Acc***	***Avg Recall***	***F_1***
3,4,5	0.832_1	0.822_1	0.822_1
3,3,3	0.827_2	0.818_2	0.818_2
2,3,4	0.827_2	0.818_2	0.817_3
1,2,3	0.820_3	0.816_3	0.813_4

Nilai akurasi yang dicapai dengan *multiple filter region size* (3, 4, 5) adalah 0.832. Nilai *average recall* dan F_1 *multiple filter region size* (3, 4, 5) juga merupakan yang paling tinggi dibandingkan *region size* lain. *Region size* (3, 3, 3) memiliki nilai akurasi dan *average recall* yang sama dengan *region size* (2, 3, 4). Percobaan selanjutnya adalah *multiple filter region size* dengan menggunakan kombinasi angka 8.



Gambar 6.8 Grafik Pengaruh *Multiple Filter Region Size* 8 *Subtask* A

Berdasarkan hasil percobaan *training* pada Gambar 6.8 Seluruh *region size* memiliki tren performa yang meningkat. performa paling optimal dimiliki oleh *region size* (8, 8, 8). Akurasi yang didapatkan *region size* (8, 8, 8) pada setiap *epoch* merupakan yang paling tinggi dibandingkan *region size* lain. Performa paling rendah dimiliki oleh *region size* (8, 9, 10). Detail nilai akurasi setiap parameter *multiple filter region size* dapat dilihat pada Tabel 6.12.

Tabel 6.12 Pengaruh *Multiple Filter Region Size* 8 *Subtask* A

Region Size	***Acc***	***Avg Recall***	***F_1***
8,8,8	0.828_1	0.813_1	0.816_1
7,8,9	0.816_2	0.802_2	0.803_2
6,7,8	0.812_3	0.801_3	0.801_3
8,9,10	0.809_4	0.802_4	0.799_4

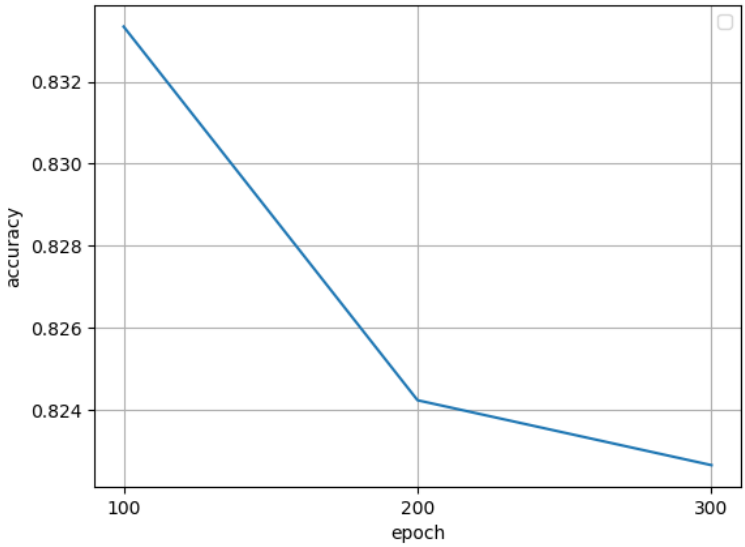
Akurasi yang berhasil dicapai *region size* (8, 8, 8) adalah 0.813. Nilai *average recall* dan F_1 *region size* (8, 8, 8) juga merupakan

yang paling tinggi dibandingkan dengan yang lain. Secara konsisten peringkat nilai F_1 dan *average recall* seluruh *region size* sama dengan nilai akurasinya.

Nilai akurasi yang didapatkan pada percobaan seluruh *multiple filter region size subtask* A tidak lebih tinggi daripada percobaan *single region size*. Akurasi terbaik yang didapatkan pada percobaan *multiple filter region size* adalah 0.832, lebih rendah daripada akurasi terbaik *single filter region size* yaitu 0.833. Selisih nilai akurasi antara parameter terbaik *single* dan *multiple filter region size* adalah 0.01, namun nilai *average recall multiple filter region size* lebih tinggi daripada *single filter region size.* Percobaan selanjutnya akan menggunakan *region size* 3 karena memiliki nilai akurasi paling optimal.

6.4.1.3 Pengaruh Nilai *Feature Maps* Setiap *Filter Region Size*

Percobaan *training* selanjutnya adalah dengan mengubah parameter *feature maps*. Nilai parameter *feature maps* yang digunakan adalah 100-300, karena perangkat keras yang digunakan hanya dapat melakukan *training* model CNN hingga *feature maps* ke-300.



Gambar 6.9 Grafik Pengaruh *Feature Maps Subtask* A

Gambar 6.9 menunjukkan tren performa dari setiap *feature maps*. Performa paling optimal dimiliki oleh *feature maps* 100. Semakin besar ukuran *feature maps* akurasi semakin turun. Detail nilai performa *training* dengan parameter *feature maps* dapat dilihat pada Tabel 6.13.

Tabel 6.13 Pengaruh *Feature Maps Subtask* A

Feature Maps	***Acc***	***Avg Recall***	***F_1***
100	0.833_1	0.820_1	0.822_1
200	0.824_2	0.816_2	0.815_2
300	0.823_3	0.815_3	0.814_3

Feature maps 100 memiliki nilai akurasi mencapai 0.833. Nilai *average recall* dan F_1 dari *feature maps* 100 juga merupakan nilai tertinggi dibandingkan dengan *feature maps* lain. Melihat seluruh hasil *feature maps* berdasarkan peringkat

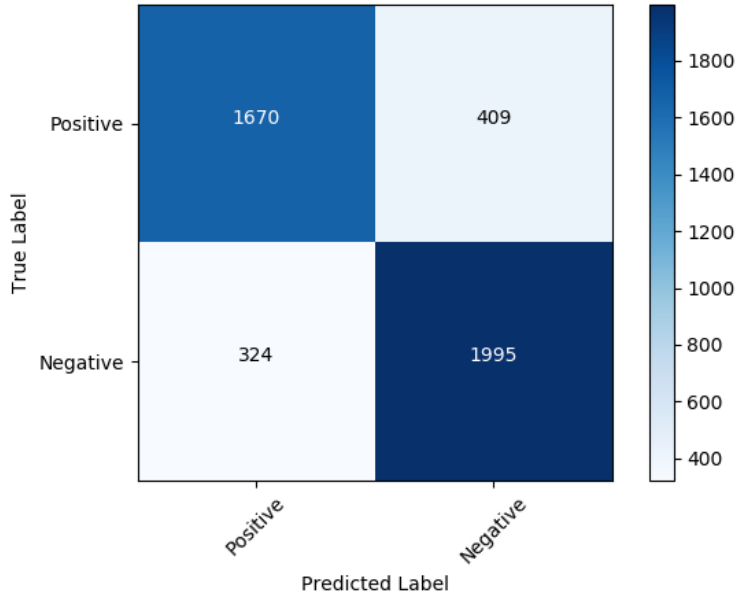
nilai akurasi, urutan nilai *average recall* dan F_1 berbanding lurus dengan urutan peringkat nilai akurasinya.

Berdasarkan hasil percobaan *training* dengan berbagai skenario, konfigurasi parameter yang memiliki nilai akurasi paling optimal pada *subtask* A dapat dilihat pada Tabel 6.14. Nilai akurasi yang diperoleh dengan konfigurasi parameter tersebut adalah 0.833.

Tabel 6.14 Konfigurasi Parameter Paling Optimal *Subtask* A

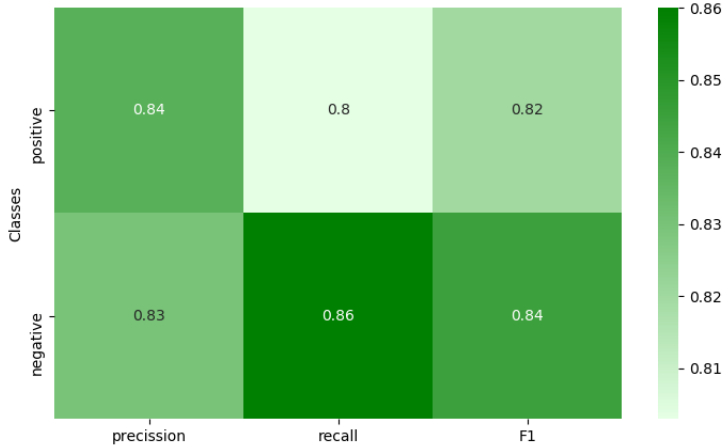
Input Word Vector	***Region Size***	***Feature Maps***
Word2Vec Skip-gram non-static	3	100

Confusion matrix hasil dari pelatihan model CNN dengan parameter pada Tabel 6.14 dibuat untuk melihat pada label kelas mana kesalahan prediksi sering terjadi.



Gambar 6.10 *Confusion Matrix Subtask* A

Gambar 6.10 menunjukkan *confusion matrix* dari konfigurasi parameter terbaik *subtask* A. Data *tweet* yang diprediksi dengan benar lebih dominan. Hal tersebut ditunjukkan dengan warna biru tua pada *confusion matrix subtask* A. Jumlah data *tweet* yang salah diprediksi positif mencapai 16% atau 324 data, kemudian data *tweet* yang salah diprediksi negatif oleh model mencapai 17 % atau 409 data.



Gambar 6.11 Hasil Pengukuran Klasifikasi *Subtask* A

Gambar 6.11 menunjukkan nilai presisi kelas positif lebih baik dibandingkan dengan kelas negatif. Nilai *recall* dan F_1 label negatif lebih baik daripada kelas positif. Visualisasi tersebut menunjukkan jika label kelas positif dan negatif memiliki distribusi data yang sama selama proses *training* model karena perbedaan hasil pengukuran setiap label tidak terlalu besar.

6.4.2 *Subtask* B

Subtask B merupakan *task* klasifikasi yang terdiri dari 3 label kelas yaitu positif, netral dan negatif. Referensi evaluasi pengukuran performa model CNN pada *subtask* B dengan menggunakan algoritma SVM dan *naive bayes* dapat dilihat pada Tabel 6.15.

Tabel 6.15 Performa *Training* Model SVM dan *Naive bayes subtask* B

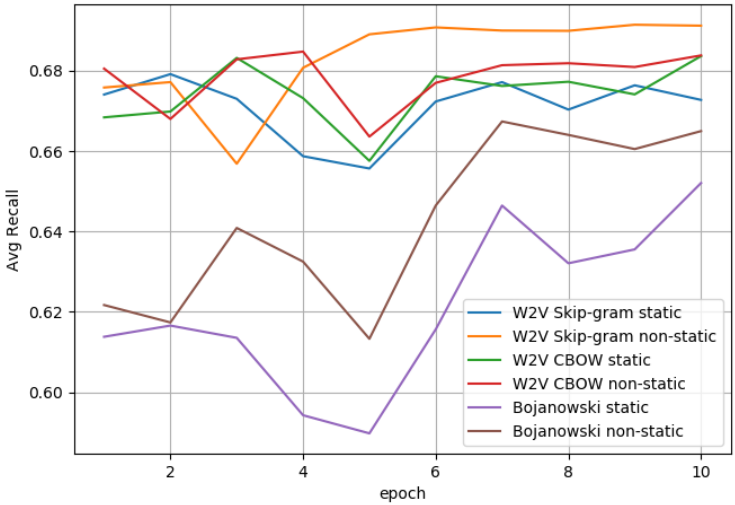
Alogitma	***Avg Recall***	***Avg Precision***	***Acc***
Naive bayes	0.634_1	0.655_1	0.671_1
SVM	0.633_2	0.635_2	0.663_2

Pengukuran utama yang digunakan pada *subtask* B adalah *average recall*. *Average recall* lebih baik dalam melakukan evaluasi terhadap data dengan distribusi kelas yang tidak seimbang daripada menggunakan akurasi standar.

Berdasarkan referensi evaluasi pengukuran performa pada Tabel 6.15 algoritma *naive bayes* memiliki nilai *average recall* paling tinggi dibandingkan dengan SVM. Nilai *average recall* yang diperoleh *naive bayes* mencapai 0.634, maka setidaknya model CNN yang dihasilkan dapat memiliki nilai *average recall* mencapai nilai 0.634 atau lebih pada *subtask* B.

6.4.2.1 Pengaruh *Input Word Vectors*

Model klasifikasi teks CNN dimulai dengan distribusi dari representasi kata atau vektor kata sebagai *input*. Model *word embeddings* yang digunakan sebagai parameter *input word vectors* pada *subtask* B meliputi model *word2vec skip-gram* dan CBOW hasil *training* model pada tahap sebelumnya, dan model *fastext* dari Bojanowski.



Gambar 6.12 Grafik Pengaruh *Input Word Vector Subtask* B

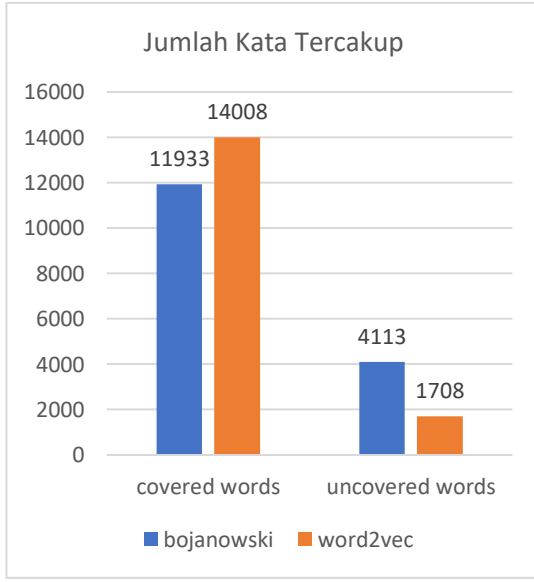
Gambar 6.12 menjelaskan performa dari setiap parameter *input word vectors* yang digunakan pada *subtask* B. Model *word2vec non-static* memiliki grafik performa yang paling baik dibandingkan dengan model lain. *Word2vec skip-gram non static* memiliki tren performa yang meningkat walaupun terjadi penurunan pada *epoch* ke-3. Detail nilai performa setiap parameter *input word vectors* dapat dilihat pada Tabel 6.16.

Tabel 6.16 Pengaruh *Input Word Vector Subtask* B

Model	***Embed Mode***	***Avg Recall***	F_1^{PN}	***Acc***
Word2Vec Skip Gram	non-static	0.691_1	0.656_1	0.709_1
Word2vec CBOW	non-static	0.684_2	0.646_2	0.698_1
Word2vec CBOW	static	0.684_3	0.643_3	0.697_2
Word2Vec Skip Gram	static	0.673_4	0.632_4	0.689_3
Bojanowski	non-static	0.665_5	0.614_5	0.672_4

Bojanowski	static	0.652_6	0.614_5	0.687_5

Berdasarkan Tabel 6.16 model *word2vec skip-gram non-static* memiliki nilai *average recall* paling tinggi dibandingkan dengan model *word embeddings* lain. Secara konsisten pengukuran F_1^{PN} dan akurasi pada model *word2vec skip-gram non-static* juga merupakan yang paling tinggi. Perbedaan performa cukup signifikan terjadi pada model *word2vec* dan *fasttext* Bojanowski yang digunakan. Hal tersebut terjadi karena perbedaan jumlah kata yang tercakup oleh model. Semakin tinggi jumlah kata yang tercakup maka nilai *average recall* juga akan semakin meningkat. Jumlah kata yang tercakup pada model *word2vec* dan *fasttext* Bojanowski dapat dilihat pada Gambar 6.13.



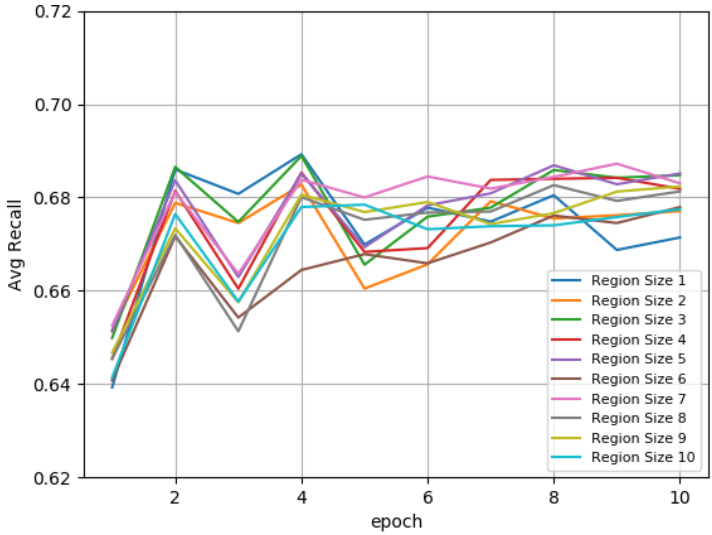
Gambar 6.13 Jumlah Kata *Subtask B* yang Tercakup Model *Word Embeddings*

Model *word2vec skip-gram non-static* akan digunakan sebagai parameter *input word vectors* pada skenario percobaan

training selanjutnya karena memiliki performa yang paling optimal.

6.4.2.2 Pengaruh *Filter Region Size*

Training model CNN dilakukan kembali dengan mengubah parameter *filter region size*, untuk mengetahui pengaruh terhadap performa *training* model. Parameter yang digunakan pada percobaan pertama adalah *single filter region size* 1-10.



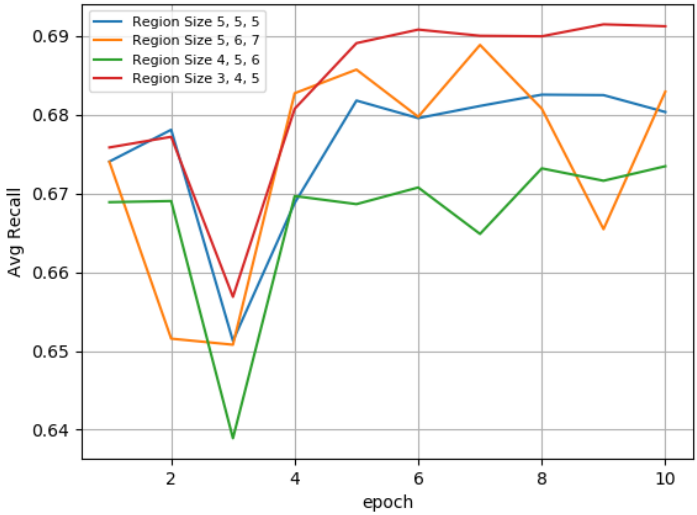
Gambar 6.14 Grafik Pengaruh *Single Filter Region Size*

Gambar 6.14 menjelaskan performa dari pengaruh *single filter region size*. Seluruh *region size* memiliki pola yang hampir sama pada setiap *epoch*. *Single filter region size* 5 dan 3 memiliki nilai paling tinggi pada *epoch* 10. Detail nilai performa pada setiap *single filter region size* dapat dilihat pada Tabel 6.17.

Tabel 6.17 Pengaruh *Single Filter Region Size Subtask* B

Region Size	***Avg Recall***	F_1^{PN}	***Accuracy***
5	0.685_1	0.650_1	0.702_2
3	0.685_1	0.644_3	0.696_3
7	0.683_2	0.640_5	0.694_4
9	0.682_3	0.645_2	0.702_2
4	0.682_4	0.645_2	0.702_2
8	0.681_5	0.644_4	0.703_1
6	0.678_6	0.639_7	0.693_6
10	0.677_7	0.640_6	0.702_2
2	0.677_7	0.638	0.693_5
1	0.671_8	0.635_9	0.690_7

Berdasarkan Tabel 6.17 *single filter region size* 5 memiliki nilai *average recall* tertinggi, begitu pun juga dengan nilai F_1^{PN}. Namun *single filter region size* 5 memiliki nilai akurasi tertinggi kedua. *Single filter region size* 3 juga memiliki nilai *average recall* tertinggi, akan tetapi nilai F_1^{PN} dan akurasinya merupakan nilai tertinggi ketiga. Sehingga nilai *single filter region size* yang paling optimal pada *subtask* B adalah 5. Langkah selanjutnya adalah melakukan percobaan dengan parameter *multiple region size*. Nilai parameter yang digunakan merupakan kombinasi 3 angka yang mengandung nilai dari *single region size* terbaik.



Gambar 6.15 Grafik Pengaruh *Multiple Region Size Subtask* B

Gambar 6.15 menjelaskan performa dari pengaruh *multiple filter region size*. Berdasarkan performa *training* setiap *epoch*, *multiple region size* 3, 4, 5 merupakan yang paling baik, walaupun terjadi penurunan pada *epoch* 3 namun nilai dari seluruh *epoch* tetap paling tinggi dibandingkan dengan *multiple filter region size* lain. Untuk detail performa dari *multiple region size* dapat dilihat pada Tabel 6.18.

Tabel 6.18 Pengaruh *Multiple Filter Region Size Subtask* B

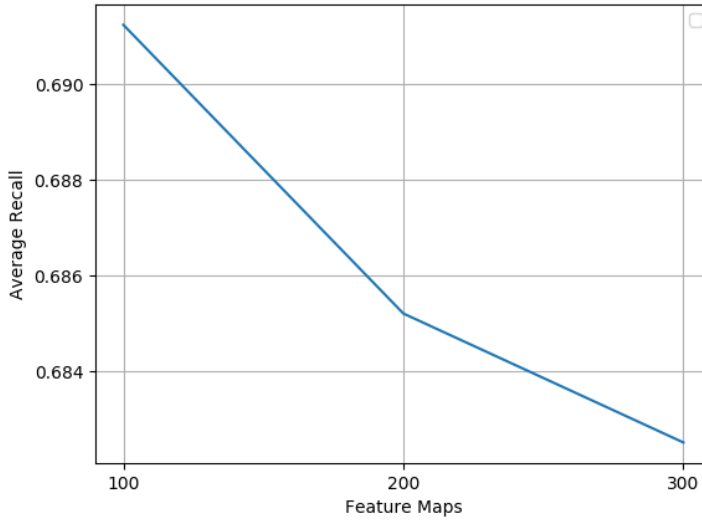
Region Size	***Avg Recall***	***F_1^{PN}***	***Accuracy***
3,4,5	0.691_1	0.656_1	0.709_1
5,6,7	0.683_2	0.650_2	0.702_2
5,5,5	0.680_3	0.645_3	0.696_4
4,5,6	0.673_4	0.637_4	0.697_3

Berdasarkan hasil pada Tabel 6.18 nilai *average recall* yang paling optimal adalah *multiple filter region size* 3,4,5.

Begitu pun juga dengan nilai F_1^{PN} dan akurasi, *multiple filter region size* 3,4,5 memiliki nilai paling tinggi.

6.4.2.3 Pengaruh Nilai *Feature Maps* Setiap *Filter Region Size*

Nilai *multiple filter region size* terbaik akan digunakan kembali untuk melakukan percobaan *training* model dengan mengubah nilai *feature maps*. Hal tersebut dilakukan untuk mengetahui pengaruh *feature maps* terhadap performa *training*. Karena keterbatasan spesifikasi perangkat keras yang digunakan, nilai *feature maps* yang digunakan hanya 100, 200 dan 300.



Gambar 6.16 Grafik Pengaruh *Feature Maps Subtask* B

Gambar 6.16 menjelaskan performa dari pengaruh *feature maps*. Berdasarkan grafik di atas, *feature maps* 100 merupakan yang paling baik dibandingkan dengan *feature maps* yang lain. Semakin besar ukuran *feature maps*, nilai *average recall* semakin turun. Untuk detail performa setiap *feature maps* dapat dilihat pada Tabel 6.19.

Tabel 6.19 Pengaruh *Feature Maps Subtask* B

Feature Maps	***Avg Recall***	F_1^{PN}	***Accuracy***
100	0.691_1	0.656_1	0.709_2
200	0.685_2	0.655_2	0.710_1
300	0.683_3	0.647_3	0.707_3

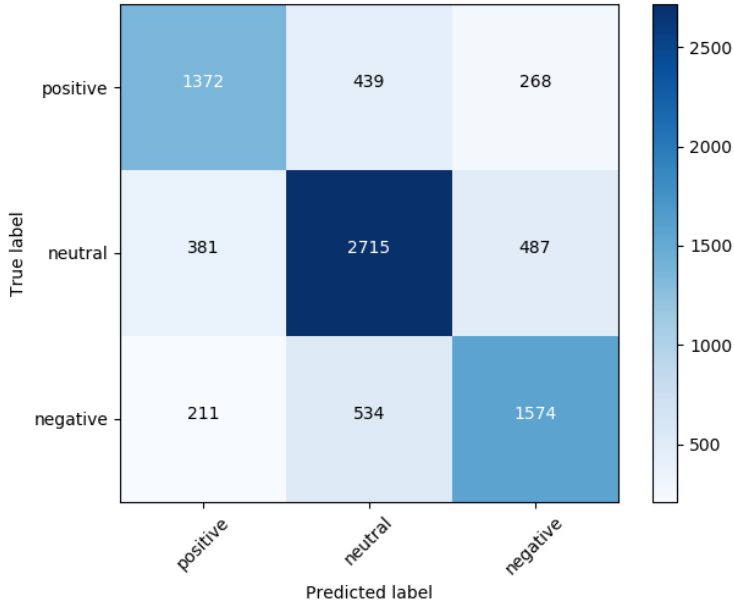
Berdasarkan hasil pada Tabel 6.19 *feature maps* terbaik adalah 100 dengan nilai *average recall* dan F_1^{PN} paling tinggi dibandingkan dengan *feature maps* lain. Kemudian dapat diketahui juga pada *subtask* B semakin besar ukuran *feature maps* maka nilai *average recall* semakin menurun. Namun untuk akurasi, nilai optimalnya berada pada *feature maps* 200.

Dari seluruh skenario yang telah dilakukan, konfigurasi parameter yang paling optimal pada *subtask* B dapat dilihat pada Tabel 6.20.

Tabel 6.20 Konfigurasi Parameter Paling Optimal *Subtask* B

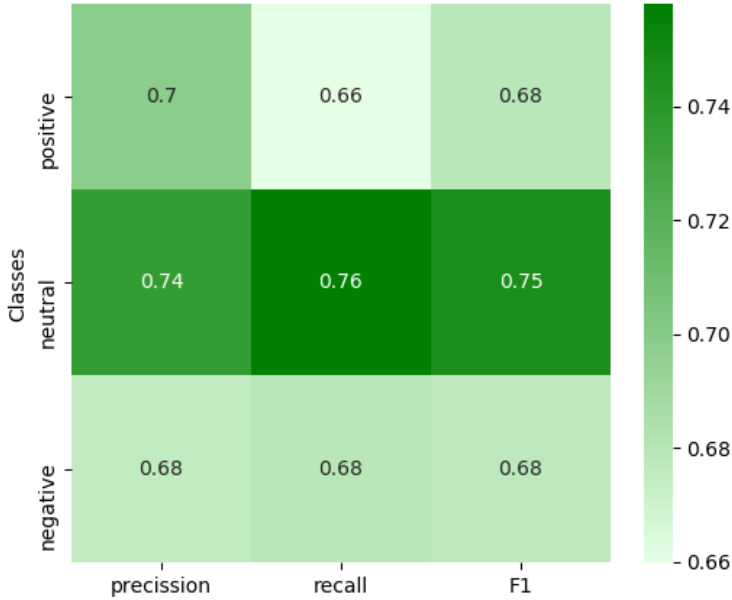
Input Word Vector	***Region Size***	***Feature Maps***
Word2Vec Skip-gram non-static	3, 4, 5	100

Konfigurasi parameter tersebut dapat mencapai nilai *average recall* hingga 0.691. Kemudian berdasarkan percobaan yang dilakukan pada *filter region size*, nilai yang dihasilkan dengan konfigurasi *multiple filter region size* lebih tinggi dibandingkan dengan *single filter region size*.



Gambar 6.17 *Confusion Matrix Subtask* B

Gambar 6.17 merupakan *confusion matrix* dari konfigurasi parameter terbaik *subtask* B. Kesalahan prediksi paling sering terjadi pada label kelas netral, dari seluruh data *tweet* yang diprediksi netral, sekitar 12% atau 439 data merupakan label kelas positif, dan 14% atau 534 data merupakan kelas negatif.



Gambar 6.18 Hasil Pengukuran Klasifikasi *Subtask* B

Berdasarkan Gambar 6.18 label netral memiliki nilai *precision*, *recall* dan F_1 paling tinggi, hal tersebut menunjukkan jika label kelas netral pada *subtask* B lebih dominan selama proses *training* model.

6.4.3 *Subtask* C

Subtask C merupakan *task* klasifikasi yang terdiri dari 5 label kelas yaitu sangat positif, positif, netral, negatif dan sangat negatif. Referensi evaluasi pengukuran model CNN pada *subtask* A menggunakan algoritma SVM dan *naive bayes*. Hasil dari *training* model menggunakan algoritma SVM dan *naive bayes* dapat dilihat pada Tabel 6.21.

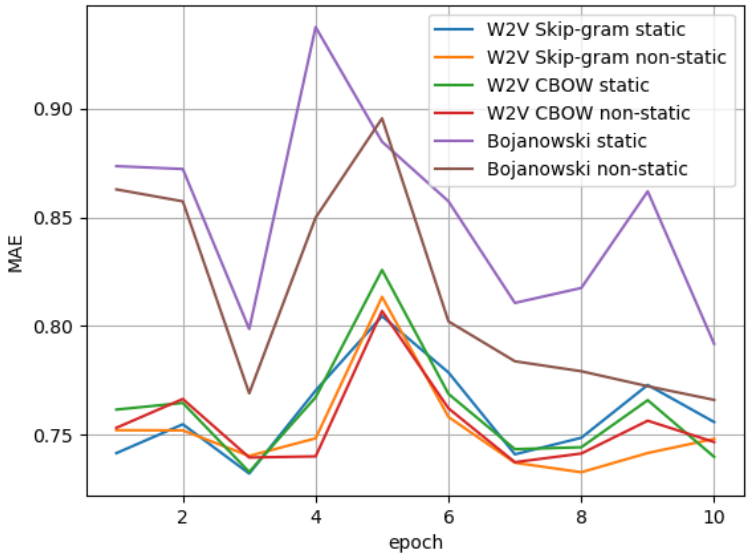
Tabel 6.21 Performa *Training* Model SVM dan *Naive Bayes Subtask* C

Alogitma	**MAE^{M}**	***Avg Recall***	***Accuracy***
SVM	0.780_1	0.392_1	0.620_2
Naive bayes	0.806_2	0.363_2	0.627_1

Pengukuran utama yang digunakan pada *subtask* C adalah *macro* MAE (*Mean Absolutre Error*). *Macro* MAE digunakan sebagai pengukuran utama karena lebih optimal untuk kumpulan data yang distribusi label kelasnya tidak seimbang dan *subtask* C merupakan *task* klasifikasi ordinal. Berdasarkan Tabel 6.21 performa SVM merupakan yang paling optimal karena memiliki nilai *macro* MAE paling rendah dibandingkan dengan SVM. Algoritma SVM memiliki nilai *macro* MAE 0.780, maka setidaknya model CNN yang dihasilkan memiliki nilai MAE 0.780 atau lebih rendah pada *subtask* C.

6.4.3.1 Pengaruh *Input Word Vectors*

Model klasifikasi teks CNN dimulai dengan distribusi dari representasi kata atau vektor kata sebagai *input*. Model *word embeddings* yang digunakan pada *subtask* C meliputi model *word2vec skip-gram* dan CBOW hasil *training* model pada tahap sebelumnya, dan model *fastext* dari Bojanowski.



Gambar 6.19 Grafik Pengaruh *Input Word Vector Subtask* C

Gambar 6.19 menjelaskan tentang performa *macro* MAE pada *training* model CNN menggunakan beberapa parameter *input word vector*. Seluruh model memiliki pola yang hampir sama pada setiap *epoch*. Peningkatan nilai *macro* MAE secara drastis terjadi pada *epoch* ke-5 untuk seluruh model. Tingkat penurunan nilai *macro* MAE model *bojanowski* lebih besar dibandingkan dengan mode *wordvec*. Hanya model *word2vec skip-gram static* yang memiliki nilai *macro* MAE pada *epoch* ke-10 lebih tinggi daripada *epoch* ke-1. Detail nilai performa dari setiap *input word vector* dapat dilihat pada Tabel 6.22.

Tabel 6.22 Pengaruh *Input Word Vector Subtask* C

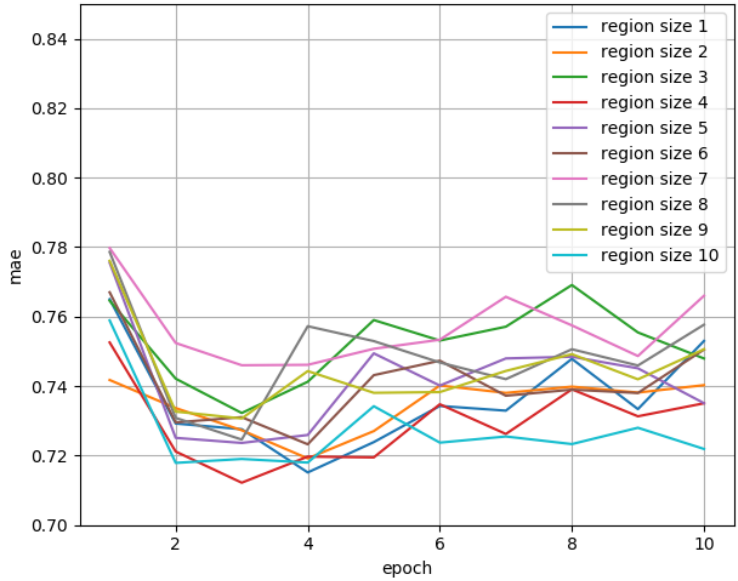
Model	***Embed Mode***	***MAE^M***	***Avg Recall***	***Acc***
Word2vec CBOW	*static*	0.740_1	0.392_3	0.650_3

Model	***Embed Mode***	***MAE^M***	***Avg Recall***	***Acc***
Word2vec CBOW	*non-static*	0.747_2	0.393_2	0.657_1
Word2Vec Skip Gram	*non-static*	0.748_3	0.397_1	0.652_2
Word2Vec Skip Gram	*static*	0.756_4	0.392_3	0.652_2
Bojanowski	*non-static*	0.766_5	0.386_4	0.640_4
Bojanowski	*static*	0.792_6	0.378_5	0.633_5

Berdasarkan Tabel 6.22 model *word2vec* CBOW *static* memiliki nilai *macro* MAE 0.740, paling rendah dibandingkan dengan model lain. Nilai *macro* MAE paling besar dimiliki model *fasttext* Bojanowski yaitu 0.792. Peringkat nilai *average recall* dan akurasi tidak sama dengan peringkat pada nilai *macro* MAE, nilai *average recall* tertinggi dimiliki oleh model *word2vec skip-gram non-static* yang merupakan nilai *macro* MAE tertinggi ketiga. Akurasi tertinggi dimiliki oleh model *word2vec* CBOW *non-static*. Selisih yang terlihat cukup besar antara kelompok model *word2vec* dan *fasstext* Bojanowski diakibatkan perbedaan jumlah kata yang tercakup oleh model. Perbedaan jumlah kata yang dicakup oleh model dapat dilihat pada Gambar 6.13 karena antara *subtask* C dan B memiliki jumlah data yang sama. Dari percobaan parameter *input word vector*, model yang paling optimal adalah *word2vec* CBOW *static* dengan nilai *macro* MAE paling rendah.

6.4.3.2 Pengaruh *Filter Region Size*

Percobaan *training* model CNN dilanjutkan dengan melakukan pengubahan terhadap parameter *region size*. Percobaan pertama diawali dengan menggunakan *single filter region size* 1-10.



Gambar 6.20 Grafik Pengaruh *Single Filter Region Size Subtask* C

Gambar 6.20 menunjukkan performa *training* model dari setiap *single filter region size*. Seluruh *region size* memiliki performa dengan pola yang hampir sama. *Region size* 10 merupakan yang paling optimal karena memiliki nilai *macro* MAE paling rendah pada beberapa *epoch* terutama pada *epoch* ke-10. Detail performa setiap *single filter region size* dapat dilihat pada Tabel 6.23.

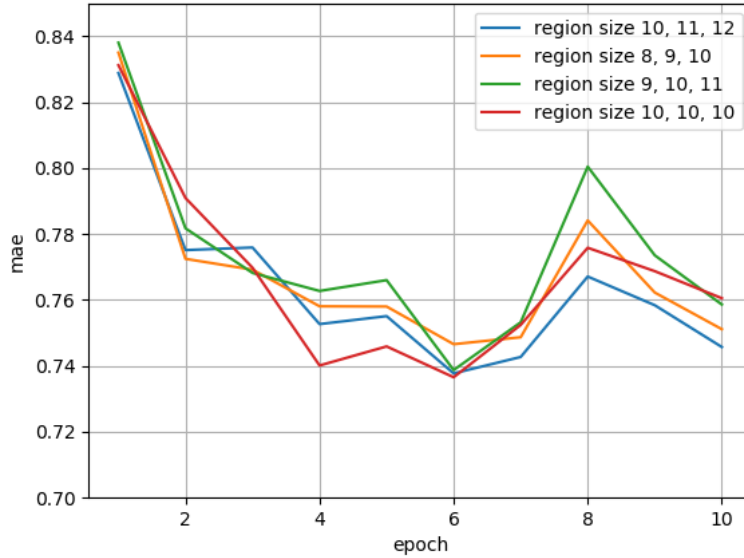
Tabel 6.23 Pengaruh *Single Filter Region Size Subtask* C

Region Size	***MAE^M***	***Avg Recall***	***Acc***
10	0.722_1	0.396_1	0.658_1
4	0.735_2	0.395_2	0.651_5
5	0.735_2	0.395_2	0.656_2
2	0.740_3	0.390_3	0.653_4
3	0.748_4	0.388_6	0.648_7
6	0.751_5	0.394_4	0.655_3

Region Size	MAE^M	*Avg Recall*	*Acc*
9	0.751_6	0.395_2	0.650_6
1	0.753_7	0.382_5	0.641_8
8	0.758_8	0.395_2	0.656_2
7	0.766_9	0.395_2	0.649_9

Berdasarkan Tabel 6.23 *region size* yang paling optimal adalah 10, karena memiliki nilai *macro* MAE paling rendah yaitu 0.722. Nilai *average recall* dan akurasi dari *region size* 10 juga merupakan yang paling tinggi dibandingkan dengan *region size* lain.

Langkah selanjutnya adalah melakukan percobaan *training* model dengan *multiple filter region size* dengan kombinasi 3 angka dari *single filter region size* terbaik. Karena keterbatasan perangkat lunak yang digunakan untuk *training*, konfigurasi awal untuk *feature maps* yang digunakan menjadi 200.



Gambar 6.21 Grafik Pengaruh *Multiple Filter Region Size Subtask* C

Gambar 6.21 menunjukkan performa dari setiap *multiple filter region size*. Seluruh *region size* memiliki performa dengan pola yang hampir sama pada setiap *epoch*. Terlihat bahwa *region size* (10, 10, 10) memiliki performa lebih optimal karena memiliki nilai *macro* MAE paling rendah pada *epoch* ke-10. Detail performa *multiple filter region size* dapat dilihat pada Tabel 6.24

Tabel 6.24 Pengaruh *Multiple Filter Region Size Subtask* C

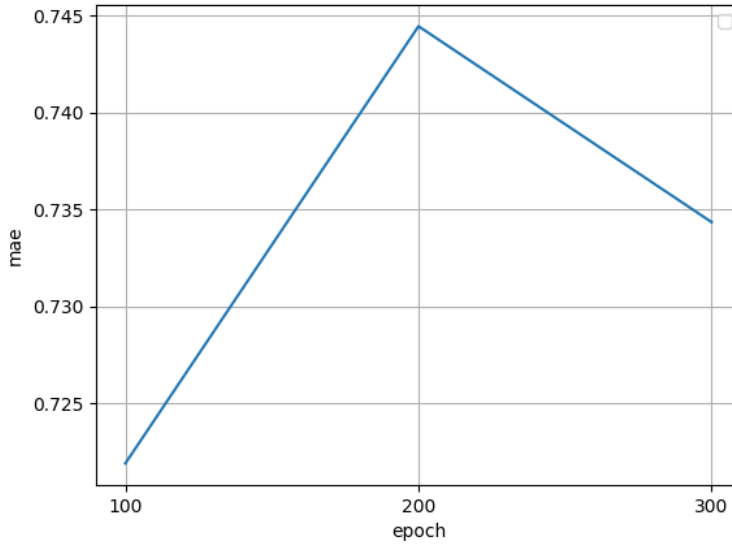
Region Size	**MAE**	***Avg Recall***	***Accuracy***
10,11,12	0.746_1	0.397_2	0.647_2
8,9,10	0.751_2	0.393_4	0.645_4
9,10,11	0.759_3	0.396_3	0.652_1
10,10,10	0.761_4	0.398_1	0.646_3

Berdasarkan Tabel 6.24 *region size* (10, 11, 12) memiliki nilai *macro* MAE paling rendah dibandingkan dengan *region size* lain yaitu 0.746. Nilai *average recall* dari *region size* (10, 11, 12) merupakan tertinggi kedua. Nilai *average recall* tertinggi dimiliki oleh *region size* (10, 10, 10) yang merupakan *region size* terendah berdasarkan nilai *macro* MAE. Peringkat nilai *average recall* dan akurasi pada seluruh *region size* tidak sama dengan peringkat dari nilai *macro* MAE. Berdasarkan hasil percobaan *training* dengan parameter *multiple filter region size*, nilai *macro* MAE-nya tidak lebih besar dari parameter *single region size*, sehingga *region size* yang paling optimal adalah 10. *Region size* 10 akan digunakan pada percobaan dengan skenario selanjutnya.

6.4.3.3 Pengaruh Nilai *Feature Maps* Setiap *Filter Region Size*

Skenario percobaan *training* model CNN pada *subtask* C dilanjutkan dengan mengubah ukuran *feature maps*. Karena

keterbatasan perangkat keras, nilai parameter yang digunakan sebagai *feature* maps adalah 100, 200 dan 300.



Gambar 6.22 Grafik Pengaruh *Feature Maps Subtask* C

Gambar 6.22 menunjukkan performa dari setiap *feature maps* yang digunakan. *Feature maps* yang paling optimal adalah 100, karena memiliki nilai *macro* MAE paling rendah daripada *feature maps* lain. Nilai *macro* MAE paling tinggi dimiliki oleh *feature maps* 200. Detail performa dari setiap *feature maps* dapat dilihat pada Tabel 6.25.

Tabel 6.25 Pengaruh *Feature Maps Subtask* C

Feature Maps	***MAE^M***	***Avg Recall***	***Accuracy***
100	0.722_1	0.396_1	0.658_1
300	0.734_2	0.393_2	0.652_3
200	0.744_3	0.392_3	0.655_2

Berdasarkan Tabel 6.25 *feature maps* 400 memiliki nilai *macro* MAE paling rendah mencapai 0.722. Nilai *average*

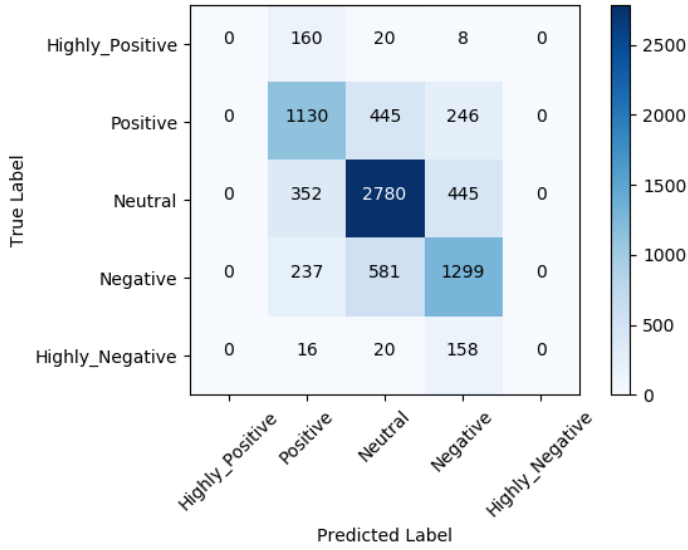
recall dan akurasi dari *feature maps* 100 juga merupakan yang paling tinggi dibandingkan *feature maps* lain. Namun jika dilihat pada hasil akurasi setiap *feature maps* semakin besar ukuran *feature maps* semakin besar nilai akurasinya.

Berdasarkan skenario percobaan yang dilakukan pada *subtask* C, nilai MAE paling optimal dapat dicapai hingga 0.722. Konfigurasi parameter terbaik pada *subtask* C dapat dilihat pada Tabel 6.26.

Tabel 6.26 Konfigurasi Parameter Paling Optimal *Subtask* C

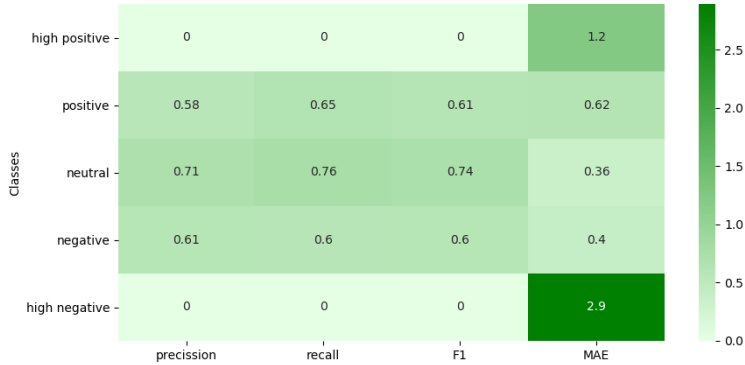
Input Word Vector	***Region Size***	***Feature Maps***
Word2Vec CBOW *static*	10	100

Dari seluruh percobaan *training* yang dilakukan, nilai *average recall* yang dihasilkan cukup rendah di antara 0.30 – 0.40, hal tersebut terjadi karena distribusi label kelas pada *subtask* C yang tidak seimbang.



Gambar 6.23 *Confusion Matrix Subtask* C

Gambar di atas merupakan *confusion matrix* dari model dengan konfigurasi parameter terbaik pada *subtask* C. Label *highly positive* dan *highly negative* tidak ada yang diprediksi secara tepat. Kesalahan prediksi yang cukup tinggi terjadi pada label *negative* yang diprediksi sebagai *neutral*, label *neutral* yang diprediksi sebagai *negative*, label *positive* yang diprediksi netral, dan label *neutral* yang diprediksi *positive*.



Gambar 6.24 Hasil Pengukuran Klasifikasi *Subtask* C

Berdasarkan Gambar 6.24 label *high negative* memiliki nilai MAE paling tinggi, kemudian *high positive* merupakan tertinggi ke-2. Nilai *precision*, *recall*, dan F_1 merupakan yang paling rendah hingga 0. Nilai MAE yang cukup tinggi pada label *high negative* dan *high positive* menunjukkan jika model klasifikasi CNN *subtask* C masih belum bisa mendeteksi secara optimal pada kelas tersebut. Distribusi jumlah label kelas *high negative* dan *high positive* juga mempengaruhi performa model klasifikasi.

6.4.4 Analisis Hasil Setiap *Subtask*

Berdasarkan performa yang diperoleh pada setiap *subtask*, model CNN memiliki performa yang lebih baik dibandingkan dengan model SVM dan *naïve bayes*.

Tabel 6.27 Hasil *Average Recall* Paling Optimal Setiap *Subtask*

Subtask	CNN	*Naïve Bayes*	SVM
A	0.820	0.780	0.777
B	0.691	0.634	0.633
C	0.396	0.363	0.392

Setiap *subtask* memiliki konfigurasi parameter paling optimalnya masing-masing. Konfigurasi parameter terbaik dari setiap *subtask* dapat dilihat pada Tabel 6.28.

Tabel 6.28 Rangkuman Konfigurasi Parameter Terbaik Setiap *Subtask*

Subtask	*Input Word Vectors*	*Region Size*	*Feature Maps*
A	*Word2Vec Skip-gram non-static*	3	100
B	*Word2Vec Skip-gram non-static*	3, 4, 5	100
C	*Word2Vec* CBOW *static*	10	100

Rangkuman performa terbaik yang diperoleh dari hasil percobaan *training* model CNN pada setiap *subtask* dapat dilihat pada Tabel 6.29.

Tabel 6.29 Rangkuman Performa Setiap *Subtask*

Subtask	*Avg Recall*	F_1	*Acc*
A	0.820	0.822	0.833
B	0.691	0.656	0.709
C	0.396	0.390	0.658

Subtask A dengan tugas klasifikasi 2 kelas memiliki performa paling baik dibandingkan dengan *subtask* B dan C.

Subtask A memiliki distribusi kelas data positif dan negatif yang seimbang. Hal tersebut menunjukkan jika model yang dihasilkan pada *subtask* A memiliki kemampuan yang cukup baik dalam melakukan klasifikasi sentimen positif dan negatif. Performa lebih rendah terjadi pada *subtask* B dan C, di mana jumlah kelas bertambah dan distribusi kelas data tidak seimbang. Pada *subtask* B data kelas netral lebih tinggi dibandingkan dengan data kelas positif dan negatif. *Subtask* C memiliki persamaan jumlah data dengan *subtask* B, namun terdapat pemisahan pada data kelas negatif ke kelas sangat negatif dan positif ke kelas sangat positif. *Subtask* C memiliki nilai *average recall* 0.396 karena data kelas sangat negatif dan sangat positif tidak ada yang diprediksi dengan benar oleh model. Hal tersebut menunjukkan jika model yang dihasilkan pada *subtask* C belum bisa secara optimal mendeteksi data yang memiliki sentimen sangat negatif atau sangat positif. Berdasarkan penjelasan tersebut distribusi kelas data dapat membantu mengoptimalkan performa dari model yang dihasilkan. Distribusi kelas data yang digunakan untuk *training* model klasifikasi dapat dilihat pada Tabel 6.1.

6.4.4.1 Uji Signifikansi Model

Uji signifikansi dilakukan untuk mengetahui tingkat perbedaan performa yang dihasilkan antara mode *non-static* dan *static* pada parameter *input word vectors*. *Training* model dilakukan kembali sebanyak 15 kali antara *non-static* dan *static* pada setiap *subtask*. Parameter yang digunakan adalah parameter terbaik dari setiap *subtask*. H_0 pengujian adalah tidak ada perbedaan performa yang dihasilkan antara *training* model menggunakan mode *non-static* maupun *static*. H_a pengujian adalah terdapat perbedaan performa yang dihasilkan antara *training* model menggunakan mode *non-static* maupun *static*. Nilai α atau tingkat signifikansinya adalah 0.05. kriteria penerimaan hipotesis adalah H_0 diterima ketika t hitung kurang dari atau sama dengan t tabel, dan H_0 ditolak ketika t hitung lebih besar dari t tabel.

Tabel 6.30 Akurasi Hasil *Training* 15 kali *Subtask* A

Percobaan	*Non-static*	*Static*
1	0.824	0.825
2	0.819	0.829
3	0.822	0.818
4	0.824	0.824
5	0.828	0.826
6	0.819	0.822
7	0.829	0.825
8	0.829	0.824
9	0.816	0.822
10	0.826	0.817
11	0.821	0.826
12	0.818	0.812
13	0.823	0.817
14	0.819	0.813
15	0.820	0.815

Nilai yang diukur pada *subtask* A adalah akurasi. Berdasarkan hasil percobaan 15 kali *training* pada *subtask* A perhitungan tingkat signifikansinya dapat dilihat pada Tabel 6.31.

Tabel 6.31 Perhitungan Nilai Signifikansi Subtask A

	non-static	*static*
Mean	0.82238709	0.820951
Variance	1.6755E-05	2.72E-05
Observations	15	15
Pooled Variance	2.19832E-05	
Hypothesized Mean Difference	0	
df	28	
t Stat	0.838967361	

P(T<=t) one-tail	0.204296735	
t Critical one-tail	1.701130934	
P(T<=t) two-tail	0.40859347	
t Critical two-tail	2.048407142	

Berdasarkan Tabel 6.31 nilai t hitung yaitu 0.838 kurang dari nilai t tabel 1.701 atau *p-value* yaitu 0.204 lebih besar dari nilai *alpha* 0.05, sehingga kesimpulan pada *subtask* A H_0 diterima, tidak ada perbedaan signifikan antara performa *non-static* dan *static*. Rata-rata performa akurasi *non-static* lebih baik daripada *static* namun tidak signifikan.

Tabel 6.32 Hasil *Training* 15 Kali *Subtask* B

Percobaan	non-static	static
1	0.685	0.682
2	0.688	0.685
3	0.684	0.683
4	0.683	0.682
5	0.682	0.680
6	0.685	0.684
7	0.689	0.681
8	0.684	0.680
9	0.679	0.684
10	0.687	0.682
11	0.679	0.688
12	0.680	0.690
13	0.676	0.683
14	0.675	0.685
15	0.681	0.686

Nilai yang diukur pada *subtask* B adalah *average recall.* Berdasarkan hasil percobaan 15 kali *training* pada *subtask* B

perhitungan tingkat signifikansinya dapat dilihat pada Tabel 6.33.

Tabel 6.33 Perhitungan Nilai Signifikansi Subtask B

	non-static	***static***
Mean	0.682385	0.683621
Variance	1.74E-05	7.27E-06
Observations	15	15
Pooled Variance	1.23E-05	
Hypothesized Mean Difference	0	
df	28	
t Stat	-0.96452	
P(T<=t) one-tail	0.17152	
t Critical one-tail	1.701131	
P(T<=t) two-tail	0.34304	
t Critical two-tail	2.048407	

Berdasarkan perhitungan tingkat signifikansi pada *subtask* B, nilai t hitung yaitu -0.964 kurang dari nilai t tabel 1.701 atau *p-value* 0.171 lebih besar dari nilai *alpha* 0.05, sehingga kesimpulan pada *subtask* B H_0 diterima, tidak ada perbedaan secara signifikan antara *non-static* dan *static*. Nilai rata-rata *average recall static* lebih baik daripada *non-static*.

Tabel 6.34 Hasil *Training* 15 Kali *Subtask* C

Percobaan	**non-static**	**static**
1	0.285	0.303
2	0.291	0.302
3	0.292	0.306
4	0.284	0.305
5	0.283	0.305
6	0.291	0.304
7	0.293	0.309

Percobaan	non-static	static
8	0.289	0.304
9	0.290	0.301
10	0.291	0.301
11	0.278	0.308
12	0.289	0.313
13	0.287	0.305
14	0.291	0.305
15	0.284	0.297

Nilai yang diukur pada *subtask* C adalah *macro* MAE. Berdasarkan hasil percobaan 15 kali *training* pada *subtask* C perhitungan tingkat signifikansinya dapat dilihat pada Tabel 6.35

Tabel 6.35 Perhitungan Nilai Signifikansi *Subtask* C

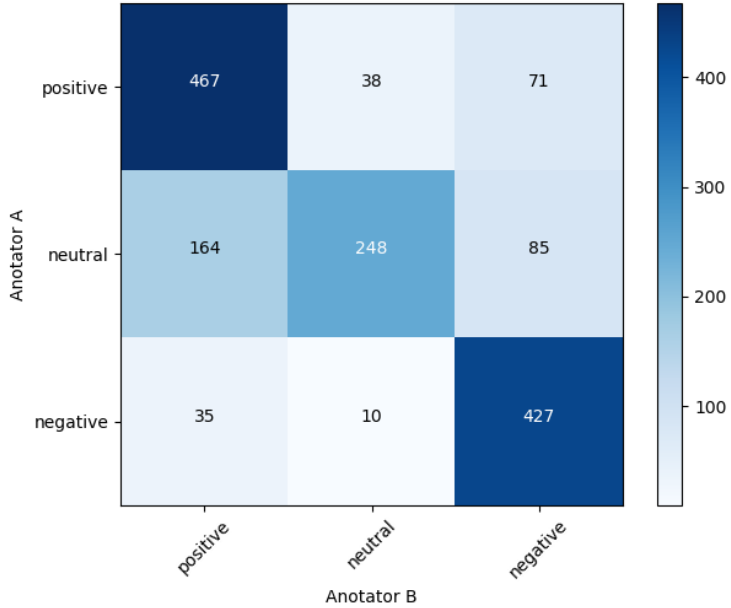
	non-static	***static***
Mean	0.287763	0.30452
Variance	1.7E-05	1.4E-05
Observations	15	15
Pooled Variance	1.55E-05	
Hypothesized Mean Difference	0	
df	28	
t Stat	-11.6435	
P(T<=t) one-tail	1.51E-12	
t Critical one-tail	1.701131	
P(T<=t) two-tail	3.02E-12	
t Critical two-tail	2.048407	

Berdasarkan perhitungan tingkat signifikansi pada *subtask* C, *p-value* 1.51E-12 lebih kecil dari nilai *alpha*, sehingga kesimpulan pada *subtask* C H_1 diterima, ada perbedaan signifikan antara performa *non-static* dan *static*. Nilai rata-rata

non-static lebih baik karena lebih rendah daripada nilai rata-rata *static*. Berdasarkan pengujian signifikansi pada 3 subtask, informasi yang didapatkan adalah bahwa ada perbedaan secara signifikan antara performa mode *static* dan *non-static* dalam penelitian ini. Perbedaan parameter terbaik terjadi pada *subtask* B dan C. Berdasarkan percobaan sebelumnya parameter terbaik dari *subtask* B adalah *non-static*, namun berdasarkan nilai rata-rata dari 15 kali percobaan *training*, *static* menjadi lebih baik daripada *non-static*. Hal serupa juga terjadi pada *subtask* C, dimana hasil percobaan *training* sebelumnya menyatakan jika *static* merupakan parameter terbaik, namun berdasarkan rata-rata nilai hasil dari 15 kali percobaan *non-static* lebih baik daripada *static*.

6.4.4.2 *Inter Anotator Agreement*

Hasil percobaan *training* model CNN pada setiap *subtask* dirasa penulis memiliki masalah yang perlu dianalisis. Salah satu permasalahan tersebut adalah data *tweet* berlabel yang digunakan selama proses *training*. Berdasarkan *confusion matrix* pada setiap *subtask*, kesalahan prediksi model terhadap label cukup tinggi. Pada *subtask* A kesalahan prediksi mencapai 16-17%, *subtask* B 30-32%, dan *subtask* C 42-100%. Dengan tingkat kesalahan prediksi tersebut data berlabel yang digunakan *training* dianalisis menggunakan penilaian koefisien *cohen's kappa*. Penilaian dilakukan untuk mengetahui tingkat persamaan persepsi antar anotator dalam melakukan pemberian label terhadap data. Informasi yang didapatkan adalah tingkat kesepakatan anotator terhadap label yang diberikan pada data *tweet*. Data *tweet* dengan studi kasus politik memiliki tingkat kesulitan dalam pemahaman isi teksnya, implikasi dari hal tersebut adalah tingkat perbedaan persepsi antar anotator menjadi cukup tinggi terhadap label yang diberikan. Jumlah kelas data yang dianalisis adalah 3 kelas yaitu positif, netral, dan negatif. Sampel data yang digunakan untuk perhitungan koefisien *cohen's kappa* adalah 1500 *tweet* yang sama dari 2 anotator.



Gambar 6.25 *Confusion Matrix* 3 kelas 2 Anotator

Gambar 6.25 menjelaskan tentang perbedaan persepsi terhadap label yang diberikan pada data *tweet*. Perbedaan pendapat yang sering terjadi adalah label netral-positif dan netral-negatif. Perbedaan pendapat juga terjadi pada label yang saling bertolak belakang yaitu positif dan negatif 6.8% data. Nilai koefisien *kappa* yang didapatkan adalah 0.606. Nilai koefisien *kappa* yang dihasilkan menunjukkan jika tingkat kesepakatan antar 2 anotator dikatakan "sedang" (0.41-0.60). Tingkat kesepakatan hasil penilaian koefisien *kappa* dapat menjadi salah satu penyebab eror yang terjadi pada *training* model CNN. Implikasi dari hal tersebut adalah data *tweet* aktual dengan labelnya menjadi berbeda dan tingkat keambiguan sentimen pada data *tweet* menjadi lebih tinggi.

Tabel 6.36 Sampel Data *Tweet* yang Ambigu

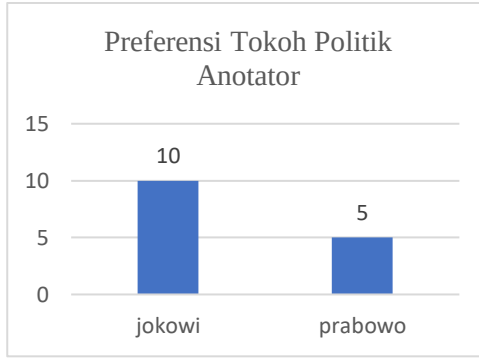
Tweet	A	B
@Yusrilihza_Mhd sy salut perjuangan yang dr dinyatakan tidak lolos ternyata ada persekongkolan yang dilakukan kpu, bisa dg penguasa, krn takut Anda pintar dalam dunia hukum. Yang nantinya bisa merepotkan mereka. Tapi sy mau muntah denger Anda baru start dah nawarin gab pdip,takut dibanding kpu lg?@rockygerung	Positif	Negatif
@mheru212azima @Gemacan70 @TsamaraDKI @PDIP_Online Dalam beberapa hal saya kurang sependapat dengan bang FH, tapi yang lebih keren narasinya dari bang FH kurasa belum ada. Belum ada yang lebih keren dari FH	Positif	Negatif
Salute dengan bang Faizal Assegaf! Bongkar semuanya! @fadlizon @prabowo pantas saja elektabilitas Prabowo dan Gerindra kian turun akibat ulah Fadli Zon!	Negatif	Positif
@JKFC23456789 @RizmaWidiono Wahai Bangsaku, mari tolak kebatilan jadikan gerakan #TolakPrabowo #TolakGerindra #TolakPKS gerakan nasional yang mengharubiru perasaan seluruh rakyat Indonesia	Positif	Negatif

Pemberian label juga dipengaruhi oleh preferensi politik dari anotator. Preferensi politik dapat menentukan apakah *tweet* bertolak belakang atau sama dengan penilaian anotator. Implikasi dari perbedaan preferensi politik dari anotator adalah perbedaan pemberian label yang bertolak belakang terhadap *tweet.*

Tabel 6.37 Preferensi Politik Anotator

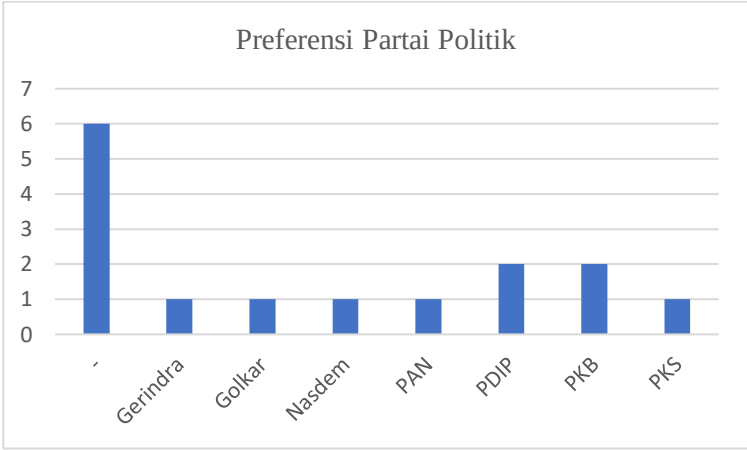
Anotator	Tokoh Politik	Partai Politik
1	Jokowi	PDIP
2	Jokowi	PKB
3	Prabowo	PKS
4	Jokowi	PDIP
5	Jokowi	-
6	Jokowi	PKB
7	Prabowo	-
8	Jokowi	Nasdem
9	Prabowo	Gerindra
10	Jokowi	-
11	Jokowi	-
12	Jokowi	Golkar
13	Jokowi	-
14	Prabowo	PAN
15	Prabowo	-

Berdasarkan Tabel 6.37 preferensi terhadap tokoh politik yang paling tinggi adalah Jokowi yaitu sekitar 67%. Kemudian 33% memiliki preferensi tokoh politik terhadap Prabowo.



Gambar 6.26 Preferensi Tokoh Politik

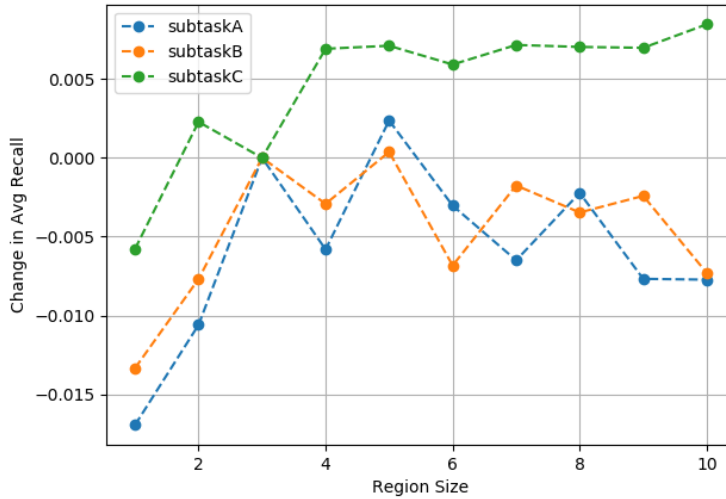
Preferensi terhadap partai politik yang paling tinggi adalah abstain atau tidak memiliki preferensi, 40% anotator tidak memiliki preferensi terhadap partai politik. 4 anotator memiliki preferensi terhadap Jokowi namun tidak dipengaruhi oleh partai politik. Pada penelitian ini partai yang memiliki preferensi paling tinggi adalah PDIP dan PKB.



Gambar 6.27 Preferensi Partai Politik

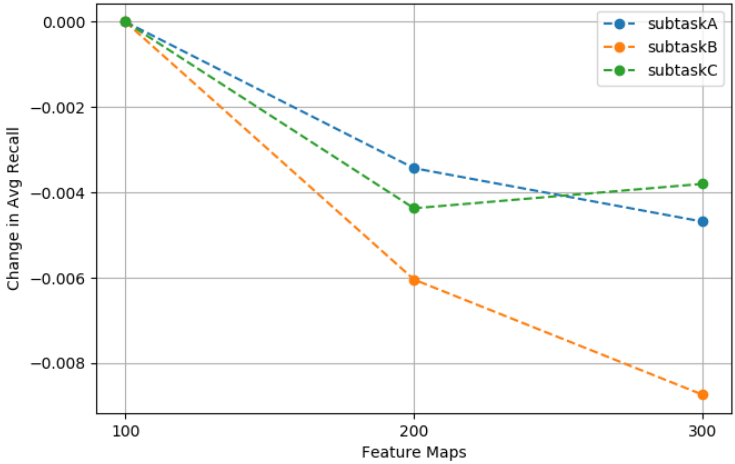
6.4.4.3 Pengaruh Perubahan Parameter

Analisis selanjutnya adalah melihat tingkat pengaruh dari parameter yang digunakan untuk percobaan *training* model CNN. Hal tersebut dilakukan dengan melihat perubahan yang dihasilkan dari setiap skenario. Pengukuran yang digunakan untuk melihat perubahan setiap skenario adalah *average recall*.



Gambar 6.28 Pengaruh *Region Size* pada Setiap *Subtask*

Gambar 6.28 menjelaskan perubahan *average recall* pada setiap *region size*, di mana *region size* 3 sebagai acuan perubahan. Grafik menunjukkan bahwa setiap *subtask* memiliki *region size* paling optimalnya masing-masing. *Subtask* A dan B memiliki *average recall* paling optimal pada *region size* 5, kemudian *subtask* C pada *region size* 10. Tren perubahan yang hampir sama dimiliki oleh *subtask* A dan *subtask* B. Pada *subtask* C semakin besar ukuran *region size* maka akan mengalami perubahan positif lebih besar.



Gambar 6.29 Pengaruh *Feature Maps* pada Setiap *Subtask*

Gambar 6.29 menunjukkan perubahan *average recall* setiap *feature maps*, di mana *feature maps* 100 sebagai acuan perubahan. Setiap *subtask* memiliki *feature maps* paling optimalnya pada ukuran 100. Setiap *subtask* mengalami perubahan negatif pada *feature maps* 200 dan 300. Perubahan negatif paling tinggi terjadi pada *subtask* B. Berdasarkan perubahan nilai *average recall* pada seluruh *subtask* ternyata setiap parameter *region size* dan *feature maps* memiliki pengaruh terhadap performa model yang dihasilkan.

BAB VII
KESIMPULAN DAN SARAN

Pada bab ini akan dijelaskan mengenai kesimpulan dari semua proses penelitian tugas akhir yang dilakukan dan saran yang dapat diberikan untuk pengembangan yang lebih baik.

7.1 Kesimpulan

Kesimpulan yang didapatkan dari proses pengerjaan penelitian tugas akhir yang telah dilakukan antara lain:

1. Pengambilan data menggunakan metode *search* API pada media sosial *twitter* terbukti lebih baik untuk mendapatkan data dalam jumlah besar dengan waktu yang lebih singkat daripada metode *stream*. Penggunaan logika perulangan untuk pembaruan *max id* pada kode program juga membantu proses pengambilan data, karena dapat mengatasi masalah batasan data yang dapat diambil pada *twitter* API.
2. Penghapusan kumpulan data *twitter* yang terduplikasi perlu dilakukan pada pra-pemrosesan data, terutama untuk kumpulan data berlabel. Hal tersebut sangat berguna untuk mengurangi perbedaan label yang berkontradiksi pada data yang sama. Secara sederhana untuk mengurangi inkonsistensi data.
3. *Filtering* bahasa menggunakan *library langdetect* untuk mendapatkan data *twitter* berbahasa Indonesia memiliki performa yang sangat baik. Presisi filter bahasa yang dihasilkan sebesar 99.5 %.
4. Model *word2vec* yang dihasilkan dari *training* model menggunakan kumpulan data *twitter* berbahasa Indonesia memiliki performa paling baik pada proses pembuatan model CNN. Hal ini dikarenakan jumlah kata yang tercakup oleh model *word2vec* lebih banyak daripada model *fasstext* Bojanowski.
5. Dari berbagai percobaan *training* model CNN dengan berbagai skenario parameter, didapatkan parameter yang menghasilkan performa paling optimal pada penelitian ini. Parameter dengan performa paling optimal pada *subtask* A

adalah *input word vector* menggunakan model *word2vec skip-gram* dengan mode *non-static*, *region size* (3), dan *feature maps* 100 pada setiap *region size*. Parameter dengan performa paling optimal pada *subtask* B adalah *input word vector* menggunakan model *word2vec skip-gram* dengan mode *non-static*, *region size* (3, 4, 5), dan *feature maps* 100 pada setiap *region size*. Kemudian parameter dengan performa paling optimal pada *subtask* C adalah *input word vector* menggunakan model *word2vec skip-gram* dengan mode *non-static*, *region size* (10), dan *feature maps* 100 pada setiap *region size* .

6. Klasifikasi teks menggunakan CNN memiliki performa lebih baik dibandingkan dengan SVM dan *naive bayes*. Akurasi yang dicapai oleh model CNN masing-masing pada setiap *subtask* adalah 0.833 pada *subtask* A, 0.709 pada *subtask* B, dan 0.658 pada *subtask* C.

7.2 Saran

Berdasarkan penelitian tugas akhir ini, adapun beberapa saran untuk pengembangan penelitian ke depan yaitu sebagai berikut:

1. Perlu dikembangkan kembali metode penghapusan data terduplikasi yang mampu mendeteksi data dengan kemiripan konten teks. Penelitian ini hanya mampu melakukan deteksi pada data yang memiliki kesamaan seluruh karakter teksnya, ketika ada perbedaan 1 karakter atau lebih walaupun memiliki konten teks yang sama tidak akan terdeteksi sebagai duplikasi.
2. Aplikasi pemberian label atau anotasi data *twitter* menggunakan tampilan antar muka yang mudah untuk dipahami oleh anotator, dapat menggunakan tampilan antar muka yang mirip dengan aplikasi *twitter* asli untuk memudahkan anotator dalam memahami isi *tweet*.
3. Menggunakan standar bahasa Indonesia dalam menentukan sebuah teks adalah sebuah opini bernilai

negatif, netral, dan positif. Kemudian menjadi panduan oleh anotator dalam proses anotasi data.

4. Melakukan percobaan pada *training* model CNN dengan menambah beberapa skenario perubahan parameter di antaranya menambah kombinasi *filter region size* yang digunakan, menambah jumlah *feature maps*, mengubah parameter pada fungsi aktivasi, dan *pooling strategies*.
5. Jumlah percobaan *training* yang dilakukan pada setiap skenario sebaiknya lebih dari 1 kali. Hal tersebut untuk mengakomodasi perubahan nilai yang dihasilkan dari setiap proses *training* dengan mengalkulasi nilai minimal, maksimal dan rata-rata.

Halaman ini sengaja dikosongkan

DAFTAR PUSTAKA

[1] "DATA TERKINI, Jumlah Penduduk Indonesia Lebih dari 262 Juta Jiwa - Tribun Jateng," 2017. [Online]. Available: http://jateng.tribunnews.com/2017/08/02/data-terkini-jumlah-penduduk-indonesia-lebih-dari-262-juta-jiwa. [Accessed: 04-Mar-2018].

[2] Asosiasi Penyelenggara Jasa Internet Indonesia - APJII, "Penetrasi & Perilaku Pengguna Internet Indonesia - Survey 2017," p. 34, 2017.

[3] "Kementerian Komunikasi dan Informatika." [Online]. Available: https://kominfo.go.id/index.php/content/detail/4286/Pengguna+Internet+Indonesia+Nomor+Enam+Dunia/0/sorotan_media. [Accessed: 04-Mar-2018].

[4] "132 Juta Pengguna Internet Indonesia, 40% Penggila Medsos." [Online]. Available: https://inet.detik.com/cyberlife/d-3659956/132-juta-pengguna-internet-indonesia-40-penggila-medsos. [Accessed: 04-Mar-2018].

[5] J. Chavez, "# Fail : The Misuse of Social Media in the 2012 US Presidential Campaign," no. May, 2012.

[6] F. Anshari, "Komunikasi Politik di Era Media Sosial," *J. Komun.*, vol. 8, no. 1, pp. 91–102, 2013.

[7] Y. Kim, "Convolutional Neural Networks for Sentence Classification," 2014.

[8] T. Chen, R. Xu, Y. He, and X. Wang, "Improving sentiment analysis via sentence type classification using BiLSTM-CRF and CNN," *Expert Syst. Appl.*, vol. 72, pp. 221–230, 2017.

[9] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," pp. 1–12, 2013.

[10] M. Bonzanini, *Mastering Social Media Mining with Python*, 1st ed. Packt Publishing, 2016.

[11] A. M. Kaplan and M. Haenlein, “Users of the world, unite! The challenges and opportunities of Social Media,” *Bus. Horiz.*, vol. 53, no. 1, pp. 59–68, 2010.

[12] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*. 2012.

[13] R. Zafarani, M. A. Abbasi, and H. Liu, “Social media mining: An introduction,” *Soc. Media Min. An Introd.*, vol. 9781107018, pp. 1–320, 2014.

[14] B. Liu, *Web Data Mining*, 2nd ed. Springer, 2011.

[15] K. Dave, K. Dave, S. Lawrence, S. Lawrence, D. M. Pennock, and D. M. Pennock, “Mining the peanut gallery: Opinion extraction and semantic classification of product reviews,” *Proc. 12th Int. Conf. World Wide Web*, pp. 519–528, 2003.

[16] N. Jindal, B. Liu, and L. Bing, “Mining comparative sentences and relations,” *Proc. 29th Annu. Int. ACM SIGIR Conf. Res. Dev. Inf. Retr. - SIGIR ’06*, vol. 21, no. 2, pp. 1331–1336, 2006.

[17] B. Liu, “Sentiment Analysis and Subjectivity,” *Handb. Nat. Lang. Process.*, no. 1, pp. 1–38, 2010.

[18] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. J. Miller, “Introduction to wordnet: An on-line lexical database,” *Int. J. Lexicogr.*, vol. 3, no. 4, pp. 235–244, 1990.

[19] A. Copestack, “Natural Language Processing,” *Nat. Lang. Process.*, pp. 2003–2004, 2004.

[20] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, “Natural Language Processing (almost) from Scratch,” 2011.

[21] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[22] J. Turian, L. Ratinov, Y. Bengio, and J. Turian, “Word Representations: A Simple and General Method for Semi-supervised Learning,” *Acm*, no. July, pp. 384–394, 2010.

[23] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “5021-

Distributed-Representations-of-Words-and-Phrases-and-Their-Compositionality," pp. 1–9.

[24] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio, "Object Recognition with Gradient-Based Learning," no. 0, pp. 319–345, 1999.

[25] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A Convolutional Neural Network for Modelling Sentences," *Proc. ACL*, pp. 655–665, 2014.

[26] Y. Zhang and B. Wallace, "A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification," 2015.

[27] *Dasar-Dasar Ilmu Politik*, 1st ed. Jakarta: Gramedia, 2006.

Halaman ini sengaja dikosongkan

BIODATA PENULIS

Penulis bernama lengkap Prasetyo Wahyu Adi Wianto, lahir di Madiun pada tanggal 4 Mei 1996. Merupakan anak pertama dari 2 bersaudara. Penulis telah menempuh beberapa pendidikan formal yaitu: SD Negeri Barata Jaya Surabaya, SMP Negeri 12 Surabaya, dan SMA Negeri 16 Surabaya.

Pada tahun 2014 pasca kelulusan SMA, penulis melanjutkan pendidikan dengan jalur SNMPTN di Jurusan Sistem Informasi FTIK – Institut Teknologi Sepuluh Nopember (ITS) Surabaya dan terdaftar sebagai mahasiswa dengan NRP 5214100030. Selama menjadi mahasiswa, penulis mengikuti berbagai kegiatan kemahasiswaan diantaranya beberapa kepanitiaan seperti Information System Expo, dan ITS Mengajar serta pernah menjadi staf Sosial Masyarakat Himpunan Mahasiswa Sistem Informasi ITS pada tahun kedua, dan BSO IECC Badan Ekskutif Mahasiswa ITS pada tahun ke tiga. Selain itu, kegiatan seperti Latihan Ketrampilan Manajemen Mahasiswa pun pernah diikuti hingga Tingkat Menengah. Di bidang akademik, penulis aktif menjadi asisten dosen pada beberapa mata kuliah seperti Pengantar Sistem Operasi, Perencanaan Sumber Daya Perusahaan dan Sistem Cerdas.

Penulis memiliki ketertarikan di bidang Akuisisi Data dan Diseminasi Informasi (ADDI), sehingga memilih topik *deep learning for* NLP sebagai tugas akhir. Penulis dapat dihubungi melalui *email* di prasetyowahyu35@gmail.com.

Halaman ini sengaja dikosongkan

LAMPIRAN A

A-1. Data hasil Percobaan *Training* Model CNN Menggunakan Parameter *Input Word Vectors*.

Sub	Model	Md	Akurasi / epoch									
			1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
A	word2vec Skip-gram	non-static	0.803	0.816	0.821	0.814	0.831	0.825	0.828	0.834	0.832	0.832
A	word2vec Skip-gram	static	0.806	0.819	0.819	0.800	0.828	0.825	0.823	0.831	0.827	0.830
A	word2vec CBOW	non-static	0.791	0.814	0.815	0.802	0.809	0.804	0.810	0.822	0.817	0.813
A	word2vec CBOW	static	0.789	0.808	0.810	0.800	0.813	0.808	0.814	0.820	0.815	0.813
A	Bojanowski	non-static	0.704	0.775	0.783	0.743	0.729	0.684	0.790	0.811	0.773	0.804
A	Bojanowski	static	0.711	0.757	0.785	0.719	0.684	0.606	0.779	0.781	0.734	0.789
B	word2vec Skip-gram	non-static	0.692	0.686	0.711	0.709	0.705	0.705	0.693	0.706	0.712	0.709

Sub	Model	Md	Akurasi / epoch									
			1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
B	word2vec Skip-gram	static	0.683	0.699	0.711	0.687	0.682	0.696	0.692	0.695	0.697	0.689
B	word2vec CBOW	non-static	0.684	0.695	0.711	0.707	0.686	0.706	0.700	0.705	0.703	0.698
B	word2vec CBOW	static	0.678	0.694	0.712	0.706	0.678	0.706	0.696	0.699	0.695	0.697
B	Bojanowski	non-static	0.630	0.646	0.680	0.669	0.648	0.676	0.670	0.676	0.676	0.672
B	Bojanowski	static	0.625	0.656	0.665	0.636	0.647	0.662	0.683	0.673	0.663	0.687
C	word2vec Skip-gram	non-static	0.635	0.621	0.630	0.634	0.638	0.654	0.655	0.655	0.658	0.652
C	word2vec Skip-gram	static	0.638	0.617	0.615	0.626	0.623	0.648	0.642	0.648	0.656	0.645
C	word2vec CBOW	non-static	0.635	0.624	0.629	0.636	0.641	0.665	0.649	0.667	0.664	0.654
C	word2vec CBOW	static	0.629	0.613	0.617	0.608	0.627	0.650	0.618	0.657	0.656	0.646

Sub	Model	Md	Akurasi / epoch									
			1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
C	Bojanowski	non-static	0.583	0.583	0.597	0.609	0.636	0.636	0.586	0.631	0.650	0.635
C	Bojanowski	static	0.581	0.584	0.596	0.610	0.622	0.610	0.623	0.599	0.615	0.636

LAMPIRAN B

B-1. Data hasil Percobaan *Training* Model CNN Subtask A Menggunakan Parameter *Region Size*

Subtask	Region Size	Akurasi / epoch									
		1	2	3	4	5	6	7	8	9	10
A	1	0.796	0.797	0.820	0.811	0.808	0.810	0.810	0.818	0.812	0.811
A	2	0.788	0.794	0.815	0.803	0.807	0.801	0.805	0.816	0.813	0.818
A	3	0.801	0.808	0.822	0.818	0.819	0.831	0.831	0.833	0.832	0.833
A	4	0.796	0.805	0.814	0.818	0.808	0.826	0.827	0.829	0.826	0.826
A	5	0.788	0.798	0.807	0.816	0.810	0.824	0.827	0.829	0.831	0.832
A	6	0.784	0.796	0.805	0.804	0.800	0.822	0.823	0.826	0.821	0.822
A	7	0.779	0.798	0.799	0.809	0.802	0.824	0.824	0.829	0.824	0.825
A	8	0.792	0.802	0.816	0.827	0.819	0.832	0.832	0.837	0.834	0.833
A	9	0.786	0.798	0.797	0.808	0.806	0.815	0.816	0.818	0.817	0.820
A	10	0.787	0.799	0.804	0.818	0.814	0.819	0.824	0.824	0.822	0.822
A	3,4,5	0.803	0.816	0.821	0.814	0.831	0.825	0.828	0.834	0.832	0.832
A	2,3,4	0.791	0.813	0.785	0.819	0.823	0.821	0.824	0.828	0.826	0.827

Subtask	Region Size	Akurasi / epoch									
		1	2	3	4	5	6	7	8	9	10
A	1,2,3	0.800	0.806	0.782	0.820	0.824	0.818	0.822	0.826	0.821	0.820
A	3,3,3	0.796	0.812	0.794	0.824	0.833	0.819	0.825	0.830	0.829	0.827

B-2. Data hasil Percobaan *Training* Model CNN Subtask B Menggunakan Parameter *Region Size*

Subtask	Region Size	Avg Recall / epoch									
		1	2	3	4	5	6	7	8	9	10
B	1	0.639	0.686	0.681	0.689	0.670	0.678	0.675	0.680	0.669	0.671
B	2	0.652	0.679	0.675	0.683	0.660	0.666	0.679	0.675	0.676	0.677
B	3	0.650	0.687	0.675	0.689	0.666	0.676	0.678	0.686	0.684	0.685
B	4	0.645	0.681	0.660	0.685	0.668	0.669	0.684	0.684	0.684	0.682
B	5	0.651	0.684	0.663	0.685	0.669	0.678	0.681	0.687	0.683	0.685
B	6	0.641	0.672	0.654	0.664	0.668	0.666	0.670	0.676	0.675	0.678
B	7	0.653	0.681	0.664	0.684	0.680	0.684	0.682	0.684	0.687	0.683
B	8	0.645	0.672	0.651	0.680	0.675	0.677	0.677	0.683	0.679	0.681

Subtask	Region Size	Avg Recall / epoch									
		1	2	3	4	5	6	7	8	9	10
B	9	0.647	0.673	0.658	0.681	0.677	0.679	0.674	0.677	0.681	0.682
B	10	0.641	0.676	0.658	0.678	0.678	0.673	0.674	0.674	0.676	0.677
B	5,6,7	0.674	0.652	0.651	0.683	0.686	0.680	0.689	0.681	0.665	0.683
B	4,5,6	0.669	0.669	0.639	0.670	0.669	0.671	0.665	0.673	0.672	0.673
B	3,4,5	0.676	0.677	0.657	0.681	0.689	0.691	0.690	0.690	0.691	0.691
B	5,5,5	0.674	0.678	0.651	0.669	0.682	0.680	0.681	0.683	0.683	0.680

B-3. Data hasil Percobaan *Training* Model CNN Subtask C Menggunakan Parameter *Region Size*

Subtask	Region Size	Avg Recall / epoch									
		1	2	3	4	5	6	7	8	9	10
C	1	0.378	0.400	0.400	0.404	0.390	0.388	0.392	0.379	0.383	0.382
C	2	0.392	0.400	0.404	0.405	0.394	0.392	0.394	0.386	0.390	0.390
C	3	0.387	0.401	0.402	0.403	0.388	0.386	0.389	0.382	0.385	0.388
C	4	0.388	0.399	0.401	0.405	0.394	0.391	0.391	0.390	0.394	0.395

Subtask	Region Size	Avg Recall / epoch									
		1	2	3	4	5	6	7	8	9	10
C	5	0.390	0.399	0.399	0.402	0.392	0.391	0.392	0.389	0.394	0.395
C	6	0.386	0.400	0.403	0.402	0.393	0.387	0.394	0.390	0.392	0.394
C	7	0.389	0.397	0.399	0.401	0.396	0.394	0.398	0.394	0.398	0.395
C	8	0.384	0.398	0.399	0.399	0.394	0.396	0.396	0.390	0.398	0.395
C	9	0.390	0.396	0.398	0.395	0.390	0.391	0.397	0.394	0.395	0.395
C	10	0.391	0.397	0.402	0.401	0.393	0.395	0.395	0.394	0.400	0.396
C	10,11,12	0.349	0.382	0.378	0.388	0.394	0.400	0.398	0.379	0.386	0.397
C	9,10,11	0.346	0.378	0.382	0.392	0.389	0.399	0.398	0.375	0.384	0.396
C	8,9,10	0.342	0.382	0.383	0.391	0.387	0.394	0.394	0.374	0.380	0.393
C	10,10,10	0.341	0.375	0.382	0.395	0.390	0.402	0.401	0.378	0.385	0.398

LAMPIRAN C

C-1. Data hasil Percobaan *Training* Model CNN Setiap Subtask Menggunakan Parameter *Feature Maps*

Subtask	Region Size	Akurasi / epoch									
		1	2	3	4	5	6	7	8	9	10
A	100	0.801	0.808	0.822	0.818	0.819	0.831	0.831	0.833	0.832	0.833
A	200	0.806	0.820	0.800	0.819	0.819	0.824	0.829	0.830	0.824	0.824
A	300	0.806	0.807	0.794	0.817	0.815	0.825	0.824	0.827	0.824	0.823
B	100	0.692	0.686	0.711	0.709	0.705	0.705	0.693	0.706	0.712	0.709
B	200	0.694	0.686	0.705	0.713	0.708	0.710	0.688	0.706	0.715	0.710
B	300	0.681	0.679	0.700	0.708	0.706	0.705	0.680	0.707	0.712	0.707
C	100	0.638	0.659	0.650	0.661	0.641	0.653	0.658	0.656	0.661	0.658
C	200	0.634	0.638	0.644	0.640	0.657	0.660	0.658	0.664	0.653	0.655
C	300	0.620	0.634	0.628	0.643	0.657	0.655	0.654	0.662	0.645	0.652