



**TUGAS AKHIR - KS141501**

**ANALISIS SENTIMEN TEKS BAHASA INDONESIA PADA MEDIA  
SOSIAL MENGGUNAKAN ALGORITMA CONVOLUTIONAL  
NEURAL NETWORK  
( STUDI KASUS : OPERATOR TELEKOMUNIKASI )**

**SENTIMENT ANALYSIS FOR INDONESIAN SOCIAL MEDIA'S  
TEXT USING CONVOLUTIONAL NEURAL NETWORK  
ALGORITHM  
( STUDY CASE : TELECOMMUNICATION OPERATOR )**

**ADRIAN AFNANDIKA  
NRP 05211440000134**

**Dosen Pembimbing:  
Renny Pradina , S.T., M.T.**

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



**ITS**  
Institut  
Teknologi  
Sepuluh Nopember

**TUGAS AKHIR - KS141501**

**ANALISIS SENTIMEN TEKS BAHASA INDONESIA PADA MEDIA  
SOSIAL MENGGUNAKAN ALGORITMA CONVOLUTIONAL  
NEURAL NETWORK  
( STUDI KASUS : OPERATOR TELEKOMUNIKASI )**

**ADRIAN AFNANDIKA**  
NRP 05211440000134

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

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



**ITS**  
Institut  
Teknologi  
Sepuluh Nopember

**FINAL PROJECT - KS141501**

**SENTIMENT ANALYSIS FOR INDONESIAN SOCIAL MEDIA'S  
TEXT USING CONVOLUTIONAL NEURAL NETWORK  
ALGORITHM  
( STUDY CASE : TELECOMMUNICATION OPERATOR )**

**ADRIAN AFNANDIKA**  
NRP 05214110000134

Supervisor:  
Renny Pradina , S.T., M.T. SCJP

**INFORMATION SYSTEMS DEPARTMENT**  
Information and Communication Technology Faculty  
Sepuluh Nopember Institute of Technology  
Surabaya 2018

**LEMBAR PENGESAHAN**

**ANALISIS SENTIMEN TEKS BAHASA  
INDONESIA PADA MEDIA SOSIAL  
MENGUNAKAN ALGORITMA  
CONVOLUTIONAL NEURAL NETWORK  
( STUDI KASUS : OPERATOR  
TELEKOMUNIKASI )**

**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:

**ADRIAN AFNANDIKA**

**0521 14 4000 0134**

Surabaya, Juli 2018

**KEPALA  
DEPARTEMEN SISTEM INFORMASI**

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



## LEMBAR PERSETUJUAN

### ANALISIS SENTIMEN TEKS BAHASA INDONESIA PADA MEDIA SOSIAL MENGUNAKAN ALGORITMA CONVOLUTIONAL NEURAL NETWORK ( STUDI KASUS : OPERATOR TELEKOMUNIKASI )

#### 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

ADRIAN AFNANDIKA


0521 14 4000 0134

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


Renny Pradina, S. T., M. T. Scjp

  
(Pembimbing 1)

Nur Aini R., S.Kom, M.Sc.Eng., Ph.D

  
(Penguji 1)

Radityo Prasetyanto W., S.Kom, M.Kom.

  
(Penguji 2)



**ANALISIS SENTIMEN TEKS BAHASA INDONESIA  
PADA MEDIA SOSIAL MENGGUNAKAN ALGORITMA  
CONVOLUTIONAL NEURAL NETWORK  
( STUDI KASUS : OPERATOR TELEKOMUNIKASI )**

**Nama Mahasiswa** : Adrian Afnandika  
**NRP** : 05211440000134  
**Departemen** : Sistem Informasi  
**Pembimbing 1** : Renny Pradina, S.T., M. T.

**ABSTRAK**

*Dewasa ini media sosial merupakan salah satu media untuk membagikan informasi secara cepat. Sebagian besar informasi yang tersebar dalam media sosial dapat berupa pendapat individu terhadap sebuah objek tertentu yang disebut sebuah sentimen. Umumnya terdapat dua macam sentimen yaitu sentimen positif maupun negatif. Hal ini dapat dimanfaatkan untuk mendapatkan insight terkait objek tersebut. Industri telekomunikasi saat ini semakin berkembang di Indonesia dimana memiliki banyak pengguna. Tidak sedikit pengguna tersebut mengungkapkan pendapatnya terkait layanan atau produk dari operator telekomunikasi di indonesia.*

*Dari fenomena tersebut analisis sentimen dapat dilakukan untuk mendapatkan insight dari objek yang akan di analisis. Namun dalam penerapannya analisis sentimen membutuhkan algoritma yang dapat melakukan klasifikasi pendapat. Dalam penelitian sebelumnya membandingkan beberapa algoritma untuk melakukan analisis sentimen dalam berbahasa inggris.*

*CNN (Convolutional Neural Network) merupakan algoritma yang memiliki akurasi terbaik dibandingkan algoritma lain, sehingga dalam penelitian ini menggunakan algoritma tersebut untuk melakukan analisis sentimen dalam berbahasa indonesia. Dalam penerapannya CNN membutuhkan input vektor kata agar dapat ditraining menjadi suatu model, sehingga dalam*

penelitian ini menggunakan dua macam library yaitu word2vec dan fasttext.

Data yang digunakan berjumlah 11.659, dengan pembagian label data sebagai berikut, label sangat positif adalah 2.310 tweets, label positif 2.352 tweets, label netral 2.328 tweets, label negatif 2.344 tweets dan label sangat negatif 2.325 tweets. Hasil terbaik merupakan CNN dengan label positif dan negatif dengan tingkat akurasi 96.80 %. Filter region size sangat mempengaruhi untuk meningkatkan akurasi model. Selain itu penggunaan learning algorithm word embedding memiliki pengaruh besar terhadap akurasi model. Penggunaan parameter model dengan tepat dapat meningkatkan akurasi hingga 11.07 %.

**Kata Kunci:** Analisis Sentimen, Klasifikasi, Convolutional Neural Network, Industri Telekomunikasi.



**ANALISIS SENTIMEN TEKS BAHASA INDONESIA  
PADA MEDIA SOSIAL MENGGUNAKAN ALGORITMA  
CONVOLUTIONAL NEURAL NETWORK  
( STUDI KASUS : OPERATOR TELEKOMUNIKASI )**

**Nama Mahasiswa** : Adrian Afnandika  
**NRP** : 05211440000134  
**Departemen** : Sistem Informasi  
**Pembimbing 1** : Renny Pradina, S.T., M. T.

**ABSTRACT**

*Today social media is one of the media to share information quickly. Most of the information spread in social media can be an individual's opinion of a particular object called a sentiment. There are two kinds of sentiments that are positive and negative sentiments. It can be used to get insight related to the object. The telecommunication industry is now growing in Indonesia where has many users. Not a few users are expressing opinions related to services or products from telecom operators in Indonesia.*

*From that phenomenom, sentiment analysis can be done to get the insight of the object to be in the analysis. However, in the application of sentiment analysis requires an algorithm that can classify opinions. In a previous study comparing several algorithms to perform sentiment analysis in English.*

*CNN (Convolutional Neural Network) is an algorithm that has the best accuracy compared to other algorithms, so in this study using the algorithm to perform sentiment analysis in Indonesian language. In implementing CNN requires word vector input to be trained into a model, so in this study using two kinds of libraries namely word2vec and fasttext..*

*The data used amounted to 11,659, with the data label distribution as follows, the very positive label is 2,310 tweets, positive label 2,352 tweets, neutral label 2,328 tweets, negative label 2,344 tweets and very negative label 2,325 tweets. The best*

*result is CNN with positive and negative label with 96.80% accuracy. The filter region size greatly influences to improve model accuracy. In addition, the use of learning algorithm word embedding has a major influence on the accuracy of the model. Proper use of model parameters can increase accuracy by up to 11.07%.*

***Keywords: Sentiment Analysis, Classification, Convolutional Neural Network, Telecommunication Industry.***

## KATA PENGANTAR

Puji dan syukur penulis tuturkan ke hadirat Allah SWT, Tuhan Semesta Alam yang telah memberikan kekuatan dan hidayah-Nya kepada penulis sehingga penulis mendapatkan kelancaran dalam menyelesaikan tugas akhir ini yang merupakan salah satu syarat kelulusan pada Departemen Sistem Informasi, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember Surabaya. Terima kasih penulis sampaikan kepada pihak-pihak yang telah mendukung demi tercapainya tujuan pembuatan tugas akhir ini. Secara khusus penulis akan menyampaikan ucapan terima kasih yang sebanyak-banyaknya kepada:

1. Allah SWT selaku Tuhan semesta alam, karena berkat rahmat-Nya penelitian ini dapat terselesaikan.
2. Bapak Heri Santoso dan Ibu Tri Candra Setiawati selaku kedua orang tua Adrian Afandika, Andrias Alfariski selaku saudara kandung dari penulis yang tiada henti memberikan dukungan dan semangat
3. Ibu Renny Pradina K., S.T., M.T., SCJP, selaku dosen pembimbing dan sebagai narasumber yang senantiasa meluangkan waktu, memberikan ilmu dan petunjuk, serta memotivasi untuk kelancaran tugas akhir.
4. Bapak Faisal johan Atletiko, S.Kom, M.Kom dan Radityo Prasetyanto W., S.Kom., M.Kom., selaku dosen penguji yang telah memberikan saran dan kritik untuk perbaikan tugas akhir.
5. Seluruh dosen Jurusan Sistem Informasi ITS yang telah memberikan ilmu yang bermanfaat kepada penulis.
6. Maulita Arumningtyas selaku teman dekat penulis dari awal perkuliahan, yang telah membantu penulis dalam segala hal baik perkuliahan dan diluar perkuliahan.
7. Alim, Calvin, Pras, Endar selaku sahabat penulis yang telah menemani penulis menikmati indahny alam indonesia.

8. Fandhi, Joni, Umar, Imad dan teman e-home lainnya yang telah menyediakan tempat untuk penulis untuk menghilangkan rasa bosan dikala suntuk kuliah.
9. Adam, Ferdian, Adit, Berli, dan teman teman FSMT lainnya yang telah menemani penulis untuk menghilangkan kebosanan dengan kegiatan futsal.
10. Alden, Dewa, Putra serta para penghuni laboratorium ADDI yang telah menemani pengerjaan tugas akhir ini selama di laboratorium.
11. Roby, Iqbal, Ucup, Ijul, Unyil, dan Icad yang menjadi teman cangkruk dikala penulis berada di kampung halaman.
12. Icak dan Redina yang telah menjadi tempat curhat penulis dikala penulis galau.
13. Yoga, Bayu, Fuad, Jundi, Iqbal, Tanjung, Erma, Nadya dan teman GIA lainnya yang telah mensupport penulis selama 9 tahun untuk menyelesaikan penelitian ini.
14. Naufal, Galih, dan Endar yang telah membantu penulis untuk melakukan labeling data untuk penelitian ini.
15. Lutfia Nuzul yang telah membantu melakukan merapikan buku TA penulis
16. Teman teman di ITS EXPO, HMSI ITS dan OSIRIS yang telah memberikan banyak kenangan manis dan pahit semasa kuliah.
17. Berbagai pihak yang tidak bisa disebutkan satu persatu yang telah turut serta sukseskan penulis dalam menyelesaikan tugas akhir.

Semoga buku tugas akhir ini dapat memberikan manfaat bagi pembaca.

Surabaya, 30 Juni 2018

Penulis,

Adrian Afnandika

## DAFTAR ISI

ABSTRAK .....	v
ABSTRACT .....	vii
KATA PENGANTAR .....	ix
DAFTAR ISI .....	xi
DAFTAR GAMBAR .....	xv
DAFTAR TABEL .....	xix
DAFTAR KODE .....	xxiii
BAB I PENDAHULUAN .....	1
1.1 Latar Belakang Masalah .....	1
1.2 Perumusan Masalah .....	3
1.3 Batasan Masalah .....	3
1.4 Tujuan Penelitian .....	4
1.5 Manfaat Penelitian .....	4
1.6 Relevansi .....	4
BAB II TINJAUAN PUSTAKA .....	7
2.1 Studi Sebelumnya .....	7
2.2 Dasar Teori .....	8
2.2.1 Analisis Sentimen .....	8
2.2.2 Machine Learning .....	9
2.2.3 Natural Language Processing .....	10
2.2.4 Word Embedding .....	11
2.2.5 Convolutional Neural Network ( CNN ) .....	13
2.2.6 Media Sosial .....	17
2.2.7 Crawling .....	18
2.2.8 Operator Telekomunikasi .....	18
BAB III METODOLOGI .....	19
3.1 Tahapan Pelaksanaan Tugas Akhir .....	19
3.2 Arsitektur Penelitian .....	20
3.3 Uraian Metodologi .....	20
3.3.1 Studi Literatur .....	21
3.3.2 Pengumpulan Data .....	21
3.3.3 Pre-Processing Data .....	21



3.3.4	Training Data .....	25
3.3.5	Analisis Model .....	26
3.3.6	Penyusunan Laporan Tugas Akhir .....	27
<b>BAB IV</b>	<b>PERANCANGAN.....</b>	<b>29</b>
4.1	Akuisisi Data Media Sosial .....	29
4.2	Perancangan <i>Crawler</i> .....	32
4.2.1	Desain Database .....	32
4.3	Perancangan <i>Pre-Processing Dataset</i> .....	34
4.3.1	Perancangan Pemberian Label <i>Dataset</i> .....	34
4.3.2	Perancangan Ekstraksi <i>Dataset</i> .....	36
4.3.3	Perancangan Penghapusan Duplikasi <i>Dataset</i> ..	37
4.3.4	Perancangan Pembersihan <i>Dataset</i> .....	37
4.4	Perancangan <i>Filtering</i> Bahasa Indonesia .....	39
4.5	Perancangan Pembuatan Model <i>Word Embedding</i> .....	39
4.5.1	Perancangan <i>Dataset</i> Model <i>Word Embedding</i>	39
4.5.2	Perancangan <i>Training</i> Model <i>Word Embedding</i>	40
4.6	Perancangan Pembuatan Model CNN .....	41
4.6.1	Perancangan <i>Dataset</i> Model CNN .....	41
4.6.2	Perancangan <i>Training</i> Model CNN .....	42
4.6.3	Perancangan <i>Testing</i> Model <i>CNN</i> .....	44
4.6.4	Perancangan Skenario <i>Training</i> Model CNN ..	44
<b>BAB V</b>	<b>IMPLEMENTASI .....</b>	<b>47</b>
5.1	Lingkungan Implementasi .....	47
5.2	Pembuatan <i>Crawler</i> Twitter .....	48
5.2.1	Pembuatan <i>Crawler Word Embedding</i> .....	48
5.2.2	Pembuatan <i>Crawler</i> Analisis Sentiment .....	52
5.3	Pembuatan <i>Filtering</i> Bahasa Indonesia .....	55
5.4	Pembuatan Model <i>Word Embedding</i> .....	58
5.4.1	Pembuatan Pembersihan <i>Dataset</i> .....	58
5.4.2	Pembuatan <i>Training</i> Model <i>Word Embedding</i> .	65
5.5	Pembuatan Model CNN .....	67
5.5.1	Pembuatan <i>Training</i> dan <i>Testing</i> Model CNN..	67
<b>BAB VI</b>	<b>HASIL DAN PEMBAHASAN.....</b>	<b>93</b>
6.1	Hasil Data Crawling .....	93
6.2	Hasil Penghapusan Duplikasi <i>Dataset</i> .....	94

6.3 Hasil Pemberian Label <i>Dataset</i> .....	96
6.4 Hasil Filtering Bahasa Indonesia .....	101
6.5 Hasil Pembersihan Dataset .....	102
6.6 Hasil Pembuatan Model Word Embedding .....	103
6.7 Hasil Pembuatan Model CNN.....	104
6.7.1 Konfigurasi Parameter Awal .....	104
6.7.2 Subtask C.....	105
6.7.3 Subtask B .....	122
6.7.4 Subtask A.....	139
6.7.5 Analisis Antar Subtask .....	154
6.8 Hasil Perbandingan Dengan Algoritma Lain .....	159
6.9 Uji Signifikansi .....	160
6.9.1 Pengaruh Model Embedding Subtask A.....	160
<b>BAB VII KESIMPULAN DAN SARAN .....</b>	<b>163</b>
7.1 Kesimpulan .....	163
7.2 Saran .....	164
<b>DAFTAR PUSTAKA .....</b>	<b>166</b>
<b>BIODATA PENULIS .....</b>	<b>169</b>
<b>LAMPIRAN A .....</b>	<b>171</b>



## DAFTAR GAMBAR

Gambar 2.1 Perbedaan CBOW dan Skip-gram.....	12
Gambar 2.2 Arsitektur Fasttext .....	13
Gambar 2.3 Proses CNN Pada Teks.....	14
Gambar 2.4 Proses Konvolusi .....	15
Gambar 2.5 Proses Konvolusi Pada Teks .....	15
Gambar 2.6 Proses Max-Pooling .....	16
Gambar 2.7 Grafik Fungsi ReLU .....	17
Gambar 3.1 Metodologi Penelitian .....	19
Gambar 3.2 Arsitektur Penelitian.....	20
Gambar 3.3 Proses Pre-Processing Data .....	22
Gambar 3.4 Proses Training Data .....	26
Gambar 4.1 Alur Anotasi Label Tweet .....	35
Gambar 4.2 Contoh Atribut Mentah .....	36
Gambar 4.3 Contoh Atribut <i>Tweeti Yang Dimabil</i> .....	37
Gambar 4.4 Contoh Hasil Pembersihan <i>Tweet</i> .....	38
Gambar 4.5 Contoh Hasil Deteksi Bahasa .....	39
Gambar 4.6 Alur Konvolusi .....	43
Gambar 4.7 Alur Skenario Training Model CNN .....	45
Gambar 5.1 Contoh Hasil <i>copy-paste</i> .....	53
Gambar 5.2 Contoh Hasil Pembersihan .....	64
Gambar 5.3 Contoh Hasil Pembagian Data.....	73
Gambar 5.4 Objek Untuk <i>Embedding</i> .....	81
Gambar 5.5 <i>Layer</i> Konvolusi .....	82
Gambar 5.6 Hasil Dari Parameter Model.....	83
Gambar 5.7 Hasil Method Forward() .....	85

Gambar 6.1 <i>Confusion Matrix</i> Anotator 1 dan 2 .....	98
Gambar 6.2 <i>Confusion Matrix</i> Anotator 1 dan 3 .....	99
Gambar 6.3 <i>Confusion Matrix</i> Anotator 2 dan 3 .....	99
Gambar 6.4 Akurasi Training Model .....	106
Gambar 6.5 Grafik Perubahan Akurasi .....	107
Gambar 6.6 Akurasi Training Model .....	109
Gambar 6.7 Perbandingan Akurasi.....	110
Gambar 6.8 Perbandingan Akurasi.....	111
Gambar 6.9 Grafik Perubahan Akurasi .....	111
Gambar 6.10 Akurasi Training Model.....	113
Gambar 6.11 Grafik Perubahan Akurasi .....	114
Gambar 6.12 Akurasi Training Model.....	115
Gambar 6.13 Perbandingan Akurasi Model .....	116
Gambar 6.14 Perbandingan Akurasi Model .....	117
Gambar 6.15 Grafik Perubahan Akurasi .....	117
Gambar 6.16 <i>Confusion Matrix Static</i> .....	119
Gambar 6.17 <i>Confusion Matrix Non Static</i> .....	119
Gambar 6.18 Perbandingan Akurasi Model .....	120
Gambar 6.19 Perbandingan Jumlah Kata .....	122
Gambar 6.20 Akurasi Training Model.....	124
Gambar 6.21 Grafik Perubahan Akurasi .....	125
Gambar 6.22 Perbandingan Akurasi Model.....	126
Gambar 6.23 Perbandingan Akurasi Model .....	127
Gambar 6.24 Perbandingan Akurasi Model .....	128
Gambar 6.25 Grafik Perbandingan Akurasi .....	128
Gambar 6.26 Akurasi training model .....	130



Gambar 6.27 Grafik Perubahan Akurasi .....	131
Gambar 6.28 Akurasi Training Model .....	133
Gambar 6.29 Perbandingan Akurasi Model .....	134
Gambar 6.30 Perbandingan Akurasi Model .....	134
Gambar 6.31 Grafik Perubahan Akurasi .....	135
Gambar 6.32 <i>Confusion Matrix Static</i> .....	136
Gambar 6.33 <i>Confusion Matrix Non Static</i> .....	137
Gambar 6.34 Perbandingan Akurasi .....	138
Gambar 6.35 Hasil training model .....	141
Gambar 6.36 Grafik Perubahan Akurasi .....	142
Gambar 6.37 Akurasi Training Model .....	143
Gambar 6.38 Perbandingan Akurasi Model .....	144
Gambar 6.39 Perbandingan Akurasi Model .....	144
Gambar 6.40 Grafik Perubahan Akurasi .....	145
Gambar 6.41 Akurasi Training Model .....	147
Gambar 6.42 Grafik Perubahan Akurasi .....	148
Gambar 6.43 Akurasi Training Model .....	149
Gambar 6.44 Perbandingan Akurasi Model .....	150
Gambar 6.45 Perbandingan Akurasi Model .....	151
Gambar 6.46 Grafik Perubahan Akurasi .....	151
Gambar 6.47 Akurasi Training Model .....	153



## DAFTAR TABEL

Tabel 3.1 Perbandingan Jenis Label.....	23
Tabel 4.1 Contoh Kata Kunci <i>Crawling Word Embedding</i> .....	29
Tabel 4.2 Contoh Kata Kunci <i>Crawling Data Topik</i> .....	31
Tabel 4.3 Kolom Tabel <i>Word Embedding</i> .....	33
Tabel 4.4 Kolom Tabel Data Topik .....	33
Tabel 4.5 Contoh Label <i>tweet</i> .....	36
Tabel 4.6 Paramter Word Embedding .....	40
Tabel 4.7 Perlakuan Label.....	41
Tabel 4.8 Paramter Model CNN .....	42
Tabel 4.9 Paramter Training Model CNN.....	44
Tabel 5.1 Spesifikasi <i>Hardware</i> .....	47
Tabel 5.2 <i>Library</i> Yang Digunakan .....	47
Tabel 5.3 Translasi Emoticon.....	62
Tabel 6.1 Hasil <i>Crawling Tweet</i> .....	93
Tabel 6.2 Hasil Tweet duplikat <i>word embedding</i> .....	94
Tabel 6.3 hasil tweet duplikat data topik.....	95
Tabel 6.4 Label Tweet.....	96
Tabel 6.5 Label Tweet.....	96
Tabel 6.6 Jumlah Distribusi Tweet berdasarkan label .....	96
Tabel 6.7 Jumlah distribusi tweet berdasarkan topik .....	96
Tabel 6.8 Distribusi Label per Anotator.....	98
Tabel 6.9 Tingkat Kesepakatan <i>Cohens Kappa</i> .....	100
Tabel 6.10 Hasil <i>Cohens Kappa</i> .....	100
Tabel 6.11 Hasil seleksi bahasa Indonesia .....	101

Tabel 6.12 Proese Pembersihan Tweet.....	102
Tabel 6.13 Hasil Parameter Word Embedding .....	104
Tabel 6.14 Konfigurasi Awal Model CNN.....	104
Tabel 6.15 Hasil Akurasi Model.....	105
Tabel 6.16 Akurasi Training Model .....	108
Tabel 6.17 Hasil Akurasi Model.....	112
Tabel 6.18 Hasil Akurasi Model.....	114
Tabel 6.19 Hasil Akurasi Model.....	118
Tabel 6.20 Hasil Akurasi Model.....	121
Tabel 6.21 Hasil Training Model .....	123
Tabel 6.22 Hasil Akurasi Model.....	125
Tabel 6.23 Hasil Akurasi model .....	129
Tabel 6.24 Hasil Akurasi Model.....	132
Tabel 6.25 Hasil Akurasi Model.....	135
Tabel 6.26 Hasil Akurasi Model.....	139
Tabel 6.27 Hasil Akurasi Model.....	140
Tabel 6.28 Hasil Akurasi Model.....	142
Tabel 6.29 Hasil Akurasi Model.....	146
Tabel 6.30 Hasil Akurasi Model.....	148
Tabel 6.31 Hasil Akurasi Model.....	152
Tabel 6.32 Hasil Akurasi Model.....	154
Tabel 6.33 Perbandingan Konfigurasi Model.....	154
Tabel 6.34 Hasil Akurasi Antar <i>Subtask</i> .....	155
Tabel 6.35 Hasil Perbandingan Akurasi <i>Single Filter Size</i> Per Subtask .....	156

Tabel 6.36 Tabel Perbandingan Akurasi Multi Region Size antar Subtask .....	157
Tabel 6.37 Perbandingan Akurasi Berdasarkan Model <i>Embedding</i> .....	158
Tabel 6.38 Perbandingan Akurasi Dengan Antar Algoritma .	160
Tabel 6.39 Hasil Akurasi.....	160
Tabel 6.40 Hasil Perhitungan .....	161





## DAFTAR KODE

Kode 5.1 Potongan Kode Implementasi <i>API</i> Untuk <i>Crawling</i>	48
Kode 5.2 Potongan Kode Parameter Proses <i>Crawling</i> .....	49
Kode 5.3 Potongan Kode Perulangan <i>Crawling</i> .....	50
Kode 5.4 Potongan Kode Untuk Penulisan Hasil <i>Crawling</i> ....	51
Kode 5.5 Potongan Kode Untuk <i>Scroll</i> Halaman <i>Browser</i> .....	52
Kode 5.6 Potongan Kode Untuk Mendapatkan Kalimat <i>Tweet</i>	54
Kode 5.7 Potongan Kode Untuk Menghilangkan <i>Tweet</i> Terpotong .....	55
Kode 5.8 Potongan Kode Untuk Mendapatkan Data <i>Tweet</i> .....	56
Kode 5.9 Potongan Kode Untuk Mendeteksi Bahasa .....	57
Kode 5.10 Potongan Kode Untuk <i>Method</i> Pembersihan.....	58
Kode 5.11 Potongan Kode Untuk Menghilangkan Simbol.....	59
Kode 5.12 Potongan Kode Untuk Mengganti <i>Emoticon</i> .....	61
Kode 5.13 Potongan Kode Urutan Pembersihan Data .....	63
Kode 5.14 Potongan Kode Parameter <i>Word Embedding</i> .....	65
Kode 5.15 Potongan Kode <i>Training</i> Model <i>Word Embedding</i>	66
Kode 5.16 Potongan Kode Parameter Kelas <i>Prepare_Data</i> ....	68
Kode 5.17 Potongan Kode Pemanggilan Kelas <i>Readdatatopik</i>	69
Kode 5.18 Potongan Kode Untuk Merubah Label Menjadi Kata .....	70
Kode 5.19 Potongan Kode Untuk Membaca Model <i>Word</i> <i>Embedding</i> .....	70
Kode 5.20 Potongan Kode Untuk <i>Method Execute()</i> .....	71
Kode 5.21 Potongan Kode Transformasi Data.....	72
Kode 5.22 Potongan Kode Mendapatkan Label Dan Kalimat <i>Tweet</i> .....	74

Kode 5.23 Potongan Kode Untuk Membuat <i>Vocabulary</i> .....	75
Kode 5.24 Potongan Kode Untuk Melakukan Proses Pembobotan Kata.....	76
Kode 5.25 Potongan Kode Untuk Mendapatkan Data <i>Training</i> .....	77
Kode 5.26 Potongan Kode Untuk Training Data.....	78
Kode 5.27 Potongan Kode Parameter Model CNN.....	79
Kode 5.28 Potongan Kode Kelas Model CNN.....	80
Kode 5.29 Potongan Kode Untuk Merubah <i>State</i> Model .....	82
Kode 5.30 Potongan Kode Untuk Perulangan Setiap <i>Epoch</i> ....	83
Kode 5.31 Potongan Kode Mekanisme <i>Training</i> Data.....	84
Kode 5.32 Potongan Kode <i>Method Forward()</i> .....	85
Kode 5.33 Potongan Kode Untuk Mendapatkan Prediksi Kalimat .....	87
Kode 5.34 Potongan Kode Untuk Menampilkan Statistik <i>Training</i> .....	88
Kode 5.35 Potongan Kode <i>Method Evaluate()</i> .....	89
Kode 5.36 Potongan Kode <i>Method Calcualte_Fold_Counts()</i> .....	90
Kode 5.37 Potongan Kode Method <i>Display_Measures()</i> .....	91

# **BAB I**

## **PENDAHULUAN**

Bab pendahuluan ini menguraikan proses identifikasi masalah penelitian yang meliputi latar belakang masalah, perumusan masalah, batasan masalah, tujuan tugas akhir, manfaat kegiatan tugas akhir dan relevansi terhadap pengerjaan tugas akhir. Berdasarkan uraian pada bab ini, harapannya gambaran umum permasalahan dan pemecahan masalah pada tugas akhir dapat dipahami.

### **1.1 Latar Belakang Masalah**

Sebagai salah satu negara dengan pengguna internet terbesar di dunia, Indonesia memiliki jumlah pengguna aktif internet sebanyak 132 juta pengguna pada tahun 2017 [1]. Pengguna tersebut menggunakan internet dengan berbagai tujuan, seperti media sosial. Media sosial memiliki proporsi pengguna yang cukup signifikan yaitu berkisar angka 40 % dari seluruh pengguna internet di Indonesia [1]. Perangkat mobile adalah salah satu *gadget* yang paling sering digunakan dalam mengakses media sosial, yakni sebesar 85 % [1]. Selain media sosial, masyarakat Indonesia menggunakan internet untuk transaksi online [2]. Nilai transaksi online di Indonesia pada tahun 2017 mencapai US\$ 5,29 Miliar atau sekitar 70 triliun rupiah [2]. Jumlah pengguna internet di Indonesia sendiri pada tahun 2016 mengalami kenaikan sebesar sebesar 51 % atau sekitar 45 juta pengguna [1]. Pada tahun 2016 banyaknya pengguna internet ini disebabkan oleh kebutuhan untuk update informasi terkini, sekitar 31,3 juta pengguna internet di Indonesia [3]. Untuk mendapatkan informasi terbaru dapat menggunakan banyak media, seperti portal berita, media sosial, atau media lainnya. Berita mancanegara memiliki posisi pertama sebesar 20.6 % dari konten berita lain yang sering dikunjungi [3]. Perkembangan media sosial sekarang sangatlah pesat. Media sosial yang paling dikunjungi di Indonesia sendiri adalah *facebook* dengan jumlah pengunjung 71,6 juta, disusul dengan *instagram* dengan jumlah pengunjung 19.9 juta, disusul dengan

youtube, google plus dan twitter dengan masing masing jumlah pengguna 14.5 juta, 7.9 juta dan 7.2 juta [3].

Maraknya pengguna sosial dapat memberikan pengaruh positif maupun negatif terhadap penyebaran informasi maupun opini publik. Mudahnya penulisan pesan / status pada sosial media menjadi salah satu penyebab mengapa media sosial digemari oleh banyak kalangan, mulai dari usia muda hingga tua. Persebaran umur pengguna internet di Indonesia pada tahun 2016 adalah sebagai berikut, umur 10 – 24 tahun memiliki jumlah pengguna 24,4 juta pengguna, umur 25 – 34 rahunmemiliki jumlah pengguna 32,3 juta pengguna, umur 35 – 44 tahun memiliki jumlah pengguna 38,7 juta, umur 45 – 54 tahun memiliki jumlah pengguna 23.8 juta dan umur diatas 55 tahun memiliki jumlah pengguna 13.2 juta [3]. Variasi dari usia pengguna internet khususnya pada pengguna media sosial membuat pesan yang tercantum pada media sosial beragam, baik itu berupa saran, kritik, atau berupa komentar lain nya. Dengan banyaknya variasi dari pesan di media sosial seperti *twitter* atau *facebook* kita dapat mengambil sebuah pengamatan terhadap sebuah topik sehingga mendapatkan informasi lebih terhadap topik tersebut berdasarkan pesan di media sosial.

Analisis sentimen secara singkat adalah sebuah metode analisis terhadap opini publik sehingga kita mendapatkan informasi terkait topik tersebut dari banyak perspektif. Dalam melakukan analisis sentimen terdapat banyak aspek yang harus diperhatikan, seperti topik yang ingin dianalisis, sumber data untuk analisis sentimen dan algoritma yang ingin digunakan. Salah satu topik yang saat ini marak diperbincangkan adalah terkait layanan operator telekomunikasi. Analisis sentimen dapat bermanfaat pada topik terkait operator telekomunikasi, dimana fokus pada layanan dari para penyedia layanan telekomunikasi seperti Internet Service Provider maupun *Telecommunications* (telco). Pada tahun 2016 jumlah pelanggan layanan telekomunikasi di Indonesia mencapai 342.9 juta pelanggan [4]. Berbeda dengan tahun sebelumnya jumlah pelanggan layanan telekomunikasi di Indonesia berjumlah 321 juta pelanggan, sehingga pada tahun 2016 mengalami kenaikan

6 % [4]. Telkomsel menduduki urutan pertama dengan jumlah pelanggan 157.4 juta disusul oleh indosat, 3 dan XL [4]. Persaingan yang ketat dari penyedia layanan telekomunikasi ini menuntut mereka untuk meningkatkan kualitas masing masing layanan mereka, dalam hal ini analisis sentimen membantu menerka opini publik terhadap layanan tersebut.

Dalam penerapannya analisis sentimen membutuhkan algoritma agar dapat bekerja dengan baik untuk mengklasifikasi kalimat. *Convolutional Neural Network* (CNN) merupakan salah satu algoritma yang dapat diterapkan untuk melakukan analisis sentimen, yang saat ini memiliki performa cukup bagus untuk melakukan klasifikasi kalimat [5]. CNN pada umumnya diterapkan untuk klasifikasi gambar, namun dengan mengadopsi metode yang sama dan penyesuaian singkat CNN dapat diterapkan untuk analisis sentimen dan mendapatkan nilai akurasi jauh lebih tinggi dibandingkan dengan algoritma lain seperti *Naive Bayes* [6].

## 1.2 Perumusan Masalah

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

1. Bagaimana metode untuk mendapatkan dataset yang sesuai dengan topik?
2. Bagaimana melakukan pre-processing *dataset*?
3. Bagaimana membuat model algoritma *Convolutional Neural Network* (CNN)?
4. Bagaimana mengoptimalkan model algoritma *Convolutional Neural Network* (CNN)?

## 1.3 Batasan Masalah

Batasan permasalahan dalam pengerjaan tugas akhir ini adalah:

1. Batasan sumber dataset berasal dari media sosial (twitter).
2. Data yang digunakan terkait topik operator telekomunikasi

3. Proses menghilangkan duplikasi data menggunakan fitur *distinct* dari *mysql*.

#### **1.4 Tujuan Penelitian**

Tujuan dari penelitian ini adalah :

1. Menerapkan metode crawling untuk mendapatkan dataset sesuai topik.
2. Menerapkan metode *pre-processing* untuk memproses dataset sebelum dilakukan proses training.
3. Membuat model analisis sentimen terhadap pesan di media sosial dengan menggunakan algoritma *Convolutional Neural Network (CNN)*.
4. Menampilkan hasil analisa sentimen dari penelitian menggunakan algoritma *Convolutional Neural Network (CNN)*.

#### **1.5 Manfaat Penelitian**

Manfaat yang diharapkan dapat diperoleh dari tugas akhir ini adalah:

1. Bagi Penulis, untuk mengetahui metode *pre-processing* serta parameter terbaik untuk melakukan analisis sentimen terhadap dataset pesan di media sosial khususnya topik terkait operator telekomunikasi menggunakan algoritma *Convolutional Neural Network (CNN)*.
2. Bagi Masyarakat, sebagai bentuk awal yang memungkinkan untuk terdapat penelitian penelitian selanjutnya dan dapat dikembangkan kedalam beberapa hal seperti bisnis maupun rancang bangun aplikasi yang memanfaatkan algoritman *Convolutional Neural Network (CNN)*.

#### **1.6 Relevansi**

Relevansi tugas akhir ini terhadap laboratorium Akuisisi Data dan Diseminasi Informasi (ADDI) adalah karena tugas akhir ini berkaitan dengan penerapan mata kuliah bidang keilmuan

laboratorium ADDI. Mata kuliah tersebut antara lain Sistem Cerdas, Sistem Pendukung Keputusan, dan Penggalian Data dan Analitika Bisnis.



*Halaman ini sengaja dikosongkan*

## BAB II TINJAUAN PUSTAKA

Bab ini akan membahas penelitian sebelumnya yang berhubungan dengan tugas akhir dan teori - teori yang berkaitan dengan permasalahan tugas akhir ini.

### 2.1 Studi Sebelumnya

Pada subbab ini dijelaskan tentang referensi penelitian yang berkaitan dengan tugas akhir. Pada bagian ini memaparkan acuan penelitian sebelumnya yang digunakan oleh penulis dalam melakukan penelitiannya.

1. Penelitian pertama yaitu berjudul *Convolutional Neural Networks for Sentence Classification* oleh Yoon Kim [6]. Dalam penelitian ini membahas performa algoritma CNN dalam melakukan analisis sentimen untuk berbagai macam tipe data, seperti review film, review pelanggan dan kumpulan pertanyaan. Penggunaan berbagai variasi tipe CNN dan algoritma pembandingan lainnya juga dilakukan dalam penelitian ini guna mendapatkan hasil dengan akurasi terbaik untuk setiap tipe data yang digunakan. Untuk mendapatkan nilai vektor kata yang nantinya akan digunakan sebagai input, dalam penelitian ini menggunakan data vektor yang disediakan google ( *unsupervised learning* ) berjumlah 100 milyar baris data. Tipe CNN yang digunakan dalam penelitian ini adalah, *CNN-Rand*, *CNN-Static*, *CNN-Non-Static*, *CNN-Multichannel*. Hasil dari penelitian ini menunjukkan bahwa algoritma CNN memiliki performa cukup baik dibanding algoritma lain nya.
2. Penelitian kedua yaitu berjudul *SemEval-2016 Task 4: Sentiment Analysis in Twitter* yang dilakukan Preslav Nakov bersama tim penelitiannya [7]. Dalam penelitian ini berfokus pada metode untuk melakukan analisis sentimen pada data *twitter*. Pada penelitian ini menjelaskan bagaimana pembagian *subtasks* dapat

mempengaruhi akurasi setiap model yang ditraining. Selain hal tersebut, pada penelitian ini juga menekankan bagaimana langkah langkah menentukan label disetiap *tweets*, jumlah label untuk setiap *tweets* dan jumlah anotator setiap *tweets*.

3. Penelitian ketiga yaitu berjudul *A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification* yang dilakukan oleh Ye Zhang dan Byron C. wallace [5]. Dalam penelitian ini menjelaskan bagaimana setiap *hyperparameter* CNN berpengaruh terhadap pengaruh model untuk analisis sentimen. (dalam penelitian ini menggunakan 9 tipe data). Parameter yang pertama adalah input vektor sebagai bahan *training*, dalam penelitian ini menyatakan bahwa *word embedding* menggunakan *word2vec* memiliki akurasi tertinggi untuk beberapa tipe data. Parameter kedua adalah ukuran *filter region*, dalam penelitian ini menyatakan bahwa 7 merupakan ukuran terbaik baik *single region size* dan *multiple region size*. Parameter ketiga adalah ukuran *feature maps* pada setiap region, jumlah terbaiknya adalah berada pada rentang 100 – 600. Parameter keempat adalah strategi yang diterapkan pada *pooling layer*, dalam penelitian ini menunjukkan bahwa *1-max pooling* memberikan akurasi tertinggi. Parameter terakhir yaitu regularisasi / normalisasi dimana parameter ini memberikan pengaruh terkecil, sedangkan parameter yang memberikan efek terbesar adalah *filter region size* dan ukuran *feature maps*.

## 2.2 Dasar Teori

Berisi teori-teori yang mendukung serta berkaitan dengan tugas akhir yang sedang dikerjakan.

### 2.2.1 Analisis Sentimen

Analisis sentimen atau *Sentiment Analysis* adalah sebuah proses atau metode memahami, mengekstrak dan mengolah data

tekstual secara otomatis untuk mendapatkan informasi sentimen yang terdapat pada sebuah kalimat opini [8]. Analisis sentimen memiliki banyak kegunaan, salah satunya untuk mengidentifikasi kecenderungan opini publik terhadap sebuah objek, produk ataupun layanan. Bahkan di Amerika terdapat sekitar 20 – 30 perusahaan memfokuskan pada layanan analisis sentimen [9]. Salah satu sumber data atau sumber opini yang sering digunakan untuk analisis sentimen adalah twitter. Analisis sentimen pada twitter telah banyak digunakan untuk berbagai tujuan seperti tokoh politik, isu sosial dan penelitian pasar [7]. Melakukan analisis sentimen dapat menggunakan berbagai macam algoritma klasifikasi seperti *Naïve Bayes* (NB), *Support Vector Machine* (SVM), dan *Artificial Neural Network* (ANN) [10]. Dalam melakukan analisis sentimen setiap data atau tweets akan memiliki label masing masing. Label setiap data atau *tweets* dalam penelitian ini berjumlah 5 yaitu sangat positif, positif, netral, negatif dan sangat negatif.

### 2.2.2 Machine Learning

*Machine Learning* adalah sebuah *subfield* dari kecerdasan buatan atau *artificial intelligence* [11]. Machine learning saat ini semakin populer dan digunakan dalam berbagai macam industri untuk menyelesaikan berbagai tugas [11]. Klasifikasi, *Clustering*, *Pattern Recognition*, Analisa Regresi dan *Forecasting* adalah beberapa contoh dari tugas / *task* yang dapat diselesaikan menggunakan machine learning. Istilah *artificial intelligence* bukanlah hal baru dalam dunia *computer science*, istilah tersebut telah muncul sejak Alan Turing mengajukan pertanyaan "Can Machines Think?" atau "Apakah mesin dapat berpikir?" [12]. Sebuah ilustrasi penerapan machine learning untuk sebuah tugas klasifikasi adalah program *spam filter* pada email [11]. Program tersebut memiliki tujuan untuk mengidentifikasi apakah sebuah email adalah spam atau bukan. Untuk dapat membedakan apakah email tersebut adalah spam atau bukan maka perlu adanya contoh email yang disebut spam dan email yang bukan disebut spam, dalam konteks ini disebut *datasets*. Selain membutuhkan *datasets* program tersebut juga

membutuhkan algoritma klasifikasi seperti SVM atau ANN untuk melakukan training data dan klasifikasi data. *Datasets* tersebut nantinya akan dipisah menjadi dua kategori yaitu training data dan testing data. Pengembangan dari *machine learning* disebut *deep learning*. *Deep learning* adalah sebuah bidang keilmuan baru dalam *machine learning* yang saat ini sedang berkembang seiring berkembangnya GPU [13].

### 2.2.3 Natural Language Processing

*Natural Language Processing* adalah sebuah proses analisis linguistik berbasis komputer terhadap teks dengan tujuan atau tugas tertentu [14]. Saat ini NLP adalah salah satu *field* yang sangat aktif diteliti oleh para ahli. NLP sendiri memiliki banyak penerapan dalam aplikasi dunia nyata, seperti translasi mesin, filter spam, ekstraksi informasi, rangkuman terotomasi dan menjawab pertanyaan [15]. Tujuan utama dari NLP adalah mencapai proses linguistik semirip mungkin seperti manusia. Istilah NLP sebelumnya lebih mengarah kepada istilah NLU atau *Natural Language Understanding*. Sebuah NLP dikatakan *true NLU* apabila dapat melakukan beberapa hal berikut [14] :

1. Parafrase dari input teks
2. Melakukan translasi teks tersebut ke bahasa lain
3. Menjawab dari pertanyaan dari teks
4. Mengambil inti dari teks

*Natural Language Processing* memiliki beberapa level dalam melakukan prosesnya Level pertama disebut *phonology*, proses ini adalah proses interpretasi input berupa ucapan kata menjadi kata kata tertentu. Dalam level ini memiliki 3 aturan yaitu *phonetic rules* yaitu proses membedakan suara setiap kata, *phonemic rules* mendeteksi variasi pengucapan apabila beberapa kata diucapkan secara bersama dan *prosodic rules* untuk mendeteksi fluktuasi penekanan intonasi dalam satu kalimat [14].

Level selanjutnya disebut *morphology*, yaitu proses memisah kata menjadi kata dasar dan imbuhan kata . Level ini lebih

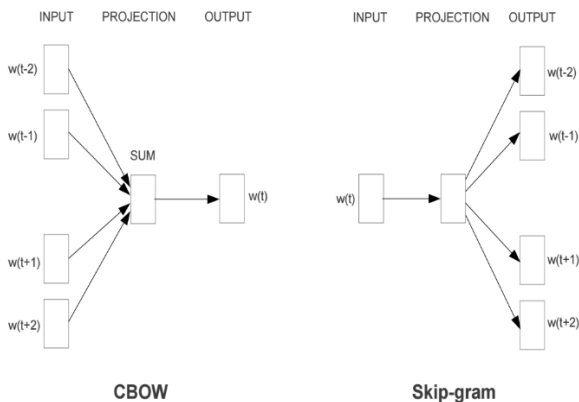
dikenal dengan nama aktivitas *stemming*. Contoh dari level ini adalah memisahkan kata “pekerjaan” menjadi “pe”, “kerja” dan “an”. Level ketiga disebut *lexical*, dimana sistem NLP menginterpretasikan definisi untuk setiap katanya. Pada level ini terdapat subproses untuk membantu NLP memahami definisi dari setiap kata, yaitu *part-of-speech-tagging*. *Syntactic* adalah level selanjutnya, dimana pada level ini fokus terhadap analisis kata dalam kalimat untuk mendeteksi *grammar* atau tata bahasa dari kalimat, sehingga output dalam level ini adalah struktural dependensi dari hubungan setiap kata. *Semantic processing* adalah level selanjutnya dari NLP yaitu proses definisi sebuah kalimat berdasarkan dari mayoritas pemikiran manusia, sehingga dalam determinasi makna sebuah kalimat memperhatikan keterkaitan setiap kata didalam kalimat [14].

## 2.2.4 Word Embedding

*Word Embedding* adalah salah satu metode untuk merepresentasikan kata kedalam nilai *vektor* tertentu. Pada awalnya *word representation* adalah istilah yang lebih dikenal daripada *word embedding* itu sendiri. Selain terdapat istilah tersebut, *word embedding* juga dikenal sebagai *distributed representation* karena memiliki kerapatan, menggunakan *low-dimensional vector* dan *real valued* [16]. *Word embedding* dapat membantu untuk menentukan kemiripan kata berdasarkan konteks tertentu dilihat dari nilai vektor kata tersebut.

### 2.2.4.1 Word2vec

*Word2vec* merupakan salah satu bentuk dari *word embedding* dimana dikembangkan oleh Google. *Word2vec* sendiri termasuk dalam kategori *neural network* yang menggunakan *hidden layer* dan beberapa *non-linier layer* didalam algoritmanya. Terdapat 2 jenis *word2vec* yaitu *Continuous Bag-of-words (CBOW)* dan *Skip-Gram Model* [17].

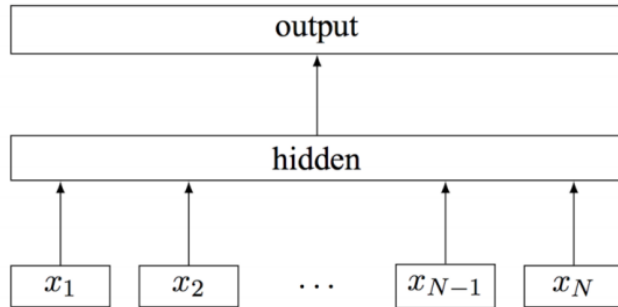


**Gambar 2.1 Perbedaan CBOW dan Skip-gram**

Secara garis besar perbedaan dari CBOW dan *Skip-Gram* adalah, arsitektur dari CBOW bertujuan untuk menghasilkan output sebuah kata dari beberapa konteks kata yang dimasukkan, sedangkan *Skip-Gram* bertujuan untuk mendeteksi beberapa kata yang sesuai secara konteks dari satu kata yang dimasukkan.

#### 2.2.4.2 Fasttext

*Fasttext* merupakan sebuah library untuk pembelajaran mesin yang lebih efisien pada klasifikasi kalimat dan *word representation*. Berbeda dengan *word2vec* algoritma ini memiliki struktur hirarki serta merepresentasikan dalam bentuk *dense vector*. Salah satu keunggulan dari *fasttext* adalah dapat mengatasi distribusi data yang kurang seimbang [18].



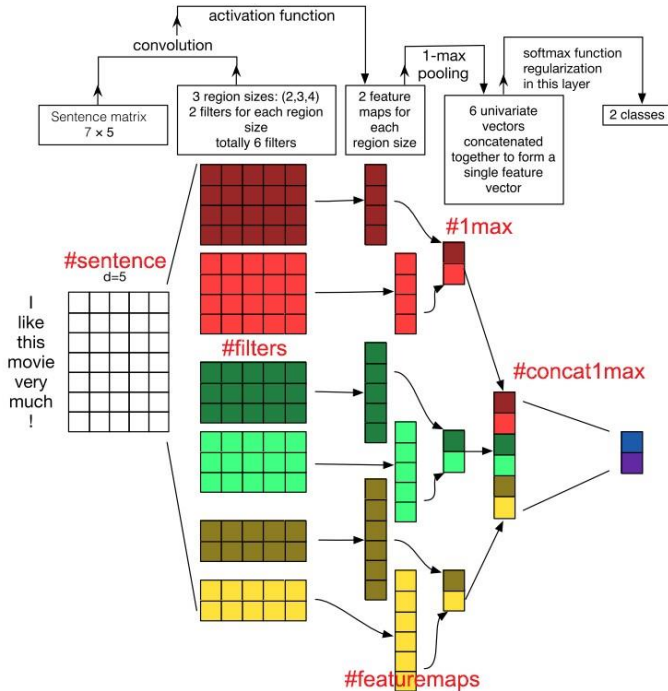
**Gambar 2.2** Arsitektur Fasttext

Arsitektur dari *fasttext* memiliki *hidden variabel* diantara *input layer* dan *output layer* [18].

### 2.2.5 Convolutional Neural Network ( CNN )

*Convolutional Neural Network* (CNN), menurut Wayan Suartika adalah pengembangan dari *Multilayer Perceptron* (MLP) yang didesain untuk mengolah data dua dimensi [13]. CNN pertama kali digunakan pada tahun 1989 oleh Yann LeCun untuk melakukan klasifikasi citra kode zip [13]. CNN dikembangkan oleh Kunihiko Fukushima dengan nama lain yaitu NeoCognitron [19]. CNN pada umumnya digunakan pada klasifikasi dua dimensi, atau gambar namun CNN juga dapat digunakan klasifikasi teks [6].





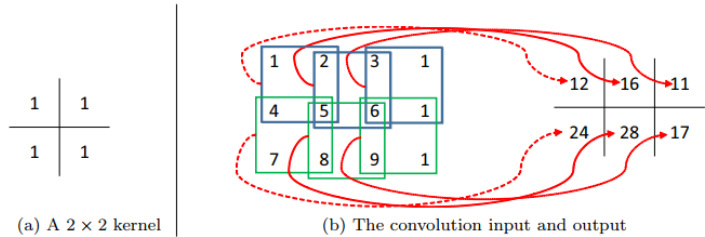
**Gambar 2.3 Proses CNN Pada Teks**

Berbeda dengan gambar yang memiliki 2 dimensi ( sumbu X dan sumbu Y ), teks hanya memiliki 1 dimensi yaitu urutan kata dalam satu kalimat. Konsep CNN adalah pengembangan dari *artificial neural network* (ANN) yaitu salah satu bentuk representasi buatan dari jaringan otak manusia dengan tujuan mensimulasikan proses pembelajaran pada otak manusia [20]. Propagasi balik atau *back propogatiom* adalah algoritma yang digunakan dalam ANN begitu juga CNN. Sebuah CNN memiliki 4 layer, yaitu *convolutional layer*, *subsampling layer*, *ReLU layer* dan *fully-connected layer* [21].

### 2.2.5.1 Convolution Layer

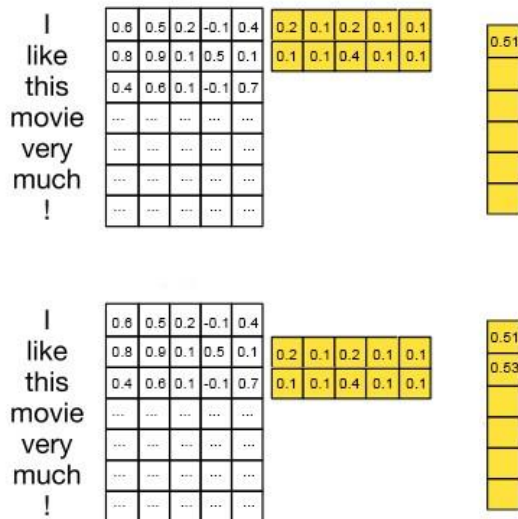
Konvolusi adalah istilah matematis untuk sebuah operasi yang berulang terhadap output sebelumnya [13]. Sedangkan *convolution layer* adalah lapisan yang

dihasilkan dari operasi *cross product* antara *input layer* dan *kernel / filter*. Operasi tersebut akan dilakukan secara terus menerus hingga semua nilai pada dimensi vektor terkalkulasi.



**Gambar 2.4 Proses Konvolusi**

Pada umumnya jumlah kernel tersebut lebih dari satu, dan merupakan salah satu bentuk parameter dari CNN. Pada tipe data dua dimensi atau gambar, operasi ini akan diulang sebanyak jumlah pixel input gambar. Berbeda dengan tipe data teks, teks harus dirubah terlebih dahulu menjadi vektor melalui proses word embedding [6].

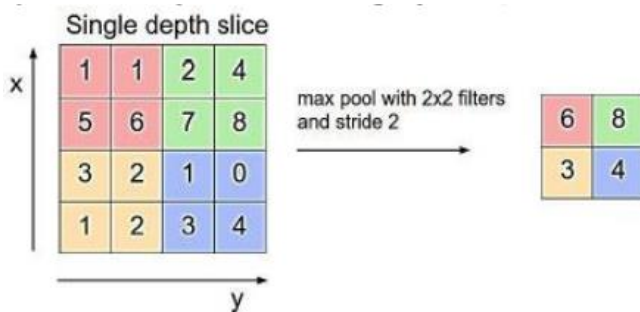


**Gambar 2.5 Proses Konvolusi Pada Teks**

Setelah merubah teks menjadi vektor proses konvolusi selanjutnya untuk tipe data teks sama dengan tipe data gambar. Dalam gambar X tersebut kotak putih adalah vektor setiap kata berukuran 7 X 5 dan kotak kuning adalah *kernel* yang berukuran 2 X 5. Kotak kuning paling kanan merupakan *convolutional layer* dari hasil *cross product* input vektor dengan kernel atau *filter region size*.

### 2.2.5.2 Subsampling Layer

*Subsampling layer* adalah proses mengurangi ukuran vektor *convolutional layer* menjadi ukuran yang lebih kecil. Salah satu tujuan dari *subsampling layer* adalah untuk mengurangi variasi posisi dari fitur. Metode yang digunakan untuk melakukan *subsampling layer* adalah *max pooling*. *Max pooling* membagi output vektor *convolutional layer* ke beberapa *grid* vektor dan mengambil nilai maksimal dari setiap *grid*, sehingga menghasilkan lapisan baru yang disebut *subsampling layer*.



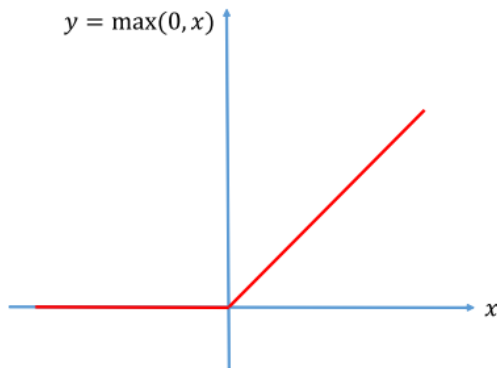
Gambar 2.6 Proses Max-Pooling

Proses ini dilakukan sebanyak jumlah *convolutional layer* yang ada, sehingga jumlah dari *subsampling layer* sama dengan *convolutional layer*.

### 2.2.5.3 ReLU Layer

*ReLU Layer* atau disebut *Rectified Linear Unit Layer* adalah sebuah layer yang bertujuan mentransformasi nilai negatif

pada vektor menjadi 0, sedangkan nilai positif tidak berubah [21].



**Gambar 2.7 Grafik Fungsi ReLU**

*ReLU layer* disebut juga fungsi aktivasi pada *neural network* termasuk CNN. Lapisan ini tidak merubah ukuran dari input vektor hanya merubah nilainya saja. Tujuan utama lapisan ini dalam implementasi CNN adalah meningkatkan *nonlinearity* [21].

#### 2.2.5.4 Fully-Connected Layer

Lapisan terakhir dalam CNN disebut *fully connected layer* dimana layer ini akan menghubungkan lapisan terakhir dari lapisan sebelumnya di setiap *kernel* menjadi satu dimensi, sehingga dapat melakukan transformasi pada dimensi agar dapat diklasifikasikan secara linear [13]. Proses konvolusi, *pooling* dan aktivasi dapat dilakukan beberapa kali dan diakhiri di lapisan ini. Setelah lapisan ini terbentuk maka akan diklasifikasikan menjadi kelas kelas tertentu [21]. Proses klasifikasi kembali ke algoritma awal *neural network* yaitu *backpropagation*.

#### 2.2.6 Media Sosial

Media sosial merupakan sebuah media bagi penggunanya untuk berpartisipasi, berbagi dan menciptakan konten di dunia virtual melalui internet untuk menyampaikan aspirasi atau

informasi kepada publik [22]. Media sosial memiliki banyak jenis seperti mikroblog, forum diskusi atau E-commerce

### **2.2.6.1 Twitter**

*Twitter* adalah media sosial yang memiliki kategori mikroblog yang memberikan fasilitas layanan jaringan sosial bagi pengguna untuk mengirimkan “status” atau lebih dikenal dengan istilah *tweets* [22]. *Twitter* pertama kali berdiri pada Maret tahun 2006 yang didirikan oleh Obvious Corp.

### **2.2.7 Crawling**

Crawling merupakan sebuah proses untuk mendapatkan informasi tertentu dari sebuah laman di sebuah website dan menyimpan informasi tersebut secara offline [22]. Crawling dapat dilakukan dengan memanfaatkan *application programming interface* (API) yang dikembangkan oleh *twitter*, sehingga dapat mendapatkan data berupa *tweets* yang dibutuhkan untuk penelitian ini.

### **2.2.8 Operator Telekomunikasi**

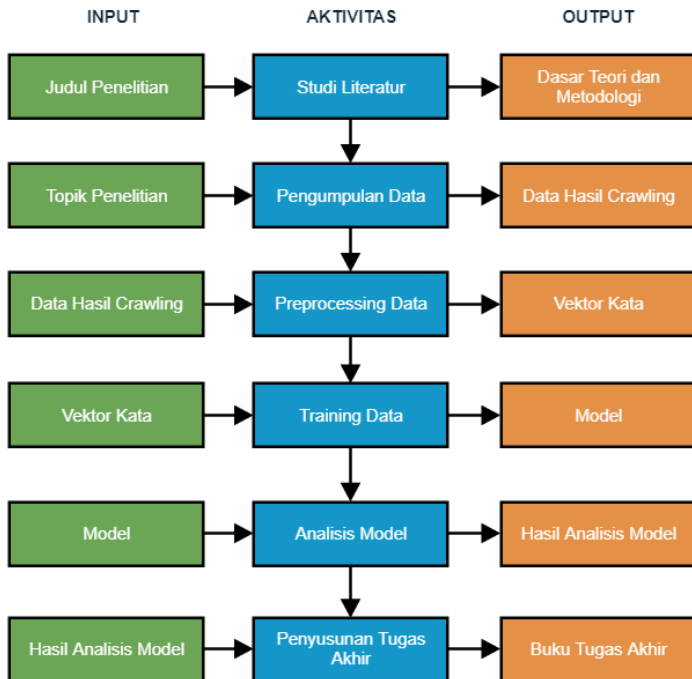
Operator telekomunikasi adalah perusahaan yang menawarkan jasa telekomunikasi meliputi sms, telpon dan akses internet kepada penggunanya [23]. Teknologi yang digunakan oleh perusahaan ini dibagi menjadi dua jenis yaitu *global system mobile communication* (GSM) dan *code division multipple acces* (CDMA). Jumlah operator telkomsel sendiri di indonesia saat ini yaitu 11, namun dalam penelitian ini dipilih 4 operator dengan pengguna terbanyak yaitu Telkomsel, Indosat, 3 dan XL.

## BAB III METODOLOGI

Bab ini menjelaskan tentang metodologi yang akan digunakan dalam penyusunan tugas akhir. Metodologi akan digunakan sebagai panduan dalam penyusunan tugas akhir agar terarah dan sistematis.

### 3.1 Tahapan Pelaksanaan Tugas Akhir

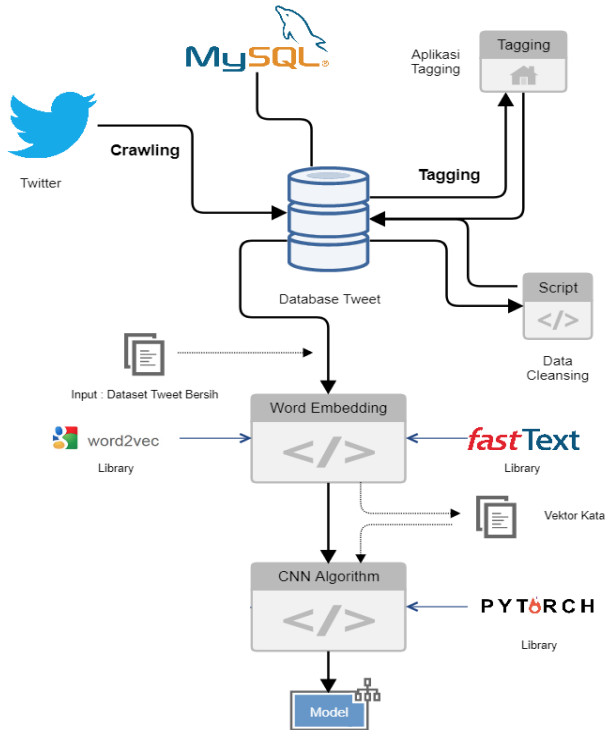
Pada subbab ini akan menjelaskan mengenai metodologi dalam pelaksanaan tugas akhir. Metodologi ini dapat dilihat pada gambar berikut.



Gambar 3.1 Metodologi Penelitian

### 3.2 Arsitektur Penelitian

Pada bagian ini akan menjelaskan arsitektur pada penelitian ini, arsitektur ini akan menjelaskan secara garis besar aktivitas beserta input, output dan metode di setiap prosesnya. Arsitektur dapat dilihat pada gambar berikut.



Gambar 3.2 Arsitektur Penelitian

### 3.3 Uraian Metodologi

Pada bagian ini akan dijelaskan secara lebih rinci masing-masing tahapan yang dilakukan untuk penyelesaian tugas akhir ini.

### 3.3.1 Studi Literatur

Pada tahap ini dilakukan dengan tujuan untuk memahami konsep, metode dan algoritma sesuai bahasan dan permasalahan, sehingga dapat memberi solusi mengenai permasalahan yang akan digunakan dalam penyusunan tugas akhir. Adapun literatur utama yang digunakan sebagai pedoman utama dalam penyusunan tugas akhir yaitu, *A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification* oleh Ye Zhang [5], *Distributed Representations of Words and Phrases and their Compositionality* oleh Tomas Mikolov [24], dan *Convolutional Neural Networks for Sentence Classification* oleh Kim Yoon [6].

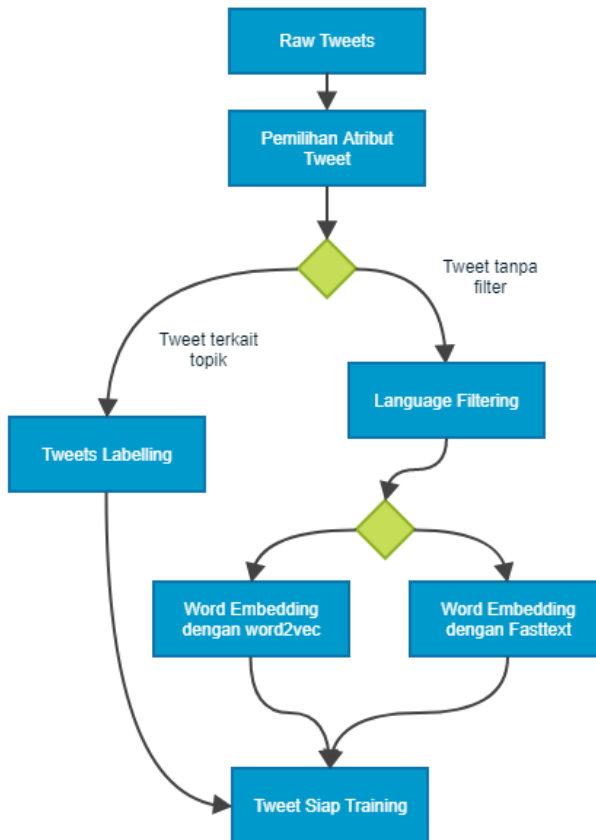
### 3.3.2 Pengumpulan Data

Tahap selanjutnya yaitu tahap pengumpulan data *tweets* yang akan digunakan untuk *training* data ditahap selanjutnya. Untuk mendapatkan *tweets* yang sesuai, maka dalam tahap *crawling* menggunakan kata kunci yang terkait dengan topik operator telekomunikasi. Selain menggunakan kata kunci yang terkait dengan topik operator, dalam penelitian ini akan mengumpulkan data yang tanpa filter untuk *word embedding*. Output dari tahap ini adalah kumpulan *tweet* mentah untuk diproses dulu pada tahap *preprocessing*

### 3.3.3 Pre-Processing Data

Tahap selanjutnya yaitu tahap *pre-processing* data, tahap ini bertujuan untuk memproses data mentah hasil *crawling* agar dapat digunakan sebagai data *training* di tahap selanjutnya. Pada aktivitas ini melewati banyak subaktivitas untuk *pre-processing* data, berikut diagram alur tahap *pre-processing* data. Output dari tahap ini adalah vektor kata hasil dari aktivitas *word embedding* sebagai input di tahap selanjutnya.





**Gambar 3.3** Proses Pre-Processing Data

### 3.3.3.1 Pemilihan Atribut Tweet

Tahap ini adalah memisahkan antara atribut yang penting dan tidak penting dalam penelitian ini. Tujuan dari pemilihan atribut ini untuk mengurangi ukuran dari dataset sehingga mempercepat proses training data. Untuk mempermudah pada aktivitas ini akan dibuat kode program sederhana untuk mengambil atribut yang di inginkan.

### 3.3.3.2 Language Filtering

Tahap ini adalah proses untuk menghilangkan *tweets* yang mengandung bahasa asing. Untuk melakukan pada aktivitas ini akan memanfaatkan library dari python yaitu *langdetect*. Untuk mempermudah pada aktivitas ini akan dibuat kode program sederhana yang menggunakan library tersebut untuk memudahkan filter bahasa. Tahap ini khusus untuk data *tweets* untuk proses *word embedding*

### 3.3.3.3 Tweets Labelling

Tahap selanjutnya melakukan pemberian label pada *tweets* yang sesuai topik operator telekomunikasi. Berdasarkan *penelitian SemEval-2016 Task 4: Sentiment Analysis in Twitter*, Setiap *tweets* akan memiliki 2 label yaitu label sentimen terhadap tweet dan label *tweets* terhadap topik tersebut [7]. Berikut perbedaan dari dua label tersebut.

**Tabel 3.1 Perbandingan Jenis Label**

<b>Tweet</b>	<b>Sentimen terhadap Tweets</b>	<b>Sentimen terhadap Topik</b>
Ahhhhh !!!, malam selasa tanpa menonton Indonesian Idol, berasa hampa, bosan dan sunyi. #KembalikanIdol2018	Negatif	( Indonesian Idol ) Positif

Untuk meningkatkan akurasi terdapat 3 buah skenario atau *subtask* dalam pemberian label yaitu :

- a. Subtask A : Positif dan Negatif.
- b. Subtask B : Positif, Negatif dan Netral.
- c. Subtask C : Sangat Positif, Positif, Netral, Negatif dan Sangat Negatif.

Untuk memberikan label pada *tweets* akan dilakukan oleh tiga orang anotator dengan tujuan mempertimbangkan asumsi masing masing individu [7]. Untuk mendapatkan

label akhir masing masing *tweets* akan dilakukan perhitungan sederhana, apabila dua orang atau lebih memilih label yang sama, maka label akhir adalah berdasarkan pemilihan tersebut, namun apabila berbeda maka akan dilakukan rata rata. Untuk menghitung rata rata, masing masing label memiliki nilai berikut :

- a. Sangat Negatif : 1
- b. Negatif : 2
- c. Netral : 3
- d. Positif : 4
- e. Sangat Positif : 5

Setelah dihitung rata rata, maka hasil tersebut akan dibulatkan apabila memiliki nilai desimal  $> 0.4$  akan dibulatkan keatas sedangkan  $< 0.4$  akan dibulatkan kebawah [7].

#### **3.3.3.4 Word Embedding dengan Word2vec**

Tahap selanjutnya yaitu tahap *word embedding* dengan library *word2vec*. Pada tahap ini bertujuan untuk merubah setiap kata pada data training menjadi nilai vector agar dapat diproses dalam algoritma CNN. Data yang digunakan pada tahap ini adalah data yang sesuai dengan topik operator telekomunikasi dan data tanpa filter. Dalam tahap ini akan dibuat kode program sederhana dengan memanfaatkan *library* bernama *Gensim* yang diimplementasikan menggunakan *python*. Dalam tahap ini akan dibuat beberapa skenario dengan bermain parameter yang tersedia sehingga mendapatkan model dengan akurasi tertinggi

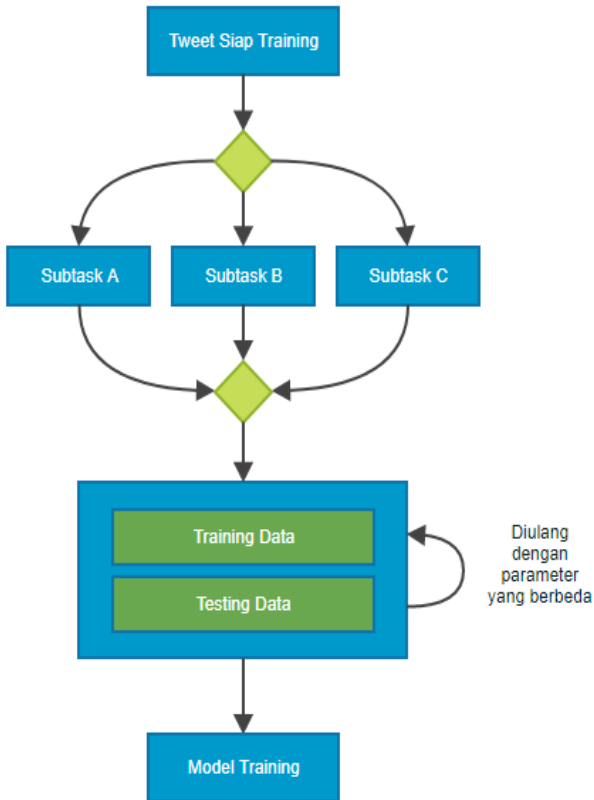
#### **3.3.3.5 Word Embedding dengan Fasttext**

*Word embedding* dapat dilakukan menggunakan library *fasttext* yang dikembangkan oleh facebook. Pada tahap ini dilakukan untuk menambah skenario data training sehingga diharapkan dapat memberikan model dengan akurasi tertinggi. Untuk menerapkan *word embedding* dengan

*fasttext* kita menggunakan data yang sesuai dengan topik operator telekomunikasi dan data tanpa filter sama seperti *word2vec* sebelumnya. Pada tahap *word embedding* ini baik *fasttext* maupun *word2vec* adalah bertipe *unsupervised learning* sehingga tidak melihat kelas / label yang terdapat pada data.

### 3.3.4 Training Data

Tahap selanjutnya yaitu tahap *training* data, dimana input data yang digunakan adalah vektor data yang terkait dengan topik operator telekomunikasi. *Tweets* akan dibagi menjadi dua jenis yaitu data training dan data testing dengan rasio 90 : 10. Dalam melakukan training data dengan CNN menggunakan *library* bernama *tensorflow* yang akan dibuat menggunakan *python*. Pada tahap ini akan dilakukan beberapa iterasi ( skenario ) berdasarkan *subtask*, metode *word embedding* dan konfigurasi parameter pada CNN. Output pada tahap ini adalah model model dari setiap skenario training yang akan dilakukan. Berikut diagram alur untuk tahap ini.



**Gambar 3.4 Proses Training Data**

### 3.3.5 Analisis Model

Tahap selanjutnya yaitu tahap analisis model yang berasal dari sebelumnya, dalam analisis model untuk menghitung performa masing masing model menggunakan metode *Cross-Entropy Loss* [25]. Metode ini sudah terdapat dalam library *tensorflow* yang digunakan pada tahap sebelumnya untuk melakukan training data. Output pada tahap ini adalah model dengan performa dan akurasi terbaik. Selain itu output pada tahap ini

adalah hasil analisis setiap model terkait nilai akurasi masing masing.

### **3.3.6 Penyusunan Laporan Tugas Akhir**

Tahapan terakhir adalah penyusunan laporan tugas akhir sebagai bentuk dokumentasi atas terlaksananya tugas akhir ini. Laporan tugas akhir dibuat sesuai dengan format yang telah ditentukan. Tahapan penyusunan laporan tugas akhir dilakukan sejak awal hingga berakhirnya proses pengerjaan tugas akhir ini.

*Halaman ini sengaja dikosongkan*

## BAB IV PERANCANGAN

Bab ini menjelaskan tentang metodologi yang akan digunakan dalam penyusunan tugas akhir. Metodologi akan digunakan sebagai panduan dalam penyusunan tugas akhir agar terarah dan sistematis.

### 4.1 Akusisi Data Media Sosial

Tahap awal perancangan dimulai dengan mengumpulkan seluruh data dari twitter baik untuk proses pembuatan model *word embedding* dan pembuatan model *CNN*. Pengambilan data dilakukan melalui *crawler*. Proses *word embedding* membutuhkan data dengan jumlah yang banyak sehingga dalam pengambilan data twitter menggunakan kata kunci yang umum dalam bahasa indonesia, berikut tabel beberapa kata kunci untuk data *word embedding*.

**Tabel 4.1 Contoh Kata Kunci Crawling Word Embedding**

No	Keyword	Alasan
1	Adalah	Merupakan salah satu kata penghubung yang sering digunakan, sehingga banyak <i>tweets</i> yang mengandung kata tersebut.
2	Maaf	Merupakan salah satu kata yang mengekspresikan kesedihan. Tata bahasa dari kata kunci ini beragam sehingga dapat menambah kosakata.
3	Cinta	Memiliki gaya bahasa yang beragam sehingga memperkaya kosa kata



No	Keyword	Alasan
4	Jadi	Merupakan salah satu kata penghubung yang sering digunakan, sehingga banyak <i>tweets</i> yang mengandung kata tersebut.
5	Kamu	Merupakan sebuah subjek dalam kalimat dan memiliki gaya bahasa yang beragam
6	Gue	<i>Tweets</i> yang terdapat kata ini sebagian besar memiliki tata bahasa yang tidak formal, sehingga kosakata dalam model nantinya cukup familiar dengan kalimat yang mengandung kata tidak formal
7	Anjir	<i>Tweets</i> yang terdapat kata ini sebagian besar memiliki tata bahasa yang tidak formal, sehingga kosakata dalam model nantinya cukup familiar dengan kalimat yang mengandung kata tidak formal

Berbeda dengan kata kunci yang digunakan untuk data *word embedding* kata kunci untuk analisis sentiment harus berdasarkan topik yang ditentukan, dimana dalam penelitian ini adalah provider telekomunikasi. Maka dari itu berikut beberapa kata kunci yang digunakan.

Tabel 4.2 Contoh Kata Kunci *Crawling* Data Topik

No	Keyword	Alasan
1	Telkomsel	Untuk mendapatkan <i>tweets</i> terkait provider tertentu maka kata kunci dari provider tersebut adalah yang utama, karena provider tersebut adalah objek dari <i>tweets</i> yang di inginkan.
2	Indosat	Untuk mendapatkan <i>tweets</i> terkait provider tertentu maka kata kunci dari provider tersebut adalah yang utama, karena provider tersebut adalah objek dari <i>tweets</i> yang di inginkan.
3	@myxl	Merupakan akun resmi dari provider XL, apabila menggunakan kata kunci 'xl' saja maka kurang presisi, karena tweet yang mengandung kata xl belum spesifik terkait dengan provider XL
4	Sinyal xl	<i>Tweets</i> yang berasal dari kata kunci @myxl sebagian berasal dari pemilik akun tersebut dan kurang merepresentasikan pengguna, sehingga kata kunci 'sinyal xl' dapat digunakan untuk mendapatkan <i>tweets</i> yang berhubungan dengan topik namun tidak mencantumkan '@myxl'

No	Keyword	Alasan
5	@triindonesia	Merupakan akun resmi dari provider TRI, apabila menggunakan kata kunci 'tri' saja maka kurang presisi, karena tweet yang mengandung kata tri belum spesifik terkait dengan provider TRI
6	Sinyal tri	<i>Tweets</i> yang berasal dari kata kunci @triindonesia sebagian berasal dari pemilik akun tersebut dan kurang merepresentasikan pengguna, sehingga kata kunci 'sinyal tri' dapat digunakan untuk mendapatkan <i>tweets</i> yang berhubungan dengan topik namun tidak mencantumkan '@triindonesia'

## 4.2 Perancangan *Crawler*

Untuk mengambil data secara otomatis maka dirancang sebuah *crawler* yang akan mengambil *tweets* dan menyimpan dalam sebuah file. Terdapat dua macam *crawler* yang digunakan, *crawler* pertama untuk data word embedding dimana memanfaatkan library *tweepy* dan *API* yang disediakan oleh twitter sedangkan *crawler* kedua untuk data topik menggunakan metode *scrapping* dari web browser.

### 4.2.1 Desain Database

Untuk melakukan perancangan *crawler*, maka perlu melakukan perancangan database untuk menyimpan data yang diambil, baik untuk *word embedding* dan CNN. Data yang diambil dari

twitter untuk *word embedding* adalah data *tweets* saja, sehingga berikut struktur tabel untuk menyimpan data tersebut.

**Tabel 4.3 Kolom Tabel *Word Embedding***

<b>Atribut</b>	<b>Tipe Data</b>	<b>Penjelasan</b>
id_tweet	Integer	Merupakan primary key dari tabel ini, berisi kode unik untuk setiap tweet
Text	Varchar	Berisi konten <i>tweets</i> yang diambil
Created_at	Datetime	Waktu kapan data tersebut ditambahkan kedalam database

Berbeda dengan *word embedding* untuk analisis sentiment membutuhkan informasi mengenai label dari setiap *tweets* apakah itu negatif, positif atau netral, berikut struktur tabelnya.

**Tabel 4.4 Kolom Tabel Data Topik**

<b>Atribut</b>	<b>Tipe Data</b>	<b>Penjelasan</b>
id_tweet	Integer	Merupakan primary key dari tabel ini, berisi kode unik untuk setiap tweet
Text	Varchar	Berisi konten <i>tweets</i> yang diambil

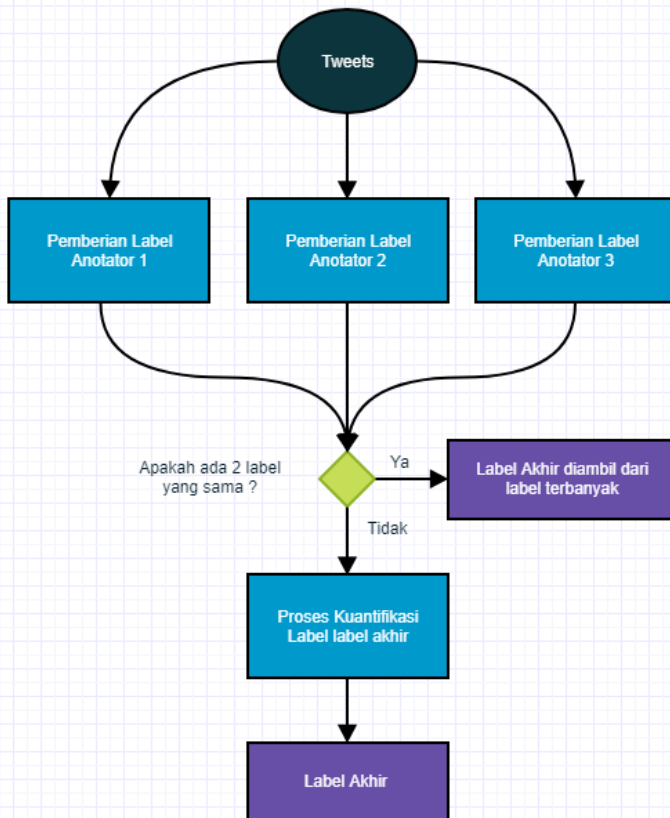
Atribut	Tipe Data	Penjelasan
Topic	Varchar	Berisi kata kunci dari <i>tweets</i> yang diambil
Label	Integer	Berisi label dari setiap <i>tweets</i> yang diambil
Created_at	Datetime	Waktu kapan data tersebut ditambahkan kedalam database

### 4.3 Perancangan *Pre-Processing Dataset*

Sebelum data diolah untuk analisis sentiment, data harus terlebih dahulu mengalami proses *pre-processing* data atau pra-pemrosesan data. Data mentah akan dipersiapkan terlebih dahulu menjadi format data yang lebih mudah dan efektif untuk *training* di tahap selanjutnya.

#### 4.3.1 Perancangan Pemberian Label *Dataset*

*Tweets* yang memiliki topik akan diberi label sesuai dengan sentimen yang dimiliki oleh *tweets* tersebut. Jumlah anotator yang akan memberikan label *tweets* berjumlah 3 orang dan memberikan masing masing 3 label terhadap *tweets*, yaitu label *tweets*, label topik dan label sarkasme, namun pada penelitian ini hanya berfokus terhadap label *tweets* saja. Jumlah anotator lebih dari satu orang memiliki tujuan untuk menghindari subjektivitas seseorang terhadap *tweets* tertentu sehingga sentimen *tweets* didapatkan dari perspektif dari lebih dari satu orang. Sehingga setiap *tweets* nantinya memiliki label *tweets* sejumlah 3 label. Berikut alur untuk menentukan label akhir dari sebuah *tweets*



**Gambar 4.1 Alur Anotasi Label Tweet**

Pada proses kuantifikasi label akan di translasikan kedalam numerik lalu dihitung rata – rata nya dan dibulatkan dengan aturan yang telah dijelaskan pada bab sebelumnya. Berikut contoh proses kuantifikasi sebuah *tweets*.



Ekstraksi atribut dilakukan dengan membuat kode program sederhana untuk mendapatkan atribut yang di inginkan saja sehingga, apabila telah dilakukan hasilnya seperti berikut.

Message	Result	Profile	Status
id_tweet	text		
99049426232080	3 tahun yg lalu makanya gahar. Hari ini ... belum makan, tapi rencananya sih makan bebek goreng. U0019603		
990494304128188088	04 ngabulka info sehat   Jagan Hindari Makanan Ini Jika Ingin Sehat		
990494293110558920	Ada bule bentaru. Gue sugulin Sabak Pondok. Bule White   've never seen a fruit like this. What 's called? Mm ( arja saku apaan ya bahasa inggrisnya, ah bodo amat   Umni ...   Shylyg		
990494165709861788	Alhamdulillah ya pokoknya mau kelup sembel bun ke apa ni. U0017648220019442001944200194420019442		
9904940726034547200	Ada Shuttle Bus SUV Gratis di Daerah Kemayoran Jakarta. Baca selengkapnya yuk		
990494023110904086	1 Mobil siapa? orang. Macet pula. Si A ngembut sim ni B Si B kemul sim ni A Gue ngomong am ni B. si B pasang muka bi ke gue Gue ngomong am ni B. si A pasang muka nyotik ke gue. Pengep bgt ngomong. Tunun aja deh lo bereslu		
99049401607460967	3 Tawonjil-Pangon ang Dipa Mendua! Hidup Muluk dan Sehat		
9904940136049141687	Ada juguk ya-lebi tak paham bahasa diah tau aku ni g' ong nak juguk kaku! lapaatu alu pik yg kena H4H4H4H4H4		
990494010722148224	'ambuk sama hitam tapi hati lain' ohh sorry tambuk   dyed color pink		
99049401044438668	20 menit before dia tengah potong rambut. Kijap lagi sampai lah ke halahahaha		
9904940176557661857	3 Keseraman di Daerah Hulu Bulungan Digerangi Banjir. Ketegingannya Sampai Segini via tribunkalim		
99049401378212443136	6 film dah nonton. U00019603 nyesel ku pater rabbit ama red spemua ga memb' banyak daerah sini		
990494015074916168	4 Movie Sinopsis Film hingga 2021: Cemas! Sengsem Infotly 'Star'. Sempai penerjemah A		
99049401576105847890	11. 10 BeritaONORA UPDATE PRASMANA CUCU DAN JAKARTA, Senin, 30 April 2018, Malam hingga Dini hari nanti, Info lengkap gt gt Sumber:Prasmanan u2013 BM&G RT @PRDIBakarta		
99049401569126927872	20 tahun kemudian, Pak. Kuntaran ang Bu Dewi hidup sebagai beban. U0001961b		
9904940151023991296	3 Position: Peringkat Pekan Olahraga Nasional. The Fast hingga Infidat Euro		
990494013034743868	100 Juta Memer Pelepasan Tanggahasi Hingga 20 April 2018		
99049401346031791156	Ada 3 daerah   yg dilawat dg 38   di Indonesia yg merajai incaran megate2 perajih di zaman kolonial Apa yg disebut b'lg 38   Banda, Bangka dan Bengkulu  , tiga tiganya untk. 'dikanak rebahan oleh negar2 perajih waktu itu. Dr. I		

Gambar 4.3 Contoh Atribut *Tweet* yang Dimambil

### 4.3.3 Perancangan Penghapusan Duplikasi *Dataset*

Data yang didapatkan dari proses *crawling* masih terdapat duplikasi, maka dari itu perlu dilakukan penghapusan data yang duplikat. Untuk melakukan hal ini menggunakan *query* sederhana yang di eksekusi di *mysql*. Berdasarkan metode yang digunakan untuk menghilangkan data duplikat, terdapat kelemahan yaitu tidak dapat menghilangkan *tweet* yang sama dengan *retweet* komentar yang berbeda. Hal tersebut dikarenakan *retweet* tersebut dihitung sebagai *tweet* baru. Sehingga proses penghapusan duplikasi *tweet* hanya dapat menghapus *tweet* yang sama persis melalui fitur *distinct* pada *mysql*.

### 4.3.4 Perancangan Pembersihan *Dataset*

Data yang didapatkan dari proses *crawling* masih terdapat banyak simbol dan karakter yang kurang berguna dalam proses training, maka dari itu perlu adanya aktivitas untuk membersihkan *tweets* dari karakter yang tidak diperlukan. Selain pembersihan dilakuka perubahan format terkait *hashtag* dan akun yang di-*mention* dalam *tweets*, hal ini dilakukan untuk penyederhanaan *tweets* agar *training* lebih efektif. Untuk lebih lengkapnya berikut urutan proses pembersihan *tweets*.

1. Menghilangkan *URL* yang terkandung dalam *tweets*



2. Menghilangkan atribut *HTML* yang terkandung dalam *tweets*
3. Merubah akun yang di *mention* dengan ‘<mention>’
4. Merubah *hashtag* menjadi ‘<hash\_tag>’
5. Merubah jumlah karakter sama yang diulang menjadi dua karakter yang sama, contoh ‘rumahhhh’ menjadi ‘rumahh’
6. Merubah *emoticons* yang terkandung dalam *tweets* menjadi bentuk kata kata, contoh ‘:D’ menjadi ‘<tertawa>’
7. Menghapus seluruh simbol dan tanda baca yang terdapat pada *tweets*
8. Merubah *tweets* menjadi huruf kecil

Setelah semua dilakukan maka berikut contoh hasil pembersihan *tweets*

```

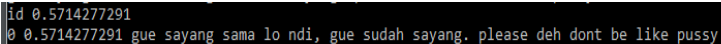
1 indosat mantap djaja koneksinya ngebut euy koyok jaran goyang cek
sini coba url mention mantap abiss elipsis senyum senyum
2 membalas mention mention mau nangis deh rasanya pdahl saya percaya
betul dengan indosat soal koneksi di wil karimun tp sudah hari
koneksi kacau balau
3 sinyal indosat asu ari musim hujan hadeuh
4 membalas mention cek dm woi layananan keluhan semua ga berfungsi
elipsis
5 ini kenapa di bale endah sinyal xl tiba bagus yah pindah kali yah
tower nya ke dekat rumah elipsis aneh liat hp sendiri
6 hotsale gb hanya gratis nelfon aja ga ada gratis sms nya tah min
mention
7 membalas mention mention dan lainnya hai kak andika ada yang bisa
saya bantu kami lihat akhir akhir ini anda sering sambat dengan
kualitas jaringan dari mention saya sarankan gunakan paket internet
dari mention agar anda tidak buang tenaga untuk marah marah
8 membalas mention mention iya min udah bisa kok hihhi tengkyu yaa
senyum seneng banget pake telkomsel sinyal lancaar
9 mention ribu saya beneran atau bohongan saya coba buat internet
lngsung tanpa daftar paket internet dan alhamdulillah bisa buat
internet trus saya buka halaman indosat elipsis baru dibuka saya dpt
sms dari indosat kamu internetan dg tarif perkb pemakai
10 sinyal kencang xl mantabbzz tertawa

```

**Gambar 4.4** Contoh Hasil Pembersihan *Tweet*

#### 4.4 Perancangan *Filtering* Bahasa Indonesia

*Tweets* yang diterima terkadang dalam bahasa asing meskipun telah menggunakan kata kunci bahasa Indonesia. Hal ini terjadi karena terkadang kata kunci yang digunakan adalah kata dalam bahasa lain. Maka dari itu perlu dilakukan *filtering* bahasa Indonesia. Tahap ini memerlukan *library* dalam *python* yang disebut *langdetect* dimana efektif untuk melakukan *filtering* bahasa. Setiap kalimat yang akan diseleksi akan memiliki indeks khusus yang merepresentasikan kandungan bahasanya. Dalam penelitian ini indeks yang digunakan untuk bahasa Indonesia adalah 0,5, agar *tweet* yang mengandung gaya bahasa Indonesia yang kurang formal tetap lolos dalam proses seleksi ini.



```
id 0.5714277291
id 0.5714277291 gue sayang sama lo ndi, gue sudah sayang. please deh dont be like pussy
```

Gambar 4.5 Contoh Hasil Deteksi Bahasa

#### 4.5 Perancangan Pembuatan Model *Word Embedding*

*Tweets* yang telah melalui proses *filtering* dan pembersihan *tweets*, selanjutnya akan menjadi input untuk *training* model *word embedding*. *Tweets* yang menjadi input hanya *tweets* yang dikhususkan untuk *word embedding* saja. Dalam penelitian ini akan menggunakan 3 model *word embedding*, dimana dua diantaranya adalah memiliki arsitektur *word2vec* dan sisanya adalah *fasttext*. Model *word2vec* sendiri dibedakan berdasarkan arsitektur atau *learning algorithm*-nya yaitu *CBOWS* dan *Skipgram*.

##### 4.5.1 Perancangan *Dataset* Model *Word Embedding*

*Dataset* yang digunakan adalah keseluruhan *tweets* yang tidak berkaitan dengan topik dalam penelitian, sehingga nantinya proses pemberian bobot vektor kata awal ( *initial embedding* ) pada analisis sentimen berdasarkan dari model yang dihasilkan dari proses *word embedding*.

#### 4.5.2 Perancangan *Training Model Word Embedding*

Tahapan *training* model *word embedding* memiliki tujuan untuk menghasilkan model *word embedding* yang akan digunakan dalam analisis sentimen selanjutnya. Untuk melakukan hal ini, menggunakan *library* berbasis *python* bernama *gensim*. Terdapat beberapa parameter yang harus ditentukan terlebih dahulu, agar mendapatkan model yang optimal. Berikut beberapa parameter yang terdapat dalam *library gensim*.

**Tabel 4.6 Paramter Word Embedding**

<b>Paramter</b>	<b>Definisi</b>
Size	ukuran output dimensi vektor setiap kata
Sg	kode <i>training alorithm</i> , 1 <i>skip-gram</i> dan 0 untuk <i>cbows</i>
Iter	jumlah perulangan ( <i>epoch</i> ) yang diterapkan dalam proses <i>training</i>
Window	Nilai maksimal jarak antara prediksi dan aktual kata dalam 1 kalimat
seed	Untuk inisiasi nilai vektor awal setiap kata secara acak
min_count	Jumlah minimal kata dalam satu kalimat
Alpha	Nilai <i>learning-rate</i> dalam <i>training</i> model <i>word2vec</i>

Dalam penelitian ini akan membuat dua model *word2vec* dan menggunakan model *fasttext* yang telah dihasilkan dari penelitian sebelumnya.

## 4.6 Perancangan Pembuatan Model CNN

Pada tahap ini akan dilakukan perancangan untuk membuat model CNN. Data yang digunakan adalah tweets yang terkait topik dan memiliki label masing masing. Data tersebut selanjutnya dibersihkan terlebih dahulu agar memiliki perlakuan sama dengan data untuk model word embedding. Keluaran pada tahap ini merupakan model CNN dengan parameter terbaik.

### 4.6.1 Perancangan *Dataset* Model CNN

*Dataset* yang digunakan dalam CNN ini terdapat 3 buah, menyesuaikan dengan subtask yang dijelaskan sebelumnya. *Dataset* ini memiliki *tweets* yang sama hanya berbeda untuk perlakuan labelnya. Berikut kondisi label untuk setiap *subtask*.

Tabel 4.7 Perlakuan Label

Subtask	Label	Perlakuan
A	Positif, Negatif	Menghapus data yang memiliki label netral, merubah label sangat positif menjadi positif, dan merubah label sangat negatif menjadi negatif
B	Positif, Netral, Negatif	Merubah label sangat negatif menjadi negatif dan label sangat positif menjadi positif
C	Sangat Positif, Positif, Netral, Negatif, Sangat Negatif	Tidak ada perlakuan

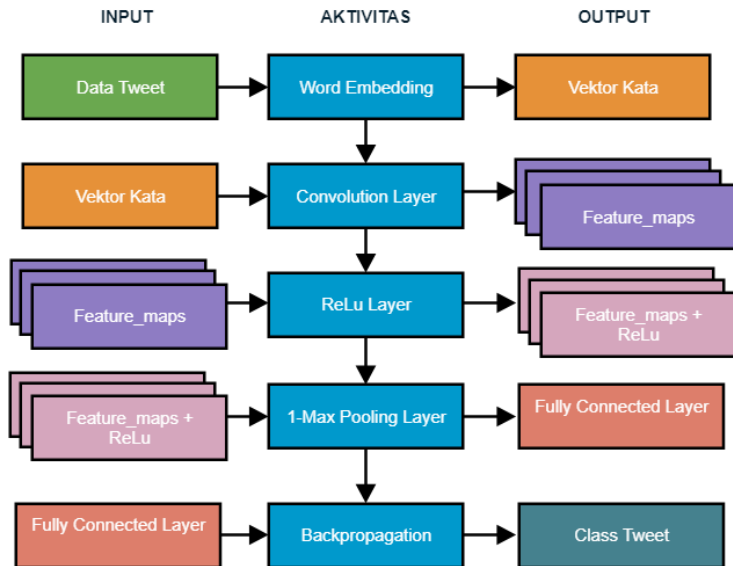
#### 4.6.2 Perancangan *Training Model CNN*

Tahap *training* model memiliki tujuan untuk menghasilkan model CNN. Dalam penerapannya terdapat *hyperparameter* yang harus ditentukan agar mendapatkan hasil terbaik. Untuk melakukan analisis sentiment dengan algoritma *CNN* menggunakan *library* berbasis *python* bernama *pytorch*. *Library* ini memiliki keunggulan dapat dijalankan di GPU sehingga menghemat durasi *training* dengan signifikan. Terdapat dua macam input utama dalam *training* model CNN, yaitu model output dari proses *word embedding* dan *datasets* yang telah disiapkan. Berikut *parameter* model CNN.

**Tabel 4.8 Paramter Model CNN**

<b>Parameter</b>	<b>Definisi</b>
Embed_num	Ukuran dimensi vektor kata dari word embedding
Label_num	Jumlah total label yang terdeteksi dalam dataset
Embed_mode	Model embedding yang akan diimplementasi dalam training ( <i>static</i> atau <i>non-static</i> )
Feature_num (Feature Maps)	Jumlah dan ukuran <i>feature maps</i> yang dihasilkan dari proses konvolusi
Kernel_width (Region Size)	Jumlah dan ukuran dari filter kata yang digunakan
Dropout_rate	Nilai index untuk proses drop out saat training model
Norm_limit	Koefisien regularisasi ( <i>l2 regularization</i> )

Selain beberapa parameter diatas, aritektur CNN memiliki alur konvolusi, dimana pada penelitian ini berikut alur konvolusi yang diterapkan.



Gambar 4.6 Alur Konvolusi

Terdapat 3 aktivitas utama dalam alur tersebut, yang pertama adalah proses konvolusi yaitu merubah vektor kata menjadi *feature maps* yang telah ditentukan jumlah dan ukuran sebelumnya. Terdapat juga *ReLU layer* dimana memiliki tujuan untuk merubah vektor yang bernilai negatif menjadi 0 dan *MaxPool* yang memiliki fungsi mengambil nilai terbesar dari setiap vektor. Parameter terhadap proses *training* model juga harus ditentukan agar dapat dibandingkan antar skenario model. Berikut beberapa parameter untuk *training* model CNN.

Tabel 4.9 Paramter Training Model CNN

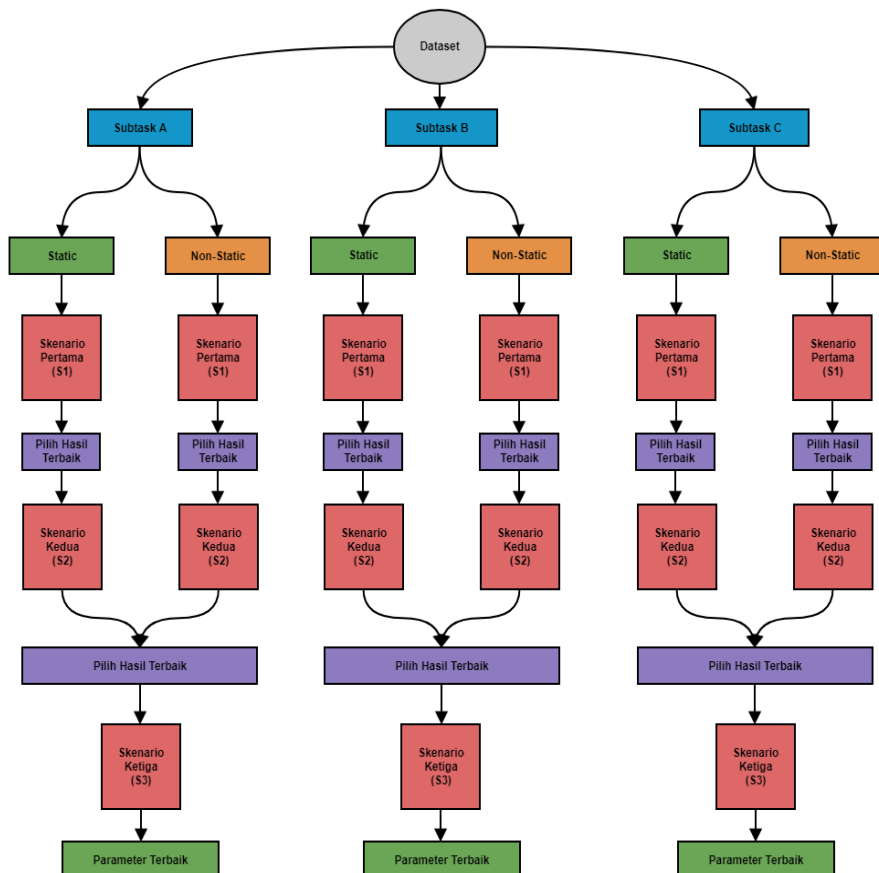
Parameter	Definisi
Epoch_num	Jumlah iterasi yang dilakukan selama training model
Fold_num	Jumlah folding data yang digunakan selama training model
Batch_size	Jumlah mini <i>batch</i> data yang digunakan selama training model
Embed_mode	Jenis model word embedding yang digunakan ( <i>word2vec</i> atau <i>fasttext</i> )

#### 4.6.3 Perancangan *Testing Model CNN*

Tahap ini berfokus pada bagaimana metode untuk menguji kualitas model yang dihasilkan. Dalam proses *training* data dibagi menjadi 10 bagian, 9 bagian digunakan untuk data *training* dan 1 bagian digunakan untuk data *testing*. Proses training dilakukan sebanyak 10 kali dengan kombinasi bagian yang berbeda untuk data *testing* dan *training* sehingga model dapat belajar lebih luas dan efektif, isitilah ini disebut *10 fold cross-validation*. Pengukuran yang dicatat setiap training model berupa akurasi, presisi, *recall*, dan *F-Measure*.

#### 4.6.4 Perancangan Skenario *Training Model CNN*

Tahap ini berfokus pada bagaimana mengatur *hyperparameter* model CNN agar mendapatkan dengan hasil terbaik. Berikut alur skenario untuk *training* model CNN.



**Gambar 4.7** Alur Skenario Training Model CNN

Pertama *dataset* dibagi 3 berdasarkan *subtask*, dimana *subtask* A kelas label berjumlah 2 buah yaitu positif dan negatif, *subtask* B berjumlah 3 buah yaitu positif, netral dan negatif sedangkan *subtask* C berjumlah 5 label yaitu sangat positif, positif, netral, negatif dan sangat negatif. Setelah dibagi berdasarkan *subtask* dibagi dengan metode *embedding* yang akan digunakan yaitu *static* dan *non-static*. Setelah itu akan dilakukan skenario



pertama (S1) yaitu menguji setiap *filter region size* dari 1 – 10 ( *single region size* ). Skenario pertama in dilakukan sebanyak 6 kali sesuai dengan jumlah *subtask* dikali metode *embedding*. Skenario kedua (S2) yaitu menguji *multiple region size* berdasarkan hasil dari skenario pertama, contoh, hasil skenario pertama terbaik adalah 4, maka *multiple region size* yang akan diuji di skenario kedua adalah 4,4,4, 2,3,4, 3,4,5 dan 4,5,6. Metode pemilihan kombinasi *region size* tersebut berkaitan dengan penelitian sebelumnya yang menyatakan bahwa kombinasi terbaik berasal dari hasil terbaik dari pemilihan *single filter region size* terbaik.

Setelah skenario kedua telah dilakukan di semua *subtask* dan *metode embedding*, dipilih hasil kombinasi terbaik dan dibandingkan antara *non-static* dengan *static* per *subtask*, sehingga menghasilkan parameter terbaik sementara untuk setiap *subtask*. Skenario ketiga (S3) adalah menguji berdasarkan model *word embedding*, dimana terdapat 3 model *word embedding* yang telah disiapkan. Hasil dari skenario ketiga merupakan parameter terbaik dari setiap *subtask*.

## BAB V IMPLEMENTASI

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

### 5.1 Lingkungan Implementasi

Pengerjaan penelitian ini menggunakan komputer dengan spesifikasi berikut.

**Tabel 5.1 Spesifikasi *Hardware***

<b>Prosesor</b>	Intel i5 8400
<b>Memory</b>	16 GB DDR4 <i>Memory</i>
<b>GPU</b>	GTX 1070 8 GB
<b>OS</b>	<i>Linux Ubuntu 16.10</i>
<b>Arsitektur</b>	64 Bit

Selain itu, terdapat beberapa library, database, bahasa pemrograman yang membantu pengerjaan penelitian ini sebagai berikut.

**Tabel 5.2 *Library* Yang Digunakan**

<b><i>Websserver</i></b>	Apache 2.4
<b>Bahasa Pemrograman</b>	Python 3.0, Javascript
<b><i>Database</i></b>	MySQL
<b>Editor</b>	Sublime, EmEditor
<b><i>Library</i></b>	<ul style="list-style-type: none"><li>• Tweepy</li><li>• Pandas</li><li>• Numpy</li><li>• Langdetect</li><li>• Mysql-connector</li><li>• Gensim</li><li>• Torchtext</li><li>• PyTorch</li></ul>

## 5.2 Pembuatan *Crawler* Twitter

Dalam penelitian ini terdapat dua macam *crawler*, yaitu untuk mendapatkan data untuk *word embedding* dan data untuk analisis sentimen. Perbedaan dari kedua *crawler* tersebut adalah *crawler* yang digunakan untuk mendapatkan data *word embedding* menggunakan API yang disediakan oleh twitter, sedangkan *crawler* yang digunakan untuk data analisis sentimen menggunakan metode *web scrapping*.

### 5.2.1 Pembuatan *Crawler Word Embedding*

Dalam pembuatan *crawler word embedding* atau disebut *crawler* 1, dalam penelitian ini menggunakan *library* tweepy. Tweepy merupakan *library* yang memiliki performa cukup bagus untuk mendapatkan *tweets* dengan jumlah yang banyak dengan memaksimalkan limit yang disediakan oleh API twitter.

```
import sys
import jsonpickle
import os
import tweepy
auth =
tweepy.AppAuthHandler('y1CA05iAK30bv1m9v2sFkkGbe', 'JB
ZB1XaXraqTcuHLLB1ZQ42YruG1EegPrhFdOA9uU1CrVqhhrq')

api = tweepy.API(auth, wait_on_rate_limit=True,

wait_on_rate_limit_notify=True)

if (not api):
    print ("Can't Authenticate")
    sys.exit(-1)
```

#### Kode 5.1 Potongan Kode Implementasi API Untuk *Crawling*

Kode 5.1 merupakan bagian untuk melakukan otorisasi agar dapat melakukan *crawling* twitter. Tweepy membutuhkan *API\_KEY* dan *API\_SECRET* yang didapat dari website twitter. Dua variabel tersebut didapatkan setelah kita mendaftarkan aplikasi kita melalui halaman *developer* twitter. Kedua variabel tersebut merupakan parameter method *AppAuthHandler()* dimana berguna untuk melakukan otorisasi berdasarkan

*API\_KEY* dan *API\_SECRET* yang dimasukkan. Kode 5.1 juga terdapat percabangan ketika gagal melakukan otorisasi maka akan keluar error.

```
keyword = "adalah"
searchQuery = input(keyword)
tweetsPerQry = 100
fName = "hasilcrawling.txt"
sinceId = None
max_id = 999999999999999999

tweetCount = 0
```

### **Kode 5.2 Potongan Kode Parameter Proses *Crawling***

Kode 5.2 merupakan variabel yang terdapat pada *crawler* ini. Variabel *keyword* berisi kata kata yang merupakan kata kunci dalam proses *crawling*, sehingga semua *tweets* yang mengandung kata tersebut akan diambil oleh *crawler* ini. Jumlah *tweets* maksimal yang diperbolehkan untuk diambil adalah 100 *tweet* per eksekusi API ( bukan 1 proses *crawling* ), dibatasi juga dalam 180 eksekusi per 15 menit dan hanya bisa mengambil tweet maksimal 1 minggu sebelumnya. Variabel *fName* menyimpan informasi nama *file* yang berfungsi sebagai media penyimpanan *tweet* dalam format *json*. *sinceId* menyimpan informasi id *tweet* minimal yang dapat diambil oleh *crawler*, namun untuk memaksimalkan proses *crawling* maka variabel tersebut diberi nilai *none*. *Max\_id* menyimpan informasi id *tweet* maksimal yang dapat diambil oleh *crawler*, namun untuk memaksimalkan proses *crawling* maka variabel tersebut diberi nilai 999999999999999999 ( tidak ada batas ).

```

with open(fName, 'w') as f:
    while True:
        try:
            new_tweets = api.search(q=searchQuery,
count=tweetsPerQry, tweet_mode='extended')

            if not new_tweets:
                print("No more tweets found")
                break

            for tweet in new_tweets:

f.write(jsonpickle.encode(tweet._json,
unpicklable=False) + '\n')

                tweetCount += len(new_tweets)
                print("Downloaded {0}
tweets".format(tweetCount))
                max_id = new_tweets[-1].id

            except tweepy.TweepError as e:

                print("some error : " + str(e))

print ("Downloaded {0} tweets, Saved to
{1}".format(tweetCount, fName))

```

### Kode 5.3 Potongan Kode Perulangan *Crawling*

Selanjutnya dibuat perulangan pertama untuk mengeksekusi method *api.search()* dengan parameter kata kunci yang diinginkan, jumlah *tweet* per eksekusi API dan *mode tweet* adalah *extended*. Alasan untuk menggunakan *extended* yaitu untuk mendapatkan seluruh *tweet* tanpa terpotong. Perulangan pertama ini akan selesai ketika tidak terdapat *tweet* dengan kata kunci tersebut, dimana akan masuk ke blok *if not new\_tweets* dan akan keluar dari perulangan. Hasil dari method *api.search()* disimpan dalam variabel *new\_tweets*. Selanjutnya dibuat perulangan kedua untuk menulis informasi yang terkandung didalam *tweet*. Untuk menulis dalam format *json* menggunakan method *jsonpickle.encode()*. Apabila terdapat error maka akan masuk ke blok *except* dan tetap melanjutkan proses *crawling*.

Tahap selanjutnya yaitu mengambil atribut yang diinginkan dan dimasukkan ke dalam *database*, untuk melakukan proses tersebut berikut kode programnya.

```
import json
import time
import datetime

timestamp = time.time()
st =
datetime.datetime.fromtimestamp(timestamp).strftime('%d%m%Y'
)
filename = 'diinginkan-c3.txt'

with open(filename, 'r') as f:
    with open('HASIL_'+ st + '_' + filename + '.sql', 'a') as
saveFile:
    for line in f:
        try:
            tweet = json.loads(line)

            if 'full_text' in tweet:
                id = tweet['id']
                created_at = tweet['created_at']
                text = tweet['full_text']

                ts = time.strftime('%Y-%m-%d %H:%M:%S',
time.strptime(tweet['created_at'],
'%a %b %d %H:%M:%S +0000 %Y'))

                abc = id, 'Tanpa Filter', text, ts
                print(text)
                print('INSERT INTO twitter VALUES
{};'.format(abc))
                saveFile.write('INSERT INTO twitter VALUES
{};'.
                    format(str(abc)))
                saveFile.write('\n')

        except:
            continue

    print('\n' + 'COMPLETE READ : {}'.
format(filename) + '\n')
```

#### Kode 5.4 Potongan Kode Untuk Penulisan Hasil *Crawling*

Proses merubah data *tweets* yang berformat *json* menjadi *syntax sql* dapat dilakukan dengan membuat perulangan sebanyak *tweets* yang disimpan didalam file sumber. Menggunakan method *json.loads()* dapat merubah format *string* menjadi objek *json* yang kemudian dapat dipilih atribut yang diinginkan. Setelah itu, ditulis kembali dengan format *insert syntax* sesuai atribut yang di inginkan, dan dapat dieksekusi didalam *server database* untuk memasukkan data.

### 5.2.2 Pembuatan Crawler Analisis Sentiment

Dalam pembuatan *crawler* analisis sentimen menggunakan metode *web scrapping* karena tidak terdapat batasan waktu sehingga data yang didapat lebih banyak. Langkah pertama untuk mendapatkan data dengan membuka halaman pencarian twitter melalui *browser* dan mengetikan kata kunci yang di inginkan.

```
var scroll = setInterval(function(){
window.scrollTo(0,5000); }, 2000);
```

#### Kode 5.5 Potongan Kode Untuk Scroll Halaman Browser

Setelah itu akses halaman *console* didalam *browser* tersebut, dan ketikan kode 5.5 agar halaman dapat secara otomatis *scroll* paling bawah. Hal ini dilakukan karena halaman pencarian twitter secara otomatis menambahkan hasil pencarian apabila halaman telah di-*scroll* hingga lokasi paling bawah. Di dalam kode program tersebut, *window.scrollTo()* berguna untuk melakukan *scroll* sebanyak 5000 pixel dengan interval 2 detik. Durasi proses crawling dapat memakan waktu 1 – 3 jam untuk mencapai halaman paling bawah. Setelah itu menyeleksi seluruh hasil pencarian di browser dengan menekan tombol *ctrl + A* dan di-*paste* di text editor sublime, berikut contoh hasilnya.

```

@Adikhshan33
3 jam3 jam yang lalu
Selengkapnya
Teu @IndosatCare teu @triindonesia eweh nu eceg pisan jaringan teh

0 balasan 0 retweet 0 suka
Balas Retweet Suka Direct message

Daniel Widjaya

@Indraocta
3 jam3 jam yang lalu
Selengkapnya
Kenapa sudah 2 malam ini jaringan internet 3 nya tidak ada jaringan, apa sedang ada perbaikan? Atau bagaimana? @triindonesia

0 balasan 0 retweet 0 suka
Balas Retweet Suka Direct message

Muhammad Fadli

@Fadli_m23
3 jam3 jam yang lalu
Selengkapnya
@triindonesia y ampun tiba2 jaringan internet ilang sendiri, ini tri lg knp? untung ada wifi

1 balasan 0 retweet 0 suka
Balas 1 Retweet Suka Direct message

Coki

```

**Gambar 5.1** Contoh Hasil *copy-paste*

Untuk mendapatkan konten dari tweets nya saja, maka dibuat kode program sehingga dapat membuat teks yang tidak diperlukan, berikut kode program nya.



```

tw = []
twfix = []
twfix2 = []
twfix3 = []

keyword = '@triindonesia'
nama_file_new = keyword+"_new.txt"

with open(nama_file+".txt",'r',encoding='utf-8')
as f:
    for x in f:
        tw.append(x)

start = 0
end = 0

for c in range(0,len(tw)):
    twit = tw[c]

    if 'Selengkapnya' in twit:
        start = c
        continue

    if 'balasan' in twit and 'retweet' in
twit:
        end = c
        temp = []
        temp.clear()

        for v in range(start+1,end-1):
            temp.append(tw[v])

        test = " ".join(temp)
        twfix.append(test)
        continue

```

#### **Kode 5.6 Potongan Kode Untuk Mendapatkan Kalimat *Tweet***

Berdasarkan pola hasil pencarian, untuk mendapatkan isi dari tweets nya saja, dapat diambil diantara baris yang mengandung kata ‘Selengkapnya’ dan baris yang mengandung kata ‘balasan’ dan ‘retweet’. Perulangan pertama dilakukan sebanyak jumlah baris didalam file, setiap mendeteksi baris yang mengandung kata ‘selengkapnya’ maka mencatat *index* tersebut, dimana

```

for g in range(0, len(twfix)):

    twit = twfix[g]
    z = len(twit)

    if z > 1 and '...' not in twit and '(cont)'
not in twit:

        twfix2.append(twit)

with open(nama_file_new, 'w', encoding='utf-8') as
h:
    for d in range(0, len(twfix2)):
        twit = twfix2[d].replace("\n", "")
        print(d, twit)
        h.write(twit)
        h.write('\n')

```

#### **Kode 5.7 Potongan Kode Untuk Menghilangkan *Tweet* Terpotong**

index ini akan menjadi *index* awal untuk mendapatkan isi tweet. Ketika mendeteksi baris yang mengandung kata ‘balasan’ dan ‘retweet’ akan disimpan pula *index* tersebut dan menjadi *index* akhir. Setiap baris diantara *index* awal dan *index* akhir akan di gabung menjadi 1 kalimat. Berbeda dengan *crawling* menggunakan API twitter, menggunakan metode *scrapping* tidak dapat memilih *mode crawling extended* atau bukan. Sehingga untuk mendapatkan *tweet* yang penuh, dapat dideteksi dengan tidak adanya karakter ‘...’ atau kata ‘(cont)’. Setelah itu hasil *crawling* ditulis dalam txt untuk diolah ditahap selanjutnya.

### **5.3 Pembuatan *Filtering* Bahasa Indonesia**

Tahap ini memiliki tujuan untuk menghapus *tweet* yang bukan bahasa indonesia. Dalam penerapannya menggunakan *library* berbasis *python* yaitu *langdetect*.

```
from langdetect import detect
from langdetect import detect_langs

import emot
import html
import os
import re
import mysql.connector

cnx = mysql.connector.connect(user='root',
password='', host='127.0.0.1',
database='word_embed')

try:
    cursor = cnx.cursor()
    cursor.execute("""
        SELECT * FROM twitter_word_embed
        """)
    result = cursor.fetchall()

finally:
    cnx.close()
```

#### **Kode 5.8 Potongan Kode Untuk Mendapatkan Data *Tweet***

Potongan kode 5.8 memiliki tujuan untuk mendapatkan data dari *database*, dengan mengambil seluruh *tweet* dari tabel *twitter\_word\_embed*. Hasil dari eksekusi query tersebut disimpan dalam bentuk array pada variabel *result*.

```

with open('cekbahasa.sql','a') as saveFile:
for num in range(0,len(result)):

cek = re.sub(r"http\S+", "", result[num][2])

try:

hasil = str(detect_langs(cek))
lang = hasil[1:3]
index = float(hasil[4:16])

if(lang == 'id' and index > 0.5 ):

idx = result[num][0]
username = result[num][3]
text = cek2
abc = idx,'Tanpa Filter',text,username

saveFile.write('INSERT INTO twitter_indo2 VALUES
{};'.
format(str(abc)))
saveFile.write('\n')

except:

print('error')

```

### Kode 5.9 Potongan Kode Untuk Mendeteksi Bahasa

Langkah selanjutnya untuk melakukan deteksi bahasa adalah membuat perulangan dengan batas akhir ukuran variabel *result*. Untuk meningkatkan efektivitas, *tweet* yang mengandung URL dihilangkan terlebih dahulu URL nya, menggunakan fitur *regex*, ketika sebuah kata terdapat frase 'http' maka kata tersebut dihilangkan. *Method detect\_langs()* dari *library langdetect* berguna untuk mendeteksi bahasa dan mengembalikan dua variabel yaitu bahasa yang terdeteksi dan indeks kandungan bahasa tersebut. Ketika indeks yang dimiliki lebih dari 0,5 dan kode bahasanya adalah 'id' maka *tweet* tersebut disimpan. Semua *tweet* yang dikategorikan bahasa indonesia disimpan dalam bentuk *syntax sql*.

## 5.4 Pembuatan Model *Word Embedding*

Tahap ini memiliki tujuan untuk membuat model *word embedding* berbasis *Word2Vec. Library* yang digunakan untuk membuat model ini adalah *gensim*. Sebelum membuat model *word embedding* terdapat proses pembersihan dataset agar *training* model *word embedding* lebih efektif.

### 5.4.1 Pembuatan Pembersihan *Dataset*

Tahap ini dilakukan sebelum *training* model *word embedding* dengan tujuan menghasilkan *dataset* yang lebih rapi.

```
def replace_URL(self, string):
    tokens = ['<url>' if 'http' in token
else token for token in string.split()]
    return ' '.join(tokens)

def replace_mention(self, string):
    tokens = ['<mention>' if
token.startswith('@') else token for token in
string.split()]
    return ' '.join(tokens)

def replace_mult_occurences(self, string):
    return re.sub(r'(\.)\1{2,}', r'\1\1',
string)
```

#### Kode 5.10 Potongan Kode Untuk *Method* Pembersihan

Potongan kode 5.10 menunjukkan proses pre-processing pertama yang dilakukan, *method replace\_url()* berguna untuk mengganti kata yang mengandung URL menjadi '<url>', *method replace\_mention()* berguna untuk merubah kata yang melakukan *mention* terhadap sebuah akun menjadi '<mention>' dan *method replace\_mult\_occurences()* berguna untuk membuat karakter yang berulang lebih dari dua buah menjadi dua buah.

```

def clean_str(self, string):

    string = re.sub(r"\.", " . ", string)
    string = re.sub(r",", " , ", string)
    string = re.sub(r":", " : ", string)
    string = re.sub(r";", " ; ", string)
    string = re.sub(r"!", " ! ", string)
    string = re.sub(r"\?", " ? ", string)
    string = re.sub(r"\(", " ( ", string)
    string = re.sub(r"\)", " ) ", string)
    string = re.sub(r"#", " <hash_tag> ",
string)
    string = re.sub(r"\[", " [ ", string)
    string = re.sub(r"\]", " ] ", string)
    string = re.sub(r"^[A-Za-z0-
9().,<_>!?\'\`]", " ", string)
    string = re.sub(r"\s{2,}", " ", string)

    return string

```

#### Kode 5.11 Potongan Kode Untuk Menghilangkan Simbol

Proses selanjutnya *method clean\_str()* memiliki tujuan yaitu menghilangkan simbol pada tweets dengan memanfaatkan *regex*.

```

def replace_emoticons(self, string):

# Campur
string = string.replace('<3', ' <hati> ')
string = string.replace(':D', ' <tertawa> ')
string = string.replace(':P', '
<menjulurkan_lidah> ')
string = string.replace(':p', '
<menjulurkan_lidah> ')
string = string.replace(':o', ' <terkejut> ')
string = string.replace(':O', ' <terkejut> ')
string = string.replace(':x', ' <cium> ')
string = string.replace(':*', ' <cium> ')
string = string.replace(':3', ' <malu-
malu_kucing> ')
string = string.replace('XD', '
<tertawa_terbahak-bahak> ')

# Senyum
string = string.replace(':)'), ' '
<senyum_senyum> ')
string = string.replace(':)', ' <senyum> ')
string = string.replace(':-)', ' '
<senyum_senyum> ')
string = string.replace(':-)', ' <senyum> ')
string = string.replace('(:', ' '
<senyum_senyum> ')
string = string.replace('((', ' <senyum> ')
string = string.replace('(:-', ' '
<senyum_senyum> ')
string = string.replace('(-:', ' <senyum> ')
string = string.replace('=)', ' '
<senyum_senyum> ')
string = string.replace('=)', ' <senyum> ')
string = string.replace('^_^', ' <senyum> ')

```

```

# Sedih
string = string.replace(':', ('', ' <sedih_sedih>
'))
string = string.replace(':', ('', ' <sedih> '))
string = string.replace(':-(', ('', ' <sedih_sedih>
'))
string = string.replace(':-(', ('', ' <sedih> '))
string = string.replace('):', ('', ' <sedih_sedih>
'))
string = string.replace('):-:', ('', ' <sedih_sedih>
'))
string = string.replace(')-:', ('', ' <sedih> '))

# Berkedip
string = string.replace(';)', ('', '
<senyum_berkedip> '))
string = string.replace(';)', ('', '
<senyum_berkedip> '))

# Tears
string = string.replace(":)"), ('', '
<menangis_bahagia> '))
string = string.replace(":)", ('', '
<menangis_bahagia> '))
string = string.replace(":'((", ('', '
<menangis_sedih> '))
string = string.replace(":'((", ('', '
<menangis_sedih> '))
string = string.replace("(:'", ('', '
<menangis_bahagia> '))
string = string.replace("(:'", ('', '
<menangis_bahagia> '))

# Some annoyed
string = string.replace('/:)', ('', ' <terganggu> '))
string = string.replace('/:\\', ('', ' <terganggu> '))

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

return string

```

**Kode 5.12 Potongan Kode Untuk Mengganti *Emoticon***



Berdasarkan potongan kode 5.12 method *replace\_emoticon()* memiliki tujuan untuk mengganti *emoticon* yang terkandung didalam *tweet* menjadi kata kata yang merepresentasikan *emoticon* tersebut. Lebih lengkapnya proses translasi *emoticon* dapat dilihat dibawah ini.

**Tabel 5.3 Translasi Emoticon**

<b>Emoticon</b>	<b>Kata Kata</b>	<b>Emoticon</b>	<b>Kata Kata</b>
<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 Datar

```

def __iter__(self):

    cnx = mysql.connector.connect(user='root',
    password='', host='127.0.0.1',
    database='word_embed')

    try:
        cursor = cnx.cursor()
        cursor.execute("""
            SELECT text,id_tweet FROM twitter_indo
            """)
        result = cursor.fetchall()

    finally:
        cnx.close()

    for num in range(0, len(result)) :

        id = result[num][1]
        message = result[num][0]
        message = self.replace_URL(message)
        message = html.unescape(message)
        message = self.replace_mention(message)
        message = self.replace_mult_occurences(message)
        message = message.replace('..', ' <elipsis> ')
        message = self.replace_emoticons(message)
        message = self.clean_str(message)
        message = message.lower()

        yield id, message

```

### Kode 5.13 Potongan Kode Urutan Pembersihan Data

Semua proses dieksekusi didalam *method* `__iter__()` dimana diawali dengan mendapatkan data dari database. Untuk mendapatkan data dari database menggunakan library *mysql-connector*. Tabel yang digunakan adalah tabel `twitter_indo` dan hasil dari eksekusi *query* tersebut disimpan dalam variabel *result*. Selanjutnya membuat perulangan sebanyak ukuran variabel *result*. Variabel *id* menyimpan dari id unik setiap *tweet* dan variabel *message* menyimpan kalimat *tweet*. Selanjutnya

memanggil keseluruhan method yang telah dijelaskan sebelumnya dan meng-*update* hasilnya didalam variabel *message*, sehingga nilai yang dikembalikan dari *method* ini adalah *tweet* yang telah bersih.

```

Awal
: indosat&nbsp;MANTAP djaja koneksinya , ngebut euy koyok jaran (goyang), cek
sini coba http://indosat.co.id@indosatCare mantap abisssss..... :)))

Replace URL
: indosat&nbsp;MANTAP djaja koneksinya , ngebut euy koyok jaran (goyang), cek
sini coba <url> @indosatCare mantap abisssss..... :)))

HTML
: indosat MANTAP djaja koneksinya , ngebut euy koyok jaran (goyang), cek sini
coba <url> @indosatCare mantap abisssss..... :)))

Mention
: indosat MANTAP djaja koneksinya , ngebut euy koyok jaran (goyang), cek sini
coba <url> <mention> mantap abisssss..... :)))

Occurence
: indosat MANTAP djaja koneksinya , ngebut euy koyok jaran (goyang), cek sini
coba <url> <mention> mantap abiss.. :)

elipsis
: indosat MANTAP djaja koneksinya , ngebut euy koyok jaran (goyang), cek sini
coba <url> <mention> mantap abiss <elipsis> :))

Emot
: indosat MANTAP djaja koneksinya , ngebut euy koyok jaran (goyang), cek sini
coba <url> <mention> mantap abiss <elipsis> <senyum_senyum>

Clean
: indosat MANTAP djaja koneksinya ngebut euy koyok jaran goyang cek sini coba
url mention mantap abiss elipsis senyum senyum

lower
: indosat mantap djaja koneksinya ngebut euy koyok jaran goyang cek sini coba
url mention mantap abiss elipsis senyum senyum

Akhir
: indosat mantap djaja koneksinya ngebut euy koyok jaran goyang cek sini coba
url mention mantap abiss elipsis senyum senyum

```

**Gambar 5.2 Contoh Hasil Pembersihan**

### 5.4.2 Pembuatan *Training Model Word Embedding*

Proses selanjutnya yaitu membuat model word *word2vec* menggunakan *library gensim*.

```
class Word2vec():
    def __init__(self):

        self.size = 300
        self.num_features = 300
        self.num_workers = multiprocessing.cpu_count()
        self.sg = 0
        self.iter = 1
        self.window = 5
        self.seed = 1
        self.min_word_count = 5
        self.context_size = 7
        self.alpha = 0.025
        self.downsampling = 1e-3
```

#### **Kode 5.14 Potongan Kode Parameter *Word Embedding***

Sebelum membuat model, langkah pertama melakukan *override* kelas *word2vec* dan memberikan *default parameter*. Masing masing parameter nantinya di-*update* ketika pemanggilan kelas *word2vec* selanjutnya.

```

if __name__ == "__main__":

    file_read = ReadDataWordEmbed()
    word = []

    for x in file_read:

        word.append(x[1].split())

    model = Word2Vec(size= 300, sg = 0, min_count = 5,
window = 5, iter = 2)

    model.build_vocab(word)

    print("\nWord2Vec vocabulary length :
{}".format(len(model.wv.vocab)))

    token_count = sum([len(sd) for sd in word])

    print("\nCorpus contains {0:,}
tokens".format(token_count))
    print("\nBuilding word2vec model...")

    start = time.time()

    model.train(word, total_examples = token_count,
epochs=model.iter)
    end = time.time()

    print("word2vec training done in {}
seconds".format(end - start))

    model.save("modelapik_cbow.bin")

    model.wv.save_word2vec_format('modelapik_cbows.bin',
binary=False)

```

#### **Kode 5.15 Potongan Kode Training Model Word Embedding**

Setelah melakukan inisiasi kelas *word2vec*, langkah selanjutnya adalah membuat objek *class ReadDataWordEmbed()* dimana merupakan kelas pembersihan tweet dan disimpan didalam variabel *file\_read*. Variabel tersebut berupa *object array* yang memiliki 2 dimensi, dimensi pertama adalah jumlah *tweet* dan dimensi kedua adalah *id* dan kalimat dari *tweet* masing masing. Selanjutnya membuat perulangan terhadap variabel *file\_read* dan memecah setiap kalimat berdasarkan kata menggunakan

*method split()*. Hasil pemecahan kalimat selanjutnya disimpan didalam variabel *word*. Selanjutnya membuat *object class word2vec* dengan parameter baru. Jumlah *vocab* yang terkandung didalam corpus kata dapat diketahui setelah membuat *vocab* terlebih dahulu menggunakan *method build\_vocab()*. Token merupakan jumlah kata yang tergantung didalam corpus sedangkan *vocab* merupakan jumlah kata yang unik didalam corpus. Proses *training* model dilakukan dengan menggunakan *method train()*. Proses *training* umumnya memakan waktu hingga 2 – 5 jam tergantung dari jumlah iterasi, jumlah token dan spesifikasi hardware yang digunakan. Untuk menyimpan model dengan format *word2vec* menggunakan *method save\_word2vec\_format()*.

## **5.5 Pembuatan Model CNN**

Tahap ini memiliki tujuan untuk membuat model CNN dengan 1 proses konvolusi. Sama seperti proses pembuatan model *word embedding*, sebelum membuat model diawali dengan pembersihan *tweet*. Aktivitas pembersihan *tweet* sama persis dengan proses sebelumnya.

### **5.5.1 Pembuatan *Training* dan *Testing* Model CNN**

Pembuatan model CNN menggunakan *library* bernama *pytorch* karena kelebihan-nya dapat menggunakan *GPU accelerator*. langkah pertama adalah menyiapkan *dataset* topik terlebih dahulu.

```
from gensim.models.keyedvectors import KeyedVectors
import numpy as np
import random
import re
import time
import torchtext
from read_data import ReadDataTopik

class prepare_data:

    def __init__(self):
        self.emb_init_value = None
        self.vocab_to_idx = None
        self.idx_to_vocab = None
        self.label_to_idx = None
        self.idx_to_label = None
        self.embed_num = 0
        self.label_num = 0
        self.train_set = None
        self.dev_set = None
```

**Kode 5.16 Potongan Kode Parameter Kelas *Prepare\_Data***

Potongan kode 5.16 merupakan inisiasi dari kelas *prepare\_data*. Kelas ini memiliki tujuan untuk menyiapkan *dataset* beserta label dari setiap tweet di *dataset*.

```

def read_dataset(self, type, subtask):

    data_topik = ReadDataTopik(subtask)

    twt_id_field = torchtext.data.Field(use_vocab=False,
    sequential=False)
    label_field = torchtext.data.Field(sequential=False)
    text_field = torchtext.data.Field()

    fields = [('twt_id', twt_id_field), ('label',
    label_field), ('text', text_field)]

    self.fields = fields

    examples = [torchtext.data.Example.fromlist([twt_id,
    self.polarity_to_label(polarity), text], fields) for
    twt_id, polarity, text in data_topik]

    self.examples = examples

```

#### Kode 5.17 Potongan Kode Pemanggilan Kelas *Readdatatopik*

*Method* ini memiliki tujuan untuk memanggil kelas *ReadDataTopik* dimana merupakan kelas pembersihan tweet sama seperti di proses pembuatan model *word embedding*. Setelah itu membuat variabel baru berjenis *field* didalam dataset dengan memanfaatkan *library torchtext*. Terdapat 3 field yaitu *twt\_id\_field* yang memiliki nilai id unik setiap *tweet*, *label\_field* yang memiliki nilai label setiap *tweet* dan *text\_field* yaitu menyimpan kalimat dari *tweet*. Tahap selanjutnya yaitu memasukkan kalimat *tweet*, id *tweet* dan label *tweet* yang didapat dari variabel *data\_topik* menggunakan perulangan sederhana, hasil tersebut disimpan didalam variabel *examples*.



```

def polarity_to_label(self, polarity):

    if int(polarity) == 1:
        label = 'strong negative'
    elif int(polarity) == 2:
        label = 'negative'
    elif int(polarity) == 4:
        label = 'positive'
    elif int(polarity) == 5:
        label = 'strong positive'
    else:
        label = 'neutral'

    return label

```

### Kode 5.18 Potongan Kode Untuk Merubah Label Menjadi Kata

*Method* `polarity_to_label()` memiliki tujuan untuk melakukan translasi dari kode label numerik menjadi bentuk kata. *Method* ini dipanggil didalam *method* sebelumnya yaitu `read_dataset()`.

```

from gensim.models.keyedvectors import KeyedVectors
from gensim.models.wrappers import FastText
class Embedding():

    def __init__(self):
        self.embed_type = None
        self.embed_dim = 0

    def read_model(self, type, cat):

        if cat == 'w2v':
            embed_path = 'w2v_indo.bin'
            print('Loading Word2Vec')
            word2vec_model =
KeyedVectors.load_word2vec_format(embed_path,
binary=False, unicode_errors='ignore')
        else:
            embed_path = 'bjn.bin'
            print('Loading Fasttext')
            word2vec_model =
FastText.load_fasttext_format(embed_path)

        self.word2vec = word2vec_model
        self.embed_dim = 300
        return word2vec_model

```

### Kode 5.19 Potongan Kode Untuk Membaca Model *Word Embedding*

Kelas *Embedding* merupakan kelas yang memiliki tujuan untuk *loading* model *word embedding* baik itu dari *word2vec* atau *fasttext*. *Method* yang digunakan untuk *loading* model berbasis *word2vec* adalah *load\_word2vec\_format()* yang dipanggil melalui objek *KeyedVectors*. Didalam *method* tersebut terdapat beberapa parameter yang dibutuhkan yaitu lokasi model *word2vec* disimpan, metode format *binary* dan pengabaian error yang terjadi didalam proses *encoding* karakter. Untuk melakukan *loading* model *fasttext* menggunakan method *load\_fasttext\_format()* dan menggunakan objek *FastText*. Berbeda dengan *method* untuk *loading* model *word2vec* sebelumnya, *method load\_fasttext\_format()* hanya membutuhkan lokasi penyimpanan model saja. Dua kelas diatas yaitu kelas *Embedding* dan kelas *prepare\_data* akan dipanggil dikelas utama yaitu kelas *main*, berikut kode programnya.

```
import prepare_data
import read_embed

def execute(width, feature, batch_size, subtask):

    in_data = prepare_data.prepare_data()
    in_data.read_dataset(args.data, subtask)

    embedding = read_embed.Embedding()
    key_vector = embedding.read_model(args.data,
args.embed)
    args.embedding_dim = embedding.embed_dim_dim =
embedding.embed_dim

    known_word, unknown_word, kim2014 =
cross_validate(args.fold_num, in_data,
embedding, args, key_vector, width, feature,
batch_size, args.embedding_mode, subtask)
```

#### **Kode 5.20 Potongan Kode Untuk *Method Execute()***

*Method execute()* merupakan method untuk melakukan proses persiapan data, training dan testing model CNN. Memiliki 4 parameter utama untuk mempermudah training dengan banyak

kombinasi parameter yang berbeda di setiap skenarionya. Variabel *in\_data* merupakan objek kelas *prepare\_data* dan mengakses method *read\_dataset()*. Variabel *embedding* merupakan objek kelas *Embedding*. Variabel *key\_vector* merupakan objek berupa model *word embedding* yang merupakan hasil *return* dari method *read\_model()*. Variabel *args.embedding\_dim* merupakan dimensionalitas vektor kata yang digunakan dalam model *word embedding*. Method *cross\_validate()* memiliki tujuan untuk melakukan *training* dan *testing* model dengan *10 folding* data.

```
def cross_validate(fold, data, embedding, args,
                  key_vector, kernel_width, feature_num,
                  batch_size, embedding_mode, subtask):

    actual_counts      = defaultdict(int)
    predicted_counts   = defaultdict(int)
    match_counts       = defaultdict(int)

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

    for i in range(fold):

        train_examples = data.examples[:]
        del
        train_examples[i*split_width:min(len(data.examples),
        (i+1)*split_width)]
        test_examples =
        data.examples[i*split_width:min(len(data.examples),
        (i+1)*split_width)]

        train_counts = defaultdict(int)
        test_counts = defaultdict(int)

        for example in train_examples:
            train_counts[example.label] += 1

        for example in test_examples:
            test_counts[example.label] += 1

    . . .
```

#### Kode 5.21 Potongan Kode Transformasi Data

Variabel *actual\_counts*, *predicted\_counts* dan *match\_counts* merupakan objek bertipe *defaultdict*. Variabel *split\_width*

merupakan nilai pembagian data di setiap bagian, untuk mendapatkan nilai tersebut dengan membagi total data dengan jumlah *fold* data yang ditentukan. Dalam penelitian ini menggunakan 10 *fold* data, sehingga semisal terdapat 10000 data/*tweets* maka disetiap bagian data adalah 1000 data dengan proporsi *training* data 9000 dan *testing* data 1000. Setelah itu dibuat perulangan sebanyak *fold* yang ditentukan, hal ini menunjukkan bahwa *training* akan diulang sebanyak *fold* yang ditentukan. Fungsi *del* adalah mengurangi 1 bagian data di variabel *train\_example* dimana merupakan data *training*, sedangkan variabel *test\_example* merupakan bagian data yang dihilangkan didalam data *training* sebelumnya, sehingga bagian data tersebut tidak ikut proses *training* model di *fold* pertama. Variabel *train\_counts* menyimpan informasi jumlah data disetiap label untuk dataset *training*, sedangkan *test\_count* menyimpan informasi jumlah data disetiap label untuk dataset *testing*.

```

Number of Train Examples : 9000
  Train-Strong Positive = 1807 (20.08%)
  Train-Positive       = 1831 (20.34%)
  Train-Neutral        = 1803 (20.03%)
  Train-Negative       = 1745 (19.39%)
  Train-Strong Negative = 1814 (20.16%)

Number of Test Examples : 1000
  Test-Strong Positive = 179 (17.9%)
  Test-Positive       = 205 (20.5%)
  Test-Neutral        = 195 (19.5%)
  Test-Negative       = 232 (23.2%)
  Test-Strong Negative = 189 (18.9%)

```

Gambar 5.3 Contoh Hasil Pembagian Data

```

def cross_validate(fold, data, embedding, args,
                  key_vector, kernel_width, feature_num,
                  batch_size, embedding_mode, subtask):

    . . .

    fields = data.fields

    train_set =
    torchtext.data.Dataset(examples=train_examples,
                           fields=fields)
    test_set =
    torchtext.data.Dataset(examples=test_examples,
                           fields=fields)

    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

    . . .

```

#### **Kode 5.22 Potongan Kode Mendapatkan Label Dan Kalimat *Tweet***

Selanjutnya, masih di *method cross\_validate()* membuat objek *field dataset* berupa kalimat *tweet*, id *tweet* dan label di *tweet* ( telah dibuat di kelas *prepare\_data* ). Setelah itu membuat objek baru berupa *dataset* menggunakan *library torchtext*. Dalam pembuatan objek *dataset* ini membutuhkan parameter yaitu kumpulan data dan *field* data.

```

def cross_validate(fold, data, embedding, args,
                  key_vector, kernel_width, feature_num,
                  batch_size, embedding_mode, subtask):
    . . .

    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)

    known_word, unknown_word, emb_init_values =
    data.create_fold_embedding(embedding, args,
                              key_vector)
    emb_init_values = np.array(emb_init_values)
    . . .

```

### Kode 5.23 Potongan Kode Untuk Membuat *Vocabulary*

Selanjutnya, masih di *method cross\_validate()* membuat *vocab* berdasarkan data *training* baik itu dari kalimat *tweet* dan label *tweet*. *method build\_vocab()* berguna untuk menghapus kata yang duplikat dari *dataset* yang diberikan, sedangkan *method dict()* berguna untuk memberikan id unik untuk setiap kata. Hasil dari *method dict()* disimpan didalam variabel *data.vocab\_to\_idx* untuk kalimat *tweet* sedangkan *data.label\_to\_idx* untuk label *tweet*. Variabel *data.idx\_to\_vocab* dan variabel *idx\_to\_label* adalah kebalikan dari variabel sebelumnya. Setelah itu mengakses *method create\_fold\_embedding()* yang terdapat di kelas *prepare\_data*. Nilai yang dikembalikan adalah jumlah kata yang terdapat didalam *vocabulary* model *word embedding*, jumlah kata yang tidak terdapat didalam *vocabulary* model *word embedding* dan

vektor kata dengan dimensionalitas sesuai model *word embedding* untuk setiap katanya.

```
def create_fold_embedding(self, embedding, args,
key_vector):

    emb_init_values = []

    a = 0
    b = 0

    for i in range(self.idx_to_vocab.__len__()):
        word = self.idx_to_vocab.get(i)
        if word == '<unk>':

emb_init_values.append(np.random.uniform(-0.25, 0.25,
args.embedding_dim).astype('float32'))

        elif word == '<pad>':

emb_init_values.append(np.zeros(args.embedding_dim).as
type('float32'))

        elif word in key_vector.wv.vocab:

emb_init_values.append(key_vector.wv.word_vec(word))
            b = b+1
        else:

emb_init_values.append(np.random.uniform(-0.25, 0.25,
args.embedding_dim).astype('float32'))
            a = a+1

    self.emb_init_values = emb_init_values

    known_word = b
    unknown_word = a

    return known_word, unknown_word,
emb_init_values
```

#### **Kode 5.24 Potongan Kode Untuk Melakukan Proses Pembobotan Kata**

Langkah pertama dalam pemberian bobot kata yaitu membuat perulangan sebanyak jumlah kata yang terdapat didalam *vocabulary* ( variabel *self.idx\_to\_vocab* ), selanjutnya membuat percabangan dengan kondisi apabila kata tersebut terdapat

didalam *vocabulary* model *word embedding*, maka kata tersebut akan diberikan bobot sesuai model *word embedding*. Apabila tidak terdapat maka akan diberi nilai acak sesuai dimensi yang diberikan. Variabel *emb\_init\_values* akan menyimpan pembobotan awal yang diberikan oleh model *word embedding*, akan di *convert* menjadi *numpy array*.

```
def cross_validate(fold, data, embedding, args,
                  key_vector, kernel_width, feature_num,
                  batch_size, embedding_mode, subtask):
    . . .

    train_iter, test_iter =
    torchtext.data.Iterator.splits((train_set,
    test_set), batch_sizes=(batch_size,
    len(test_set)), device=-1, repeat=False)

        train_bulk_dataset = train_set,
        train_bulk_size = len(train_set),

        train_bulk_iter =
    torchtext.data.Iterator.splits(datasets=train_bulk_dat
    aset, batch_sizes=train_bulk_size, device=-1,
    repeat=False)[0]

    . . .
```

#### Kode 5.25 Potongan Kode Untuk Mendapatkan Data *Training*

Selanjutnya, masih di *method cross\_validate()* akan dilakukan aktivitas pemecahan *dataset training* menjadi mini *batch* untuk meningkatkan performa proses *training* model nantinya. Method *Iterator.splits()* dari *library torchtext* berguna untuk aktivitas ini. Mini *batch* hanya diaplikasikan pada *train\_set* atau *dataset training*. Hal ini dapat dilihat dari parameter kedua didalam *method iterator.splits()*, variabel *batch\_size* ditempatkan diurutan pertama saja, sedangkan ditempat kedua berisi jumlah data *test\_set*, dengan kata lain tidak dibuat mini *batch*. Variabel *batch\_size* menyimpan ukuran default mini *batch* yaitu 50 data/*tweet*. Selain memecah dataset menjadi mini batch method



*iterator.splits()* akan merubah kata didalam kalimat tweets menjadi id di *vocabulary* nya. Hasil dari method tersebut disimpan didalam *train\_iter* dan *test\_iter*. Variabel *train\_bulk\_iter* merupakan dataset *training* namun tidak dipecah menjadi mini *batch* atau diperlakukan sama dengan data *testing*. Variabel tersebut dibuat karena untuk proses *testing* akan dilakukan dua kali setiap *epoch*, menggunakan dataset *testing* (variabel *test\_iter*) dan dataset *training* (variabel *train\_bulk\_iter*).

```
def cross_validate(fold, data, embedding, args,
                  key_vector, kernel_width, feature_num,
                  batch_size, embedding_mode, subtask):
    . . .

    kim2014 = model.CNN_Kim2014(embed_num, label_num -
                                1, args.embedding_dim,
                                embedding_mode, emb_init_values, kernel_width,
                                feature_num)

    kim2014.cuda()

    trained_model = train(kim2014, train_iter,
                          test_iter, data.label_to_idx, data.idx_to_label,
                          train_bulk_iter, i, subtask)

    return known_word, unknown_word, kim2014
```

#### Kode 5.26 Potongan Kode Untuk Training Data

Selanjutnya, masih di *method cross\_validate()* akan dilakukan inisiasi model dengan mengakses kelas *CNN\_Kim2014*. Model ini mengikuti dari penelitian yon kim di tahun 2014 dengan proses 1 konvolusi.

```
class CNN_Kim2014(nn.Module):

    def __init__(self, embed_num, label_num,
                 embedding_dim, embedding_mode, initial_embedding,
                 kernel_width, feature_num):

        super(CNN_Kim2014, self).__init__()

        self.embed_num      = embed_num
        self.label_num      = label_num
        self.embed_dim      = 300
        self.embed_mode     = embedding_mode
        self.channel_in     = 1
        self.feature_num    = feature_num
        self.kernel_width  = kernel_width
        self.dropout_rate   = 0.5
        self.norm_limit     = 3

        . . .
```

#### Kode 5.27 Potongan Kode Parameter Model CNN

Kelas model ini membutuhkan beberapa parameter utama seperti jumlah vocabulary, dimensionalitas vektor kata, jumlah label yang terdapat didalam dataset, model embedding, Jumlah dan ukuran *feature maps* yang dihasilkan dari proses konvolusi, Jumlah dan ukuran dari filter kata yang digunakan dan parameter lainnya.

```

class CNN_Kim2014(nn.Module):

    def __init__(self, embed_num, label_num, embedding_dim,
embedding_mode, initial_embedding, kernel_width,
feature_num):

    . . .

    assert (len(self.feature_num) == len(self.kernel_width))

        self.kernel_num = len(self.kernel_width)
        self.embedding = nn.Embedding(self.embed_num,
self.embed_dim, padding_idx=1)

self.embedding.weight.data.copy_(torch.from_numpy(initial_
embedding))
    if self.embed_mode == 'static':
        self.embedding.weight.requires_grad = False

        convo = [nn.Conv1d(self.channel_in,
self.feature_num[i],self.embed_dim*self.kernel_width[i],
stride=self.embed_dim) for i in range(self.kernel_num)]
        self.convs = nn.ModuleList(convo)
        self.linear = nn.Linear(sum(self.feature_num),
self.label_num)

```

### Kode 5.28 Potongan Kode Kelas Model CNN

Selanjutnya, fungsi `assert` adalah untuk membuat percabangan ( seperti `if-else` ), dengan tujuan untuk mengecek jumlah *filter* yang ditentukan sesuai dengan jumlah *feature maps* yang nantinya dihasilkan. Setelah itu membuat objek *embedding* dengan memanfaatkan *method* `nn.Embedding()` dari *library* `torch.nn`. *Method* ini membutuhkan dua parameter utama yaitu, jumlah *vocab* dan dimensionalitas vektor serta parameter tambahan yaitu id kata “<pad>” berada. Parameter *padding\_idx* penting karena, ukuran vektor dalam 1 kalimat / *tweet* menyesuaikan kalimat dengan kata terbanyak dalam 1 mini *batch*. Ketika kalimat memiliki kata kurang dari jumlah kata maksimal maka akan diberikan kata tambahan yaitu “<pad>”. Kata “<pad>” tersebut memiliki id unik yaitu 1 didalam *vocab*.

```
Embedding(14318, 300, padding_idx=1)
```

Gambar 5.4 Objek Untuk *Embedding*

Contoh dari kelas tersebut, memiliki parameter jumlah *vocab* adalah 14318 dengan dimensi vektor kata 300 dan id kata “<pad>” adalah 1.

Langkah selanjutnya yaitu memberikan bobot awal setiap kata menggunakan hasil dari model *word embedding*. Hal ini dapat dilakukan menggunakan *method weight.data.copy\_()* dengan parameter model *word embedding* yang telah di-*convert* menjadi format *array numpy*. Setelah itu membuat percabangan dengan kondisi apabila model *embedding* yang digunakan adalah *static* maka tidak menggunakan *gradient*. *Stochastic gradient* yang digunakan adalah *adadelta* dimana memiliki tujuan untuk melakukan optimasi training disetiap *epoch*.

Langkah selanjutnya membuat objek untuk melakukan proses konvolusi dimana menggunakan *method nn.Conv1d()* dari *library torch.nn*. Parameter pertama yang digunakan adalah jumlah *channel* yang digunakan, dimana jumlahnya adalah 1, karena pada umumnya CNN digunakan untuk gambar yang memiliki 3 *channel* ( RGB ). Parameter selanjutnya adalah ukuran *feature maps* yang digunakan. Parameter selanjutnya adalah ukuran *filter* yang digunakan, namun ukuran *filter* akan dikalikan dimensional vektor karena kata kata dari *tweet* telah diubah dalam bentuk vektor. Parameter terakhir adalah ukuran *stride*, *stride* merupakan jarak perpindahan disetiap pergerakan *filter*. Jumlah lapisan konvolusi sesuai jumlah *filter* yang digunakan, apabila menggunakan *single-region* maka jumlah lapisan konvolusinya adalah satu buah, sedangkan jika *multi-region* akan menyesuaikan jumlahnya. Maka dari itu variabel *convo* akan menyimpan dalam bentuk *array* dan akan ditransformasi menjadi bentuk daftar modul menggunakan *method nn.ModuleList()*

```

Layer Pertama
Feature Maps : 100 Filter Region Size : 4
Conv1d(1, 100, kernel_size=(1200,), stride=(300,))

Layer Kedua
Feature Maps : 100 Filter Region Size : 5
Conv1d(1, 100, kernel_size=(1500,), stride=(300,))

Fungsi Linear
Linear(in_features=200, out_features=5, bias=True)

```

Gambar 5.5 Layer Konvolusi

Contoh gambar 5.5 menggunakan multi-region filter yaitu 4 dan 5 dengan masing masing memiliki feature maps 100, sehingga akan dihasilkan dua filter nantinya. Setelah itu membuat fungsi *linear* dengan menggunakan method *nn.linear()* dengan parameter dari lapisan *fully-connected* dan dipetakan menjadi sejumlah label yang terdeteksi. Contoh diatas untuk fungsi linear membutuhkan lapisan input dengan ukuran 50 X 200 dimensi dan akan dihasilkan menjadi 50 X 5, 50 merupakan ukuran disetiap mini *batch*.

Kembali lagi ke kelas *main* dan di *method cross\_validate()*, model akan melalui proses training menggunakan *method train()*.

```

def train(model, train_iter, test_iter, label_to_idx,
          idx_to_label, train_bulk_iter, fold, subtask):

    parameters = filter(lambda p: p.requires_grad,
                        model.parameters())
    optimizer = torch.optim.Adadelta(parameters)
    model.cuda()
    model.train()

    . . .

```

Kode 5.29 Potongan Kode Untuk Merubah *State Model*

Variabel *parameters* menyimpan hasil dari inisiasi awal model CNN dalam bentuk iterator, nilai variabel ini akan selalu update selama proses training.

```

Deskripsi Model
CNN_Kim2014(
  (embeddings): Embedding(14318, 300, padding_idx=1)
  (convs): ModuleList(
    (0): Conv1d(1, 100, kernel_size=(1200,), stride=(300,))
    (1): Conv1d(1, 100, kernel_size=(1500,), stride=(300,))
  )
  (linear): Linear(in_features=200, out_features=5, bias=True)
)

Inisiasi Awal
convs.0.weight
convs.0.bias
convs.1.weight
convs.1.bias
linear.weight
linear.bias

```

**Gambar 5.6 Hasil Dari Parameter Model**

Variabel tersebut menyimpan bobot awal dan bias dari setiap lapisan yang dideklarasikan di kelas model sebelumnya. Setelah itu, melakukan optimisasi menggunakan *method torch.optim.Adadelta()* dengan parameter method adalah variabel *parameters*. Setelah itu membuat seluruh proses training dijalankan di GPU dengan *method model.cuda()* dan membuat *state* model dalam bentuk *training* dengan *method model.train()* yang berasal dari *library pytorch*.

```

def train(model, train_iter, test_iter, label_to_idx,
          idx_to_label, train_bulk_iter, fold, subtask):
    . . .
    for epoch in range(1, args.epoch_num+1):
        corrects_sum = 0
        go = time.time()

        for batch in train_iter:
            text_numerical, target = batch.text, batch.label

            if args.cuda:
                text_numerical, target =
                text_numerical.cuda(), target.cuda()

```

**Kode 5.30 Potongan Kode Untuk Perulangan Setiap Epoch**

Selanjutnya membuat perulangan sebanyak *epoch* yang ditentukan dan didalam perulangan tersebut dibuat perulangan kembali sebanyak jumlah mini *batch* yang ada. Selanjutnya, melakukan ekstraksi kalimat *tweet* dan label masing masing dari *dataset*.

```
def train(model, train_iter, test_iter, label_to_idx,
          idx_to_label, train_bulk_iter, fold, subtask):
    . . .

    for batch in train_iter:
        . . .

        text_numerical.data.t_()
        target.data.sub_(1)
        optimizer.zero_grad()
        forward = model(text_numerical)
        loss = F.cross_entropy(forward, target)
        loss.backward()
        optimizer.step()

        corrects = (torch.max(forward,
1) [1].view(target.size()).data == target.data).sum()
        accuracy = 100.0 * corrects / batch.batch_size
```

### Kode 5.31 Potongan Kode Mekanisme *Training Data*

*Dataset* akan di *transpose* agar sesuai dengan urutan kata sesungguhnya, dan selanjutnya mengurangi label sebanyak 1 karena label terakhir adalah <unk>. Selanjutnya menghilangkan *gradient* terlebih dahulu sebelum mengakses method *forward()* di kelas model ( kelas *CNN\_Kim2014*).

```

def forward(self, input):

    batch_width = input.size()[1]
    x = self.embedding(input).view(-1, 1,
self.embed_dim*batch_width)

    conv_results = [

        F.max_pool1d(F.relu(self.convs[i](x)),
batch_width - self.kernel_width[i] + 1).view(-1,
self.feature_num[i])

        for i in range(len(self.feature_num))
    ]

    x = torch.cat(conv_results, 1)
    x = F.dropout(x, p=self.dropout_rate,
training=self.training)
    x = self.linear(x)

    return x

```

#### Kode 5.32 Potongan Kode Method Forward()

*Method forward()* merupakan *method* bersifat *abstract* didalam kelas model, sehingga method harus di *override* terlebih dahulu dan disesuaikan dengan alur konvolusi yang digunakan. Langkah pertama adalah menerapkan proses *embedding* terhadap data *training* yang digunakan.

```

Max Word          : 47
Batch             : 50 X 47
Embed             : 50 X 1 X 14100

Filter Pertama
Convo             : 50 X 100 X 44
ReLu              : 50 X 100 X 44
Pooling          : 50 X 100 X 1

Filter Kedua
Convo             : 50 X 100 X 43
ReLu              : 50 X 100 X 43
Pooling          : 50 X 100 X 1

Fully Connected  : 50 X 200
linear           : 50 X 5

```

Gambar 5.7 Hasil Method Forward()



Proses *embedding* merupakan proses merubah kata menjadi vektor kata. Berdasarkan contoh diatas, jumlah kata terbanyak didalam satu *tweet* adalah 47, sehingga didalam *batch* tersebut memiliki dimensi  $50 \times 47$  dimana 50 merupakan ukuran *batch*. Dimensionalitas vektor yang digunakan adalah 300 sehingga setelah masuk didalam proses *embedding* ukuran vektor berubah menjadi  $50 \times 1 \times 14100$  dimana 14100 merupakan hasil perkalian 47 dengan 300, saat ini setiap kata dalam *tweet* telah menjadi vektor kata. Selanjutnya masuk kedalam proses konvolusi filter pertama dimana menggunakan variabel *self.convs* yang telah di inisiasi sebelumnya.

Output dari proses ini disebut *feature maps* dengan spesifikasi yang telah ditentukan yaitu  $50 \times 100 \times 44$ . Nilai 100 didapatkan dari parameter *feature maps* yang telah di inisiasi sebelumnya sedangkan 44 didapatkan dari perhitungan  $(47 - 4) + 1$ , dimana 4 merupakan ukuran *filter* yang diterapkan dan begitu juga untuk *filter* kedua dengan ukuran 5. Sehingga setelah proses konvolusi pertama didapatkan dua *feature maps* dengan masing masing memiliki ukuran  $50 \times 100 \times 44$  dan  $50 \times 100 \times 43$ .

Proses aktivasi dilakukan setelahnya dengan fungsi *ReLU*, proses ini tidak mempengaruhi bentuk vektor karena hanya merubah nilai negatif menjadi 0, sehingga bentuk vektor output dari proses ini sama dengan sebelumnya. Proses aktivasi dilakukan dengan bantuan *method relu()* dari *library nn.Functional()*. Langkah selanjutnya yaitu *pooling* layer menggunakan *1-max pooling*. Proses ini akan mengambil nilai terbesar dari setiap dimensi kedua didalam *feature maps*, sehingga menghasilkan bentuk vektor  $50 \times 100 \times 1$  dan  $50 \times 100 \times 1$ . Proses *max-pooling* menggunakan *method max\_pool1d()* dari *library nn.Functional()*.

Selanjutnya menggabungkan vektor dari kedua *feature maps* tersebut menjadi 1 vektor dengan ukuran  $50 \times 200$ , vektor ini disebut *fully-connected* layer. Vektor terakhir ini akan dimasukkan didalam fungsi linear yang akan dipetakan menjadi

label yang ditentukan, sehingga output dari proses ini adalah vektor 50 X 5.

```
def train(model, train_iter, test_iter, label_to_idx,
          idx_to_label, train_bulk_iter, fold, subtask):
    . . .

    for batch in train_iter:
        . . .

        loss = F.cross_entropy(forward, target)
        loss.backward()
        optimizer.step()

        corrects = (torch.max(forward,
1) [1].view(target.size()).data == target.data).sum()
        accuracy = 100.0 * corrects / batch.batch_size
```

#### Kode 5.33 Potongan Kode Untuk Mendapatkan Prediksi Kalimat

Langkah selanjutnya melakukan proses cross entropy dari *tweet* menggunakan *method cross\_entropy()* dari *library nn.Functional()*. *Method torch.max()* berfungsi untuk mendapatkan label prediksi dengan input hasil fungsi linier dari kelas model. Hasil yang didapatkan berupa loss dari setiap training dan nilai akurasi. Nilai akurasi ini dihitung berdasarkan data dari mini *batch* tersebut ( bukan dari data *testing* ).

```

def train(model, train_iter, test_iter, label_to_idx,
          idx_to_label, train_bulk_iter, fold, subtask):
    . . .
    actual, predicted, acc = evaluate(model,
                                     train_bulk_iter, 'training', epoch, fold)

    epoch_actual_counts, epoch_predicted_counts,
    epoch_match_counts, actual, predicted =
    calculate_fold_counts(actual, predicted, label_to_idx,
                          idx_to_label, 'training')

    display_measures(acc, epoch_actual_counts,
                     epoch_predicted_counts,
                     epoch_match_counts, actual, predicted, idx_to_label, 'train
                     ing', epoch, fold, go, subtask)
    actual, predicted, acc = evaluate(model,
                                     test_iter, 'testing', epoch, fold)

    epoch_actual_counts, epoch_predicted_counts,
    epoch_match_counts, actual, predicted =
    calculate_fold_counts(actual, predicted, label_to_idx,
                          idx_to_label, 'testing')

    display_measures(acc, epoch_actual_counts,
                     epoch_predicted_counts,
                     epoch_match_counts, actual, predicted, idx_to_label, 'test
                     ing', epoch, fold, go, subtask)

    actual, predicted, acc = evaluate(model,
                                     test_iter, 'testing', epoch, fold)

    epoch_actual_counts, epoch_predicted_counts,
    epoch_match_counts, actual, predicted =
    calculate_fold_counts(actual, predicted, label_to_idx,
                          idx_to_label, 'testing')

    display_measures(acc, epoch_actual_counts,
                     epoch_predicted_counts,
                     epoch_match_counts, actual, predicted, idx_to_label, 'test
                     ing', epoch, fold, go, subtask)

```

#### **Kode 5.34 Potongan Kode Untuk Menampilkan Statistik *Training***

Setelah perulangan terkait mini *batch* disetiap *epoch* selesai dilakukan, maka semua data training telah dimasukkan kedalam algoritma untuk *epoch* pertama. Setiap akan dievaluasi dua kali, menggunakan data *training* yang sama dan menggunakan data

*testing*. *Method evaluate()* memiliki dua parameter utama yaitu *model* dan *data\_iter*.

```
def evaluate(model, data_iter, type, epoch, fold):

    model.eval()
    corrects, avg_loss = 0, 0

    data_iter.sort_key = lambda x: len(x.text)

    for batch in data_iter:

        text_numerical, target = batch.text, batch.label

        if args.cuda:
            text_numerical, target = text_numerical.cuda(),
            target.cuda()

        text_numerical.data.t_()
        target.data.sub_(1)

        forward = model(text_numerical)
        loss = F.cross_entropy(forward, target,
            size_average=False)

        avg_loss += loss.data[0]
        corrects += (torch.max(forward,
            1)[1].view(target.size()).data == target.data).sum()

    size = len(data_iter.dataset)
    avg_loss = avg_loss/size

    accuracy = 100.0 * corrects/size

    cor = corrects

    acc = 100 * (cor/size)

    if type == 'testing':

        track_accuracy.append([fold+1, epoch, acc])

    return target.data, torch.max(forward,
```

### Kode 5.35 Potongan Kode *Method Evaluate()*

Parameter *type* berguna untuk mendeteksi data apa yang sedang digunakan dalam proses evaluasi apakah data training atau data testing, sedangkan parameter *epoch* dan *fold* menunjukkan status

*epoch* dan *fold* saat itu. Langkah selanjutnya membuat *state* model dengan method *model.eval()*. Proses selanjutnya mendapatkan kalimat *tweet* dari *dataset* beserta label aktualnya dan kembali mengakses *method forward()* didalam kelas model untuk mendapatkan label prediksi. Langkah selanjutnya melakukan proses *cross entropy* dari *tweet* menggunakan *method cross\_entropy()* dari *library nn.Functional()*. Hasil yang didapatkan berupa *loss* dari setiap *testing* dan nilai akurasi. Nilai akurasi disimpan didalam variabel *accuracy*. Nilai akurasi tersebut akan disimpan didalam variabel *track\_accuracy*.

```
def calculate_fold_counts(actual, predicted,
                          label_to_idx, idx_to_label, type):

    assert len(actual) == len(predicted)

    fold_actual_counts = defaultdict(int)
    fold_predicted_counts = defaultdict(int)
    fold_match_counts = defaultdict(int)

    for i in range(len(actual)):

        idx = actual[i] + 1
        label = idx_to_label[idx.item()]
        fold_actual_counts[label] += 1

        if actual[i] == predicted[i]:
            fold_match_counts[label] += 1

    for i in range(len(predicted)):

        idx = predicted[i] + 1
        label = idx_to_label[idx.item()]
        fold_predicted_counts[label] += 1

    return fold_actual_counts, fold_predicted_counts,
           fold_match_counts, actual, predicted
```

**Kode 5.36 Potongan Kode Method *Calculate\_Fold\_Counts()***

*Method calculate\_fold\_counts()* memiliki tujuan untuk mendapatkan statistic jumlah aktual, prediksi dan jumlah kecocokan antara aktual prediksi untuk setiap label yang ada. Untuk mendapatkan jumlah tersebut dengan membuat perulangan sederhana, dengan tambahan kondisi, ketika label antara aktual dan prediksi sama, maka variabel *fold\_match\_counts* akan bertambah. Hasil tersebut disimpan didalam variabel bertipe *defaultdict()*.

```
def display_measures(acc, actual_counts,
                    predicted_counts,
                    match_counts, actual, predicted, idx_to_label, type, epoch,
                    fold, go, subtask):

    precisions = defaultdict(float)
    recalls     = defaultdict(float)
    f_measures = defaultdict(float)

    for label in actual_counts.keys():

        precision = match_counts[label] /
                    predicted_counts[label] if predicted_counts[label] > 0
                    else 0
        recall    = match_counts[label] /
                    actual_counts[label]   if actual_counts[label] > 0 else
                    0
        f_measure = 2 * precision * recall / (precision +
                    recall) if (precision + recall) > 0 else 0

        rc_temp2 += 100 * recall
        pr_temp2 += 100 * precision
        fm_temp2 += 100 * f_measure

        precisions[label] = 100 * precision
        recalls[label]    = 100 * recall
        f_measures[label] = 100 * f_measure

        if type == 'testing':

            track_recall.append([fold+1, epoch, label, 100 *
                                recall, 'recall'])
            track_fmeasure.append([fold+1, epoch, label, 100 *
                                   f_measure, 'f_measure'])
            track_precision.append([fold+1, epoch, label, 100
                                   * precision, 'precision'])
```

**Kode 5.37 Potongan Kode Method Display\_Measures()**

*Method display\_measures()* memiliki tujuan untuk menghitung informasi terkait hasil training, seperti pengukuran *recall*, *precision* dan *f-measure*. Rumus untuk menghitung nilai *precision* disimpan didalam variabel *precision*, sedangkan nilai *recall* disimpan didalam variabel *recall* dan *f-measure* disimpan didalam variabel *f\_measure*. Ketika mengakses *method display\_measures()* dengan data *testing*, maka hasil pengukuran tersebut disimpan didalam variabel *track\_recall* untuk *recall*, *track\_fmeasure* untuk *f-measure* dan *track\_precision* untuk *precision*. Hasil dari seluruh variabel *track* akan menyimpan pengukuran tersebut di setiap *epoch* dan akan ditulis dalam bentuk format *txt* untuk analisis hasil selanjutnya.

## BAB VI HASIL DAN PEMBAHASAN

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

### 6.1 Hasil Data Crawling

Proses *crawling* twitter dilakukan sejak 20 Desember 2017 hingga 1 Mei 2017. Hasil *tweet* yang terkumpul total adalah 62.873.706 dengan pembagian 62.834.464 adalah *tweet* yang tidak memiliki topik untuk proses pembuatan model *word embedding* dan 45.242 *tweet* terkait topik penelitian yaitu telekomunikasi untuk analisis sentiment. Hasil tersebut adalah hasil kotor dari proses *crawling*, sebelum diterapkannya *pre-processing*.

**Tabel 6.1 Hasil Crawling Tweet**

Kategori	Tweet
Analisis Sentimen	Makasiiiihih @triindonesia hadiah pulsanya sudah sampai dg sempurna. #3BangkitIndonesia
Analisis Sentimen	denger2 yg ada sinyal cmn XL doang? Bagus, besok live streaming non-stop mah \('`)/
Analisis Sentimen	@Telkomsel ini memang paling cacat logikanya. Beli paket 90rb 14GB 30 hari bonus tcash 10rb (masa aktif bonus cm beberapa hari). Bonusnya buat beli kuota 400MB 7 hari, tp gabisa kepake. Karena prioritasnya kuota yg 14GB. Mending gausah ngasih bonus tong.
Word Embedding	!!! Spoiler Alert ! Jadi selama 2 tahun itu Gintoki berkelana buat menyelidiki Ryuketsu karena ada kemungkinan kalau Utsuro bakal terlahir kembali , sampai akhirnya dia ketemu sama kakek tua di



Kategori	Tweet
	kuil yg menemukab seonggok daging di sungai yg kemudian berubah jadi bayi
Word Embedding	( c ) membalas , namun dirinya menahan diri karena yang berada di hadapannya adalah seorang wanita . Ia di ajarkan untuk tidak membalas jika seorang wanita bermain fisik dengannya . Huh , namun sepertinya Fionna bukan wanita seutuhnya . Mana ada seorang ( c )
Word Embedding	adamfawara1 Makanya , heran gue

## 6.2 Hasil Penghapusan Duplikasi *Dataset*

Hasil dari crawling twitter baik tweet dengan topik atau tanpa topik masih menyisakan duplikasi data, maka dari itu dengan menggunakan query sederhana, akan menghilangkan data duplikat. Berikut query nya

```
CREATE TABLE TWEET_NO_DISTINCT LIKE TWEET;

INSERT INTO TWEET_NO_DISTINCT VALUES
(SELECT * FROM TWEET GROUP BY TEXT);
```

Untuk menghilangkan *tweet* duplikat, langkah pertama dengan membuat tabel baru dengan kolom yang sama. Selanjutnya melakukan *select* dengan menggunakan fitur *group by*, hal ini akan menghilangkan secara otomatis text dan mengambil id *tweet* yang maksimal. Hasil seleksi tersebut akan dimasukkan kedalam tabel baru yang telah dibuat.

**Table 6.2 Hasil Tweet duplikat *word embedding***

Id_tweet	Tweet
987532534618570752	"Aku tau kok kalau bapak kamu itu adalah astonot..?? kok kamu tau sech..?? karena aku selalu melihat banyaknya bintang di mata kamu"

<b>Id_tweet</b>	<b>Tweet</b>
987517412793700353	"Aku tau kok kalau bapak kamu itu adalah astonot..?? kok kamu tau sech..?? karena aku selalu melihat banyaknya bintang di mata kamu"
987698513227087872	RT @_onew12: Akhirnya kita satu frame gengs, bapak jae kiyowo sekali @jaehwanna1_ @jinkidgu @Jinkiy1 <a href="https://t.co/VgJ3k8Q12b">https://t.co/VgJ3k8Q12b</a>
987536406942597121	RT @_onew12: Akhirnya kita satu frame gengs, bapak jae kiyowo sekali @jaehwanna1_ @jinkidgu @Jinkiy1 <a href="https://t.co/VgJ3k8Q12b">https://t.co/VgJ3k8Q12b</a>
987676588606025729	RT @_onew12: Akhirnya kita satu frame gengs, bapak jae kiyowo sekali @jaehwanna1_ @jinkidgu @Jinkiy1 <a href="https://t.co/VgJ3k8Q12b">https://t.co/VgJ3k8Q12b</a>

**Tabel 6.3 hasil tweet duplikat data topik**

<b>Topik</b>	<b>Tweet</b>
TRI	"@SpidolBekas: PakeTri mantep euy, tiap Jerman bikin 1 gol gw dapet bonus 150 mb @triindonesia"
TRI	"@SpidolBekas: PakeTri mantep euy, tiap Jerman bikin 1 gol gw dapet bonus 150 mb @triindonesia"

Setelah melakukan proses menghilangkan data duplikat, data yang dimiliki sekarang adalah 16.296.013 dimana 16,253,293 merupakan data tanpa topik dan 42.720 adalah data terkait topik. Jumlah pengurangan data yang cukup signifikan sebesar 46.583.693 diakibatkan oleh banyaknya akun yang melakukan *re-tweet* dari *tweet* yang telah di-post sebelumnya. Ketika sebuah akun melakukan *re-tweet* maka *crawler* akan tetap mengambil *re-tweet* tersebut karena terdeteksi sebagai *tweet*, dengan kalimat *tweet* sama persis ditambah kata "RT" diawal *tweet*.

### 6.3 Hasil Pemberian Label *Dataset*

Proses pemberian label data dilakukan oleh 3 orang, dilakukan setelah proses penghapusan duplikasi data. Berikut beberapa hasil dari pemberian label.

“ Ini kenapa di bale endah sinyal XL tiba2 bagus yah? Pindah kali yah tower nya ke dekat rumah.. \*aneh liat hp sendiri”

**Tabel 6.4 Label Tweet**

Label 1	Label 2	Label 3	Label Akhir
Sangat Positif	Positif	Sangat Positif	Sangat Positif

“ Membalas @rapdodge @triindonesia Saya juga cukup heran mengapa 4G 3 speednya bisa naik dirumah akan saya SS sisa paket saya, semoga aja di sekolah cepat membaik (6) ”

**Tabel 6.5 Label Tweet**

Label 1	Label 2	Label 3	Label Akhir
Positif	Positif	Positif	Positif

Untuk mendapatkan label akhir dilakukan perhitungan yang telah dijelaskan sebelumnya. Sehingga setelah dilakukan proses pemberian label, berikut hasil keseluruhan data.

**Tabel 6.6 Jumlah Distribusi Tweet berdasarkan label**

Label	Jumlah Tweet
Sangat Negatif	2325
Negatif	2,3,44
Netral	2384
Positif	2352
Sangat Positif	2310

Distribusi label untuk setiap topik dapat dilihat didalam tabel berikut.

**Tabel 6.7 Jumlah distribusi tweet berdasarkan topik**

Topik	Label	Jumlah
Telkomsel	Sangat Negatif	585

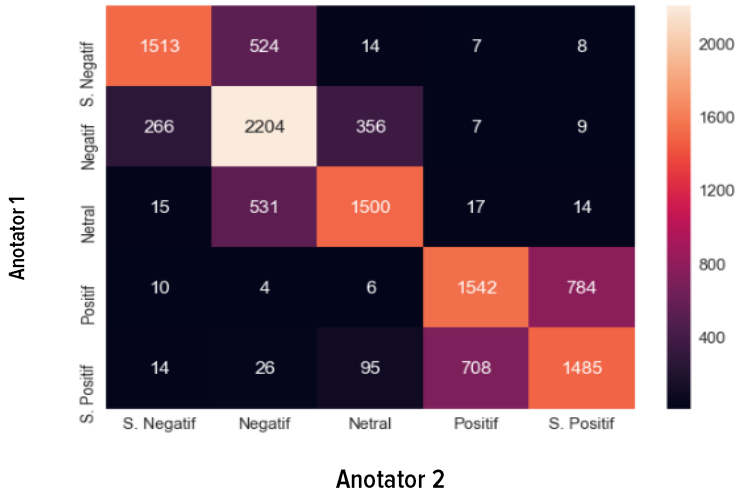
<b>Topik</b>	<b>Label</b>	<b>Jumlah</b>
Indosat	Sangat Negatif	568
Tri	Sangat Negatif	575
XI	Sangat Negatif	597
Telkomsel	Negatif	595
Indosat	Negatif	576
Tri	Negatif	588
XI	Negatif	585
Telkomsel	Netral	593
Indosat	Netral	577
Tri	Netral	589
XI	Netral	569
Telkomsel	Positif	597
Indosat	Positif	582
Tri	Positif	580
XI	Positif	593
Telkomsel	Sangat Positif	597
Indosat	Sangat Positif	562
Tri	Sangat Positif	580
XI	Sangat Positif	571

Berdasarkan hasil proses pemberian label, dapat dilihat bahwa dataset ini merupakan dataset dengan kategori balanced atau seimbang, karena selisih jumlah tweet antar label termasuk kecil. Jumlah pemberi label didalam penelitian ini adalah 3 orang. Berikut jumlah *tweet* dengan masing masing label untuk setiap anotator label.

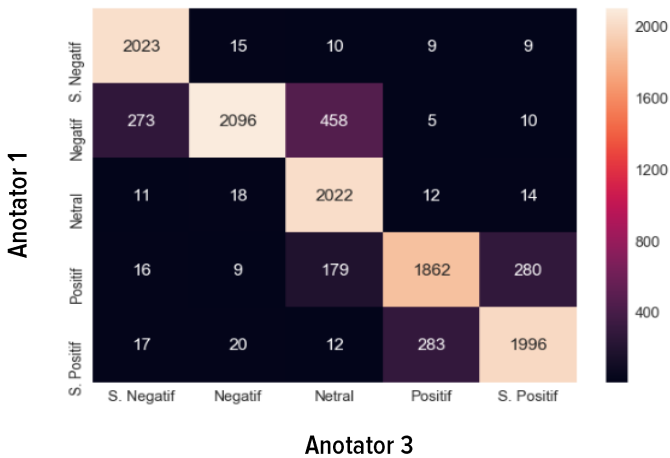
**Tabel 6.8 Distribusi Label per Anotator**

Label	Anotator 1	Anotator 2	Anotator 3
Sangat Negatif	2066	1818	2340
Negatif	2842	3289	2158
Netral	2077	1971	2681
Positif	2346	2281	2171
Sangat Positif	2328	2300	2309

Berdasarkan tabel 6.8 berikut *confusion matrix* berdasarkan anotator 1 dan anotator 2.

**Gambar 6.1 Confusion Matrix Anotator 1 dan 2**

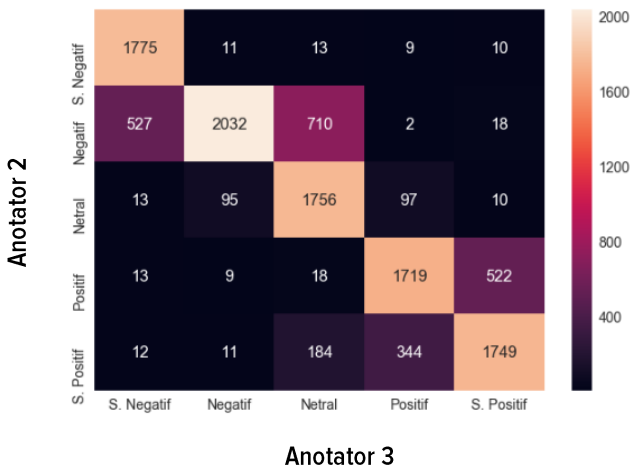
Perbandingan untuk anotator 1 dan anotator 3 dapat dilihat di *confusion matrix* dibawah ini.



Perbandingan untuk anotator 2 dan anotator 3 dapat dilihat dari

**Gambar 6.2** *Confusion Matrix* Anotator 1 dan 3

*confusion matrix* dibawah ini



**Gambar 6.3** *Confusion Matrix* Anotator 2 dan 3

Melihat tingkat kesepakatan antar anotator dapat menggunakan pengukuran *cohens kappa*. *Cohens kappa* dapat mengukur tingkat kesepakatan diantara dua anotator untuk melakukan pemberian kelas terhadap sebuah *tweet*. Nilai yang dihasilkan *cohens kappa* berkisar antara 0 – 1 dengan aturan semakin mendekati 1 maka semakin baik. Berikut tabel tingkat kesepakatan *cohens kappa*.

**Tabel 6.9 Tingkat Kesepakatan *Cohens Kappa***

<b>Indeks</b>	<b>Tingkat Kesepakatan</b>
< 0.20	Rendah
0.21 – 0.40	Sedang
0.41 – 0.60	Cukup
0.61 – 0.80	Kuat
0.81 – 1.00	Sangat Kuat

Dalam penelitian ini untuk menghitung nilai *cohens kappa* menggunakan *library sklearn* lebih tepatnya menggunakan method *metrics.cohen\_kappa\_score()*. Nilai *cohens kappa* yang dihasilkan didalam penelitian ini sebagai berikut.

**Tabel 6.10 Hasil *Cohens Kappa***

<b>Anotator</b>	<b>Indeks</b>
1 dan 2	0.631
2 dan 3	0.718
1 dan 3	0.822

Berdasarkan tabel 6.10 nilai *cohens kappa* yang dihasilkan berbeda untuk masing masing anotator. Hal ini wajar karena perspektif setiap orang berbeda beda. Hasil *cohens kappa* antara anotator 1 dan 2 termasuk kategori kuat dengan nilai 0.631, sedangkan anotator 2 dan 3 memiliki kategori yang sama dengan nilai 0.718 yaitu kategori kuat. Sedangkan anotator 1 dan 3 memiliki nilai terbaik yaitu 0.822 termasuk kategori sangat kuat. Sehingga rata rata dari ketiga anotator tersebut adalah 0.723 dan termasuk kategori kuat.

#### 6.4 Hasil Filtering Bahasa Indonesia

*Dataset* untuk proses word embedding selanjutnya akan diseleksi untuk menghilangkan tweet yang bukan berbahasa Indonesia. Berikut beberapa hasil proses seleksi bahasa indonesia

Tabel 6.11 Hasil seleksi bahasa Indonesia

<b>Tweet</b>	<b>Bahasa</b>	<b>Index</b>
@1106_leda Selamat siang. Saat ini perjalanan KA dari Yogyakarta-Surabaya sudah kembali normal. Trims.	Indonesia	0.99999654
@1194luna teka nakakapanibago sagot mo parang may something pm kita wait	Tagalog	0.999972589
@11_jisol Saya maunya top 1 di hati kamu juga. Ga cuma di rank mention	Indonesia	0.9999964139
@1,2,30percent bisa chat line kita kak	Indonesia	0.7142854778
@1,2,3OCGV Boleh, jadi siapa dek?	Indonesia	0.9999973731
@127OMile jungyeon yiyang ningning? mereka srg juga kan?	Indonesia	0.8571390943
@13sisu @SchloTo @Liebelovepeace Einmaleins abfragen, bei Freiarbeit mithelfen, Sich was laut Vorlesen lassen usw.	Belanda	0.9999987332

Dari hasil seleksi bahasa indonesia, *library langdetect* cukup bisa menentukan bahasa dari sebuah *tweet* yang diberikan. Namun hasil deteksi bahasa yang diberikan semakin kurang akurat apabila kalimat tersebut memiliki jumlah kata yang



sedikit. Hasil dari seleksi bahasa indonesia menyisakan data untuk pembuatan model word embedding yaitu 10.254.945 *tweet*.

## 6.5 Hasil Pembersihan Dataset

Sebelum dataset akan digunakan baik untuk pembuatan model word embedding ataupun sentimen analisis akan dilakukan proses pembersihan disetiap tweet dengan tujuan membuat dataset lebih teratur. Berikut proses pemberihan dataset.

**Tabel 6.12 Proese Pembersihan Tweet**

<b>Aktivitas / Kondisi</b>	<b>Hasil</b>
Tweet Awal	indosat&nbsp;MANTAP djaja koneksinya , ngebut euy koyok jaran (goyang), cek sini coba http://indosat.co.id @indosatCare mantap abisssss..... :))))
Mengganti URL	indosat&nbsp;MANTAP djaja koneksinya , ngebut euy koyok jaran (goyang), cek sini coba <url> @indosatCare mantap abisssss..... :))))
Menghilangkan Atribut HTML	indosat MANTAP djaja koneksinya , ngebut euy koyok jaran (goyang), cek sini coba <url> @indosatCare mantap abisssss..... :))))
Menghilangkan Mention Akun	indosat MANTAP djaja koneksinya , ngebut euy koyok jaran (goyang), cek sini coba <url> <mention> mantap abisssss..... :))))
Menghilangkan Karakter Berulang	indosat MANTAP djaja koneksinya , ngebut euy koyok jaran (goyang), cek sini coba

Aktivitas / Kondisi	Hasil
	<url> <mention> mantap abiss.. :))
Menghilangkan <i>Elipsis</i>	indosat MANTAP djaja koneksinya , ngebut euy koyok jaran (goyang), cek sini coba <url> <mention> mantap abiss <elipsis> :))
Translasi <i>Emoticon</i>	indosat MANTAP djaja koneksinya , ngebut euy koyok jaran (goyang), cek sini coba <url> <mention> mantap abiss <elipsis> <senyum_senyum>
Menghilangkan Simbol	indosat MANTAP djaja koneksinya ngebut euy koyok jaran goyang cek sini coba url mention mantap abiss elipsis senyum senyum
Merubah Menjadi Huruf Kecil	indosat mantap djaja koneksinya ngebut euy koyok jaran goyang cek sini coba url mention mantap abiss elipsis senyum senyum
Tokenisasi	['indosat', 'mantap', 'djaja', 'koneksinya', 'ngebut', 'euy', 'koyok', 'jaran', 'goyang', 'cek', 'sini', 'coba', 'url', 'mention', 'mantap', 'abiss', 'elipsis', 'senyum', 'senyum']

Setelah proses tokenisasi, dataset siap untuk digunakan baik untuk proses *word embedding* ataupun analisis sentimen

## 6.6 Hasil Pembuatan Model Word Embedding

Model word embedding yang dihasilkan memiliki jumlah vocabulary sebanyak 495884 untuk model word2vec sedangkan vocabulary untuk model fasttext adalah 300686 . Jumlah model

yang dihasilkan adalah 2 buah model dengan paramter sebagai berikut.

**Tabel 6.13 Hasil Parameter Word Embedding**

<b>Paramter</b>	<b>Model Skipgram</b>	<b>Model CBOW</b>
Size	300	300
Sg	1	0
Iter	2	2
Window	5	5
seed	1	1
min_count	5	5
Alpha	0.025	0.025

## 6.7 Hasil Pembuatan Model CNN

Skenario pembuatan model dalam penelitian ini terdapat 3 jenis seperti telah dijelaskan di bab 4.6.4 sebelumnya.

### 6.7.1 Konfigurasi Parameter Awal

Sebelum melakukan skenario pertama, harus terlebih dahulu ditentukan konfigurasi model dan proses training awal agar semua skenario dapat dibandingkan hasilnya dengan jelas. *Hyperparameter* akan ditentukan berdasarkan penelitian sebelumnya, berikut tabel terkait konfigurasi awal.

**Tabel 6.14 Konfigurasi Awal Model CNN**

Parameter	Nilai
Folding Data	10
Epoch	10
SGD	Adadelta
Mini Batch	50
Word Embedding Model	Word2Vec Skip-gram
Feature Maps	100
Activation Function	ReLU
Pooling	1-Max Pooling
Dropout Rate	0.5

L2 Norm Constrain	3
-------------------	---

## 6.7.2 Subtask C

Percobaan pertama dilakukan untuk subtask C dimana memiliki 5 buah kelas.

### 6.7.2.1 Model *Embedding Static*

#### 6.7.2.1.1 Pengaruh *Filter Region Size*

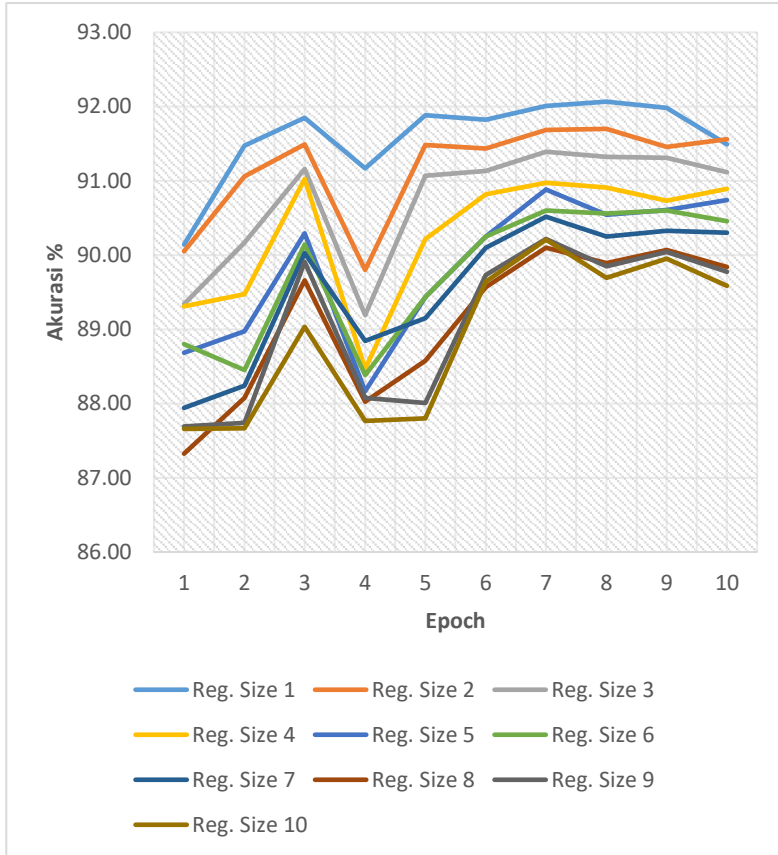
Skenario satu adalah menguji efek perubahan nilai *region size* terhadap akurasi model. Nilai *region size* yang digunakan adalah 1,2,3,4,5,6,7,8,9 dan 10. Output dari skenario ini adalah nilai *region size* yang terbaik dan dapat dikembangkan menjadi *multi-region size*. Pengukuran utama yang digunakan dalam skenario ini adalah akurasi karena persebaran data yang cukup seimbang dan dibantu dengan pengukuran lainnya yaitu *recall*, *precision* dan *F-Measure*. Berikut hasil dari skenario pertama untuk subtask C dengan varian *static*.

**Tabel 6.15 Hasil Akurasi Model**

Region Size	Accuracy	F-Measure	Recall	Precision
<b>1</b>	88.11	85.08	85.39	85.05
<b>2</b>	87.70	85.03	84.98	85.27
<b>3</b>	87.47	84.23	83.89	84.69
<b>4</b>	87.13	84.70	85.05	84.54
<b>5</b>	86.52	83.77	84.07	83.62
<b>6</b>	86.46	83.49	83.12	84.07
<b>7</b>	86.44	83.73	83.18	84.49
<b>8</b>	85.93	83.31	82.43	84.37
<b>9</b>	85.51	82.62	81.34	84.30
<b>10</b>	85.78	83.10	81.51	84.98

Dari hasil percobaan tersebut, *filter region size* dengan ukuran 1 memiliki nilai akurasi tertinggi, begitu juga dengan hasil berdasarkan nilai *F-Measure* dan *Recall*. Hasil dari *precision*

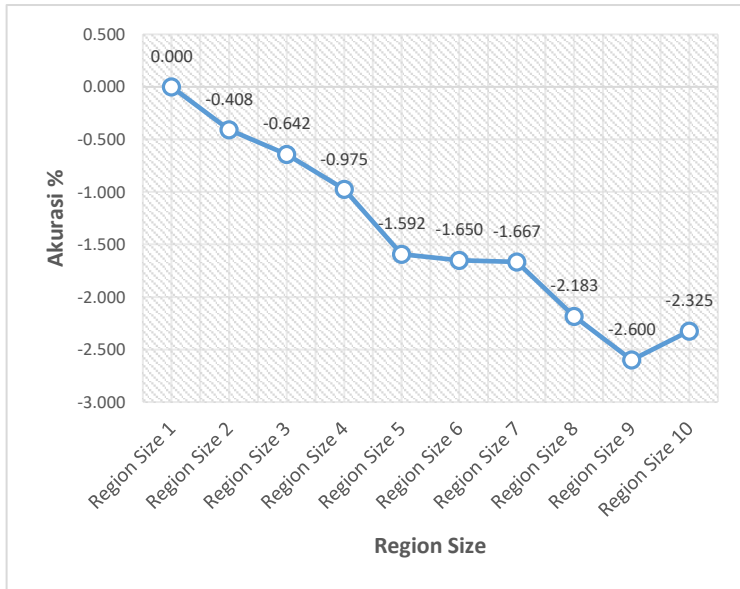
sedikit berbeda yaitu menunjukkan *region size* 2 dengan nilai tertinggi, namun hasil dari percobaan ini tetap mengacu terhadap pengukuran utama.



**Gambar 6.4 Akurasi Training Model**

Berdasarkan grafik akurasi disetiap *epoch training* diatas, ukuran *region size* 1 selalu berada di akurasi tertinggi. Hal ini menunjukkan bahwa jumlah kata yang dibutuhkan untuk menentukan sentimen sebuah *tweet* adalah satu kata. Berdasarkan percobaan pertama ini menunjukkan bahwa semakin besar filter *region size* membuat akurasi semakin

menurun. Hal ini berarti membuat kombinasi untuk *multi region size* yaitu, 1,1,1 dan 1,2,3 untuk *feature maps* 100 dan 200.



**Gambar 6.5 Grafik Perubahan Akurasi**

Berdasarkan gambar 6.5 tren dari nilai akurasi semakin menurun seiring bertambah besar ukuran *filter region size*

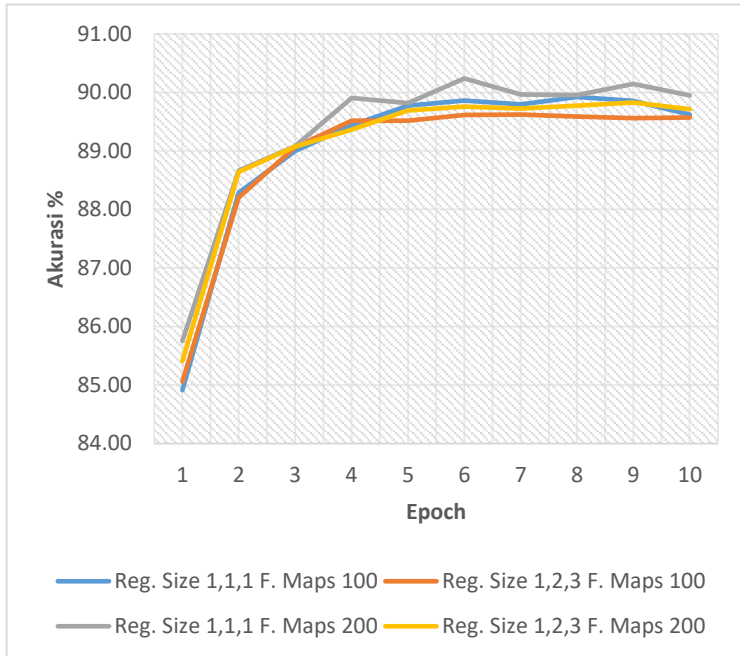
#### **6.7.2.1.2 Pengaruh Multi Region Size dan Feature Maps**

Skenario dua adalah menguji efek perubahan nilai *multi-region size* dan perubahan nilai *feature maps* terhadap akurasi model. Nilai *multi-region size* didapatkan berdasarkan hasil skenario sebelumnya. Pengukuran utama yang digunakan dalam skenario ini adalah akurasi karena persebaran data yang cukup seimbang dan dibantu dengan pengukuran lainnya yaitu *recall*, *precision* dan *F-Measure*. Berikut hasil dari skenario kedua untuk subtask C dengan varian *static*.

**Tabel 6.16 Akurasi Training Model**

Region Size	Feature Maps	Accuracy	F-Measure	Recall	Precision	MAE
<b>1,1,1</b>	<b>100</b>	89.62	88.48	87.54	89.58	0.2197
<b>1,2,3</b>	<b>100</b>	89.57	88.23	87.38	89.23	0.2190
<b>1,1,1</b>	<b>200</b>	89.95	89.03	87.41	90.88	0.2118
<b>1,2,3</b>	<b>200</b>	89.72	88.44	87.34	89.65	0.2173

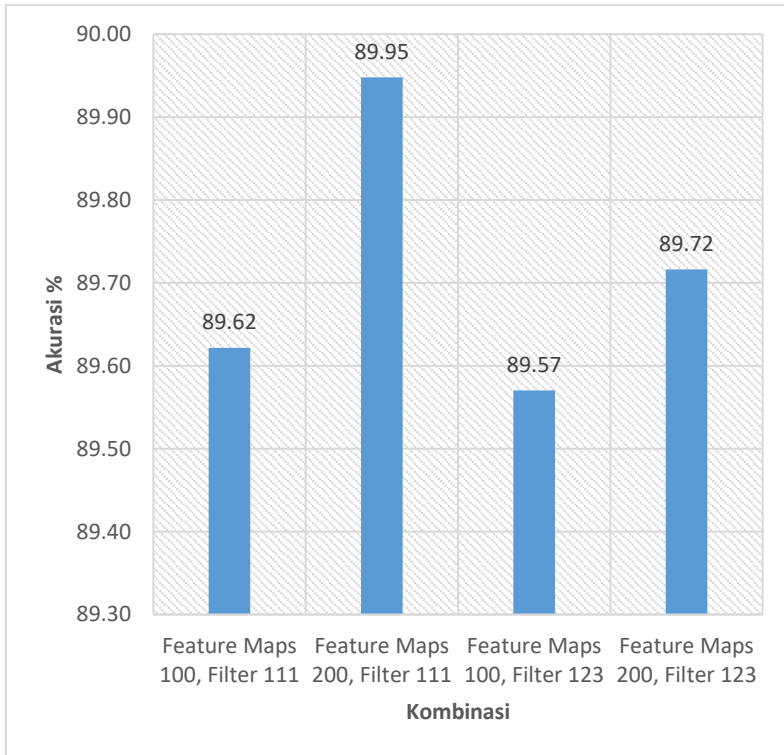
Dari tabel 6.16 berdasarkan pengukuran akurasi, kombinasi antara *feature maps* 200 dan *region size* 1,1,1 memiliki nilai akurasi, *F-Measure* dan *precision* tertinggi, namun nilai *recall* yang rendah. Nilai *region* tertinggi adalah kombinasi *feature maps* 100 dan *filter size* 1,1,1. Parameter terbaik yang dihasilkan dari skenario 2 tetap dari akurasi tertinggi yaitu kombinasi antara *feature maps* 200 dan *region size* 1,1,1 meskipun pada pengukuran *recall* tidak menduduki urutan pertama. Selisih nilai *recall* dari kombinasi antara *feature maps* 200 dan *region size* 1,1,1 dengan kombinasi antara *feature maps* 100 dan *region size* 1,1,1 relatif kecil yaitu 0.13 %.



**Gambar 6.6 Akurasi Training Model**

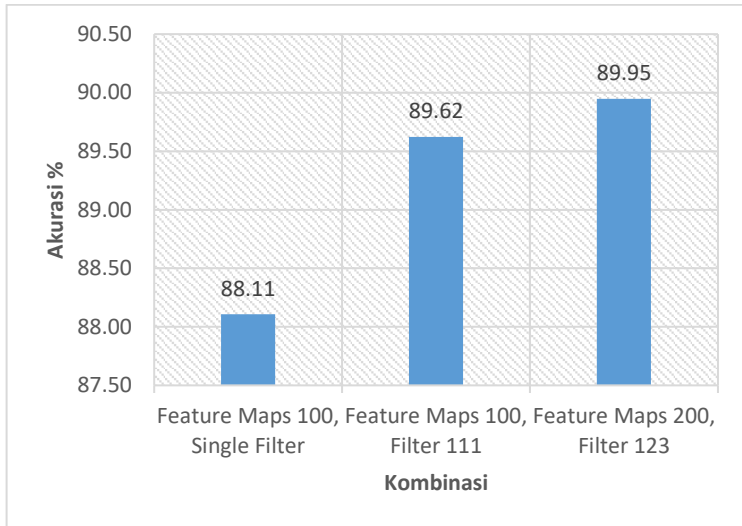
Berdasarkan gambar 6.6 proses *training* untuk *multi region size* memiliki stabilitas akurasi yang lebih bagus daripada *single region size*. kombinasi antara *feature maps* 200 dan *region size* 1,1,1 selama proses *training* sebagian besar menduduki posisi teratas dibandingkan kombinasi lainnya. Nilai akurasi dari *multi region size* terbukti lebih tinggi dibandingkan *single region size* dengan selisih akurasi 1.84 %. Hal ini membuktikan pula bahwa dengan jumlah *filter* akan meningkatkan akurasi model.





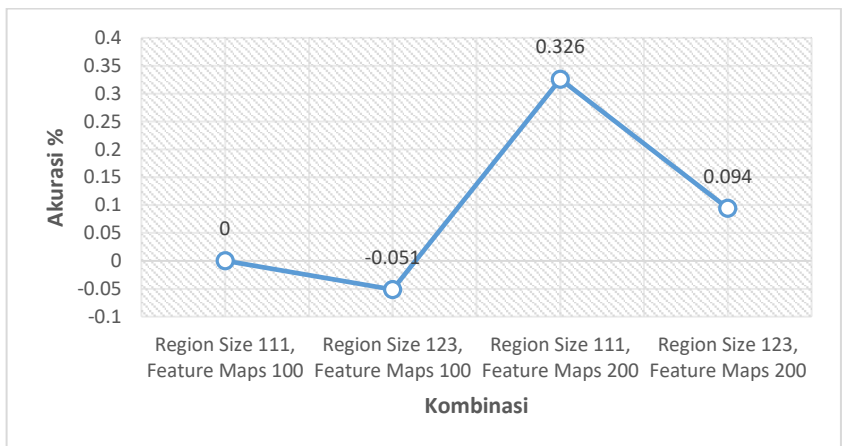
**Gambar 6.7 Perbandingan Akurasi**

Berdasarkan perbedaan dari nilai *feature maps*, nilai *feature maps* lebih tinggi memiliki nilai akurasi lebih tinggi. Dalam penelitian ini *feature maps* dibatasi hanya 100 dengan 200 karena terdapat batasan dari *hardware*. Peningkatan akurasi hingga 0.33 % saat menggunakan *feature maps* 200 untuk kombinasi *filter size* 1,1,1, sedangkan untuk kombinasi *filter size* 1,2,3 meningkat sebesar 0.15 %.



**Gambar 6.8 Perbandingan Akurasi**

Berdasarkan jumlah *filter region size* yang digunakan *multi filter region size* memiliki performa yang lebih baik. Akurasi dapat meningkat hingga 1.84 % jika menggunakan *multi filter region size*.



**Gambar 6.9 Grafik Perubahan Akurasi**

Berdasarkan grafik perubahan akurasi diatas, menunjukkan bahwa *feature maps* 200 memiliki peningkatan data lebih besar.

### 6.7.2.2 Model *Embedding Non-Static*

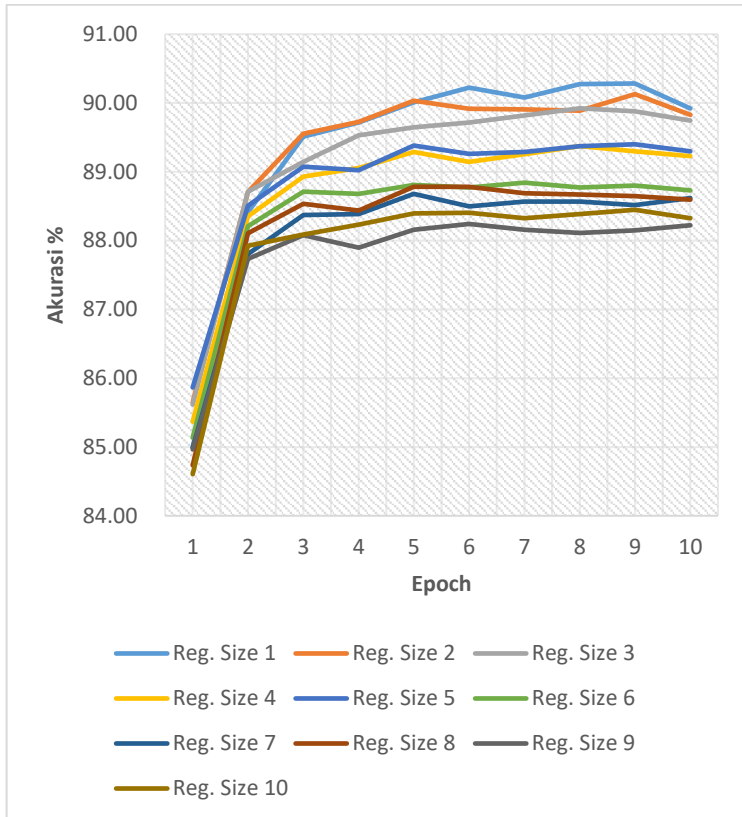
#### 6.7.2.2.1 Pengaruh *Filter Region Size*

Skenario satu adalah menguji efek perubahan nilai *region size* terhadap akurasi model. Nilai *region size* yang digunakan adalah 1,2,3,4,5,6,7,8,9 dan 10. Output dari skenario ini adalah nilai *region size* yang terbaik dan dapat dikembangkan menjadi *multi-region size*. Pengukuran utama yang digunakan dalam skenario ini adalah akurasi karena persebaran data yang cukup seimbang dan dibantu dengan pengukuran lainnya yaitu *recall*, *precision* dan *F-Measure*. Berikut hasil dari skenario pertama untuk subtask C dengan varian *non-static*.

**Tabel 6.17 Hasil Akurasi Model**

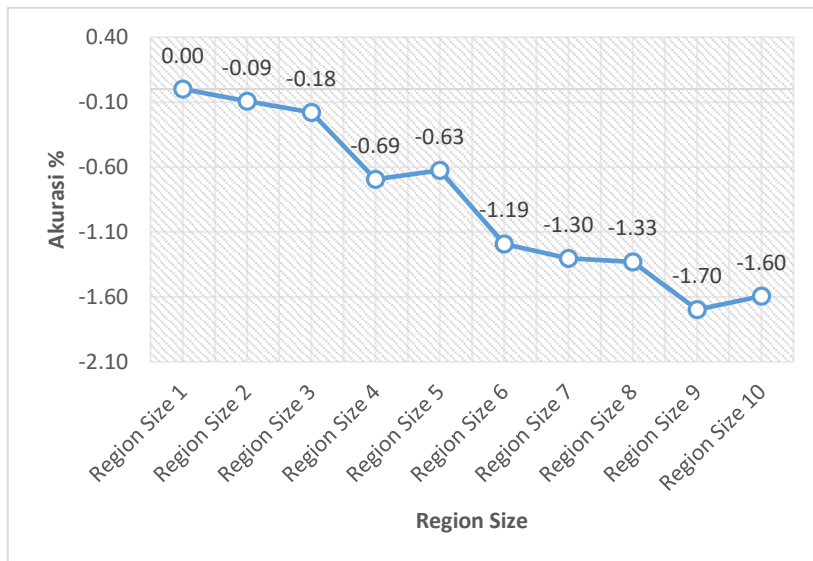
Region Size	Accuracy	F-Measure	Recall	Precision
<b>1</b>	89.92	98.55	98.54	98.57
<b>2</b>	89.83	98.55	98.49	98.61
<b>3</b>	89.74	98.49	98.41	98.57
<b>4</b>	89.23	98.36	98.41	98.32
<b>5</b>	89.30	98.43	98.46	98.40
<b>6</b>	88.73	98.26	98.25	98.27
<b>7</b>	88.62	98.24	98.21	98.27
<b>8</b>	88.59	98.25	98.30	98.22
<b>9</b>	88.22	98.06	98.16	97.97
<b>10</b>	88.33	98.27	98.28	98.27

Berdasarkan tabel 6.17, nilai *region size* 1 memiliki tingkat akurasi, *F-measure* dan *recall* tertinggi. Hasil *F-Measure* dari *region size* 1 dan 2 memiliki nilai yang sama dan hasil dari *precision* sedikit berbeda yaitu menunjukkan *region size* 2 dengan nilai tertinggi, namun hasil dari percobaan ini tetap mengacu terhadap pengukuran utama.



**Gambar 6.10 Akurasi Training Model**

Berdasarkan gambar 6.10 akurasi 6.10, region size 1 dan 2 secara bergantian menduduki posisi tertinggi terutama setelah *epoch* 3. Hal ini sesuai dengan hasil di *epoch* terakhir dimana *region size* 1 dan 2 memiliki perbedaan hasil yang tipis. Berdasarkan percobaan pertama ini menunjukkan bahwa semakin besar filter region size membuat akurasi semakin menurun. Hal ini berarti membuat kombinasi untuk *multi region size* yaitu, 1,1,1 dan 1,2,3 untuk *feature maps* 100 dan 200.



**Gambar 6.11 Grafik Perubahan Akurasi**

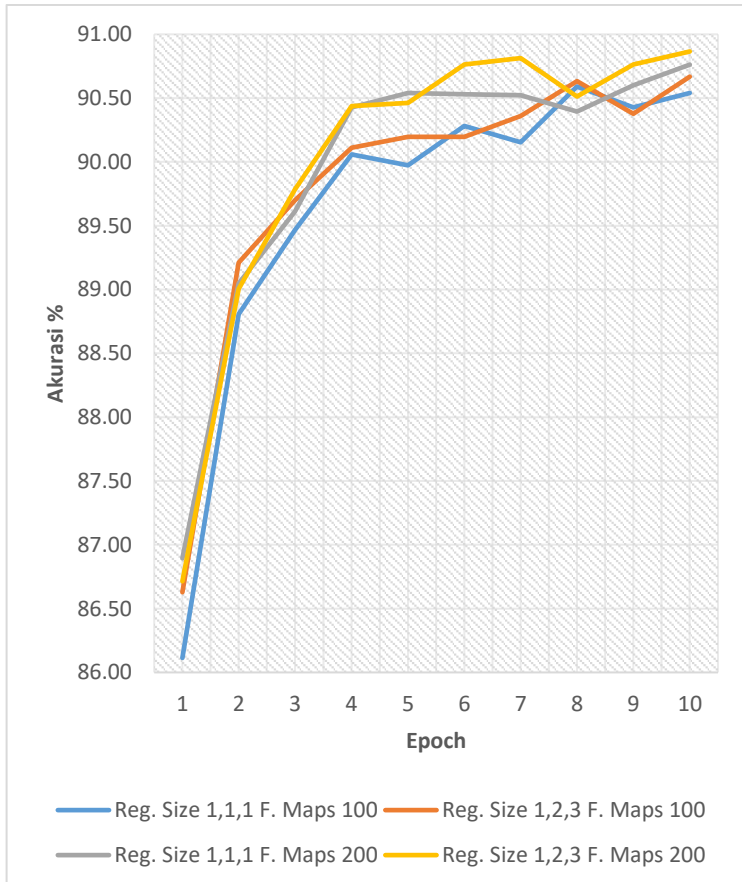
### 6.7.2.2.2 Pengaruh *Multi Region Size* dan *Feature Maps*

Skenario dua adalah menguji efek perubahan nilai *multi-region size* dan perubahan nilai *feature maps* terhadap akurasi model. Nilai *multi-region size* didapatkan berdasarkan hasil skenario sebelumnya. Pengukuran utama yang digunakan dalam skenario ini adalah akurasi karena persebaran data yang cukup seimbang dan dibantu dengan pengukuran lainnya yaitu *recall*, *precision* dan *F-Measure*. Berikut hasil dari skenario kedua untuk subtask C dengan varian *non-static*.

**Tabel 6.18 Hasil Akurasi Model**

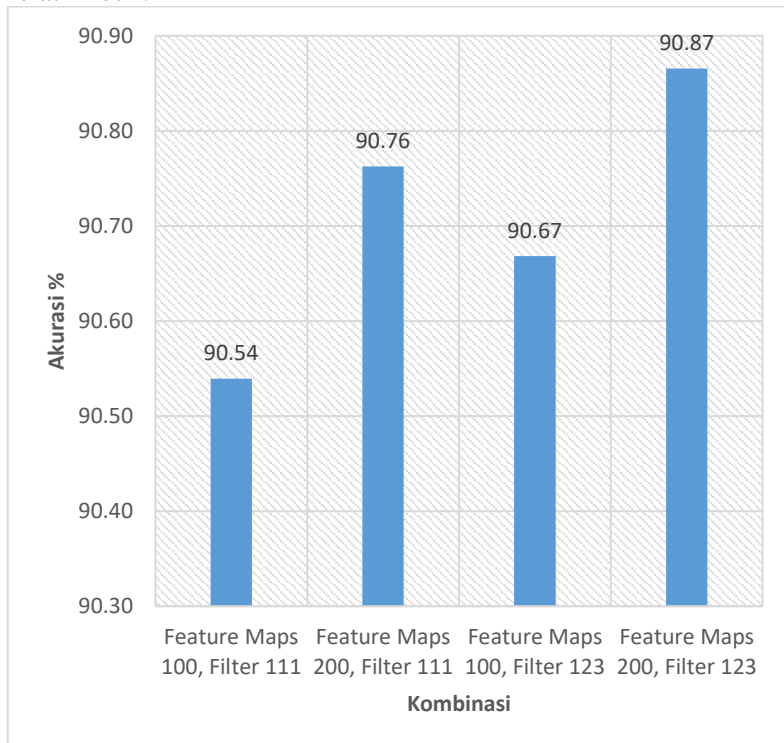
Region Size	Feature Maps	Accuracy	F-Measure	Recall	Precision	MAE
1,1,1	100	90.54	89.21	90.61	87.96	0.1979
1,2,3	100	90.67	88.66	90.06	87.36	0.1956
1,1,1	200	90.76	88.03	88.59	87.56	0.1946
1,2,3	200	90.87	87.88	88.24	87.64	0.1948

Dari hasil percobaan diatas kombinasi *region size* 1,2,3 dengan *feature maps* 200 memiliki nilai akurasi paling tinggi, namun berdasarkan perhitungan lain kombinasi tersebut memiliki posisi lebih rendah dari kombinasi *region size* 1,1,1 dengan *feature maps* 100, baik dari *recall*, *precision* dan *F-Measure*. Hal ini tidak mempengaruhi hasil dari skenario ini, karena mengikuti pengukuran utama yaitu akurasi.



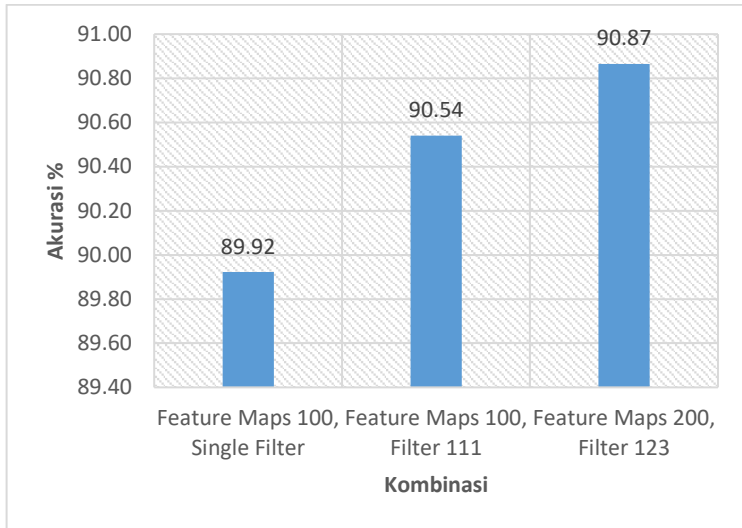
**Gambar 6.12 Akurasi Training Model**

Dari gambar 6.12 diatas kombinasi *region size* 1,2,3 dengan *feature maps* 200 menduduki posisi teratas sejak epoch ke 6 dibandingkan kombinasi lainnya. Untuk *multi filter region size* model *embedding non static* selisih akurasi disetiap epochnya relatif kecil.



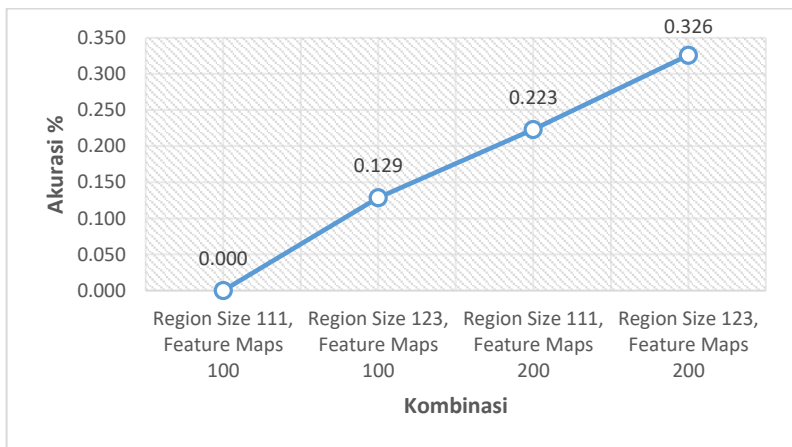
**Gambar 6.13 Perbandingan Akurasi Model**

Berdasarkan perbedaan dari nilai *feature maps*, nilai *feature maps* lebih tinggi memiliki nilai akurasi lebih tinggi. Hal tersebut terjadi baik untuk kombinasi *multi filter region size* 1,1,1 dan 1,2,3. Dalam penelitian ini *feature maps* dibatasi hanya 100 dengan 200 karena terdapat batasan dari *hardware*.



**Gambar 6.14 Perbandingan Akurasi Model**

Berdasarkan jumlah *filter region size* yang digunakan *multi filter region size* memiliki performa yang lebih baik. Akurasi dapat meningkat hingga 0.94 % jika menggunakan *multi filter region size*.



**Gambar 6.15 Grafik Perubahan Akurasi**



Berdasarkan gambar 6.15 bahwa trend akurasi meningkat dengan *feature maps* yang lebih tinggi

### 6.7.2.3 Pengaruh Model Embedding

Percobaan selanjutnya setelah menguji variasi *single filter region size*, *multiple region size* hingga *feature maps* adalah menguji pengaruh model *word embedding* yang diterapkan untuk proses merubah kata menjadi vektor kata. Sebelum membandingkan pengaruh model *word embedding*, ditentukan terlebih dahulu antara model *embedding static* dengan *non static* yang memiliki akurasi terbaik, berikut hasilnya.

**Tabel 6.19 Hasil Akurasi Model**

Embedding	Accuracy	F-Measure	Recall	Precision
<b>Static</b>	89.95	89.03	87.41	90.88
<b>Non Static</b>	90.87	87.88	88.24	87.64

Dari hasil tersebut dapat dilihat model dengan *embedding non static* memiliki akurasi yang lebih tinggi dibandingkan dengan varian *static*. Namun memiliki perbedaan dipengukuran *F-Measure* sekitar 1.15 %. Hal tersebut dipengaruhi oleh nilai *precision* yang lebih rendah 3.24 %. Untuk melakukan analisa lebih lanjut, berikut *confusion matrix* untuk kedua model diatas.

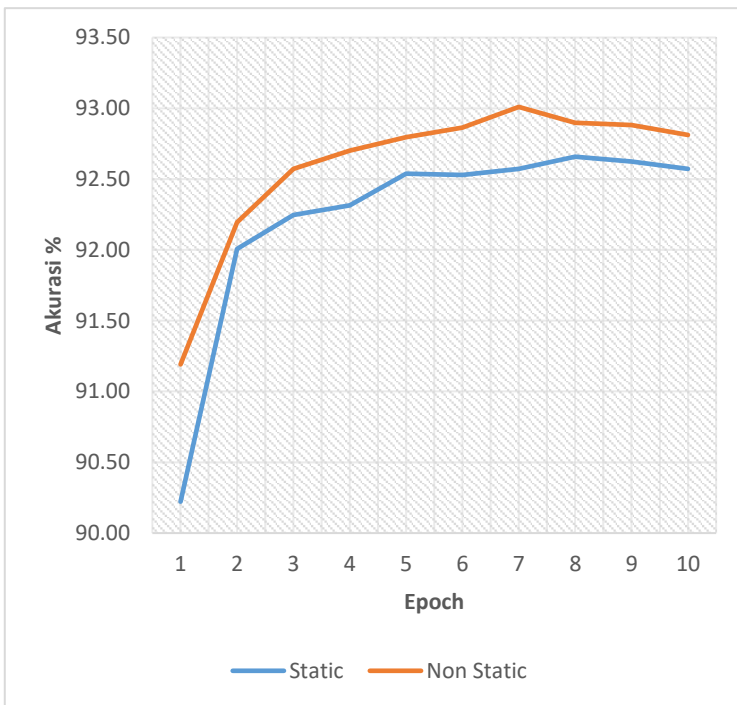
		ACTUAL				
		S. Negatif	Negatif	Netral	Positif	S. Positif
PREDICTED	S. Negatif	2176	70	14	28	66
	Negatif	60	1855	31	62	100
	Netral	19	42	2113	85	111
	Positif	32	105	60	2049	125
	S. Positif	33	49	42	73	2259

Gambar 6.16 *Confusion Matrix Static*

		ACTUAL				
		S. Negatif	Negatif	Netral	Positif	S. Positif
PREDICTED	S. Negatif	2175	83	17	30	49
	Negatif	65	1862	31	66	84
	Netral	18	46	2144	79	83
	Positif	33	123	62	2053	100
	S. Positif	36	58	62	76	2224

Gambar 6.17 *Confusion Matrix Non Static*

Dari tabel *confusion matrix* diatas menunjukkan bahwa rendahnya nilai *F-Measure* didalam model *embedding non-static* dibanding dengan model *embedding static* dipengaruhi oleh rata rata presisi yang lebih rendah. *Confusion matrix* didapatkan dari hasil penjumlahan nilai *confusion matrix* disemua *fold* di *epoch* 10. Nilai rata rata presisi yang rendah ini dikarenakan nilai presisi label sangat positif cukup rendah yaitu 84.89 %, label yang paling sering salah diprediksi sangat positif adalah label positif, sejumlah 125 *tweet*. hal ini cukup wajar karena perbedaan kedua label tersebut cukup sedik it, sehingga fitur yang terekam didalam sistem klasifikasi masih cukup bias. Namun apabila dilihat berdasarkan pengukuran utama model *embedding non – static* tetap unggul didalam akurasi, sehingga model *embedding non – static* tetap menjadi yang terbaik.



**Gambar 6.18 Perbandingan Akurasi Model**

Berdasarkan gambar 6.18, model *embedding non-static* memiliki tingkat stabilitas akurasi lebih tinggi disetiap *epoch* nya. Hal tersebut dilihat dari standard deviasi yang lebih rendah yaitu 1.227842 dari *epoch* 1 hingga 10, sedangkan untuk model *embedding static* adalah 1.285879. Perbedaan ini dipengaruhi karena *mode non-static* menggunakan *adadelta optimizer* untuk memperbarui berat vektor kata. Sehingga proses *training* model *embedding non-static* lebih efisien.

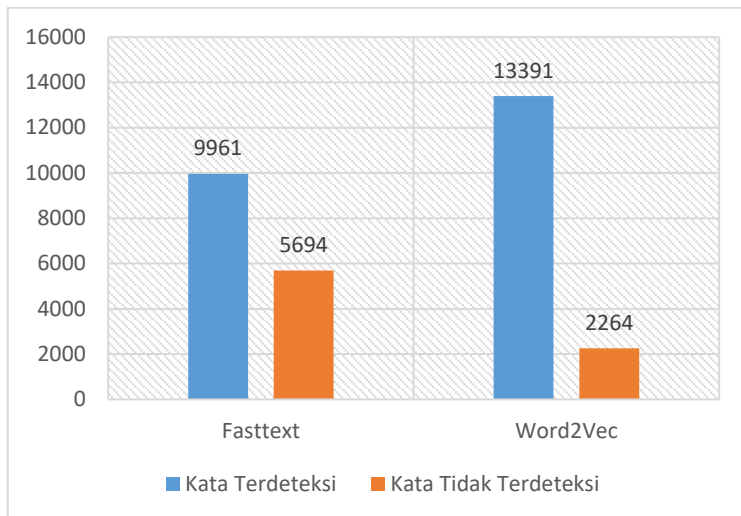
#### 6.7.2.4 Pengaruh Model Word Embedding

Pengaruh model word embedding terhadap akurasi cukup signifikan, terutama saat menggunakan learning alorithm yang berbeda.

**Tabel 6.20 Hasil Akurasi Model**

Word Embedding	Accuracy	F-Measure	Recall	Precision
<b>Fasttext</b>	89.27	86.74	89.29	89.86
<b>w2v Skipgram</b>	90.87	87.88	88.24	87.64
<b>w2v CBOW</b>	87.81	85.12	84.47	85.87

Berdasarkan hasil akurasi tersebut, model *word embedding* tersebut *word2vec* dengan *learning algorithm skipgram* memiliki vektor kata yang lebih sesuai untuk *training* data model CNN. Selisih akurasi dari perbedaan *learning* cukup tinggi yaitu 3.06 % sehingga model *skipgram* merupakan model *word embedding* terbaik. Pengaruh penggunaan algoritma tidak terlalu berubah secara signifikan, namun menggunakan fasttext memiliki nilai akurasi yang lebih rendah.



**Gambar 6.19 Perbandingan Jumlah Kata**

Gambar 6.19 menjelaskan jumlah kata berada didalam corpus word embedding untuk algoritma fasttext lebih sedikit dengan model word embedding algoritma word2vec. Dengan tingginya jumlah kata yang tidak terdeteksi, maka nilai vektor saat pertama kali di inisiasi lebih banyak yang memiliki nilai random.

### 6.7.3 Subtask B

Percobaan pertama dilakukan untuk *subtask* B dimana memiliki 3 buah kelas.

#### 6.7.3.1 Model *Embedding Static*

##### 6.7.3.1.1 Pengaruh *Filter Region Size*

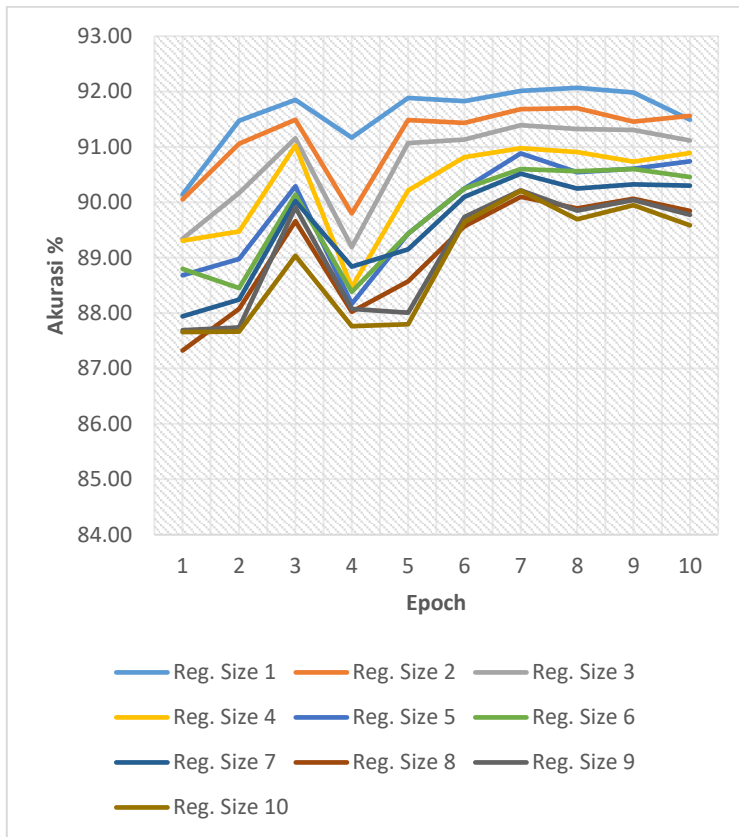
Skenario satu adalah menguji efek perubahan nilai *region size* terhadap akurasi model. Nilai *region size* yang digunakan adalah 1,2,3,4,5,6,7,8,9,10 sama seperti percobaan sebelumnya untuk *subtask* C. Pengukuran utama yang digunakan sama seperti *subtask* sebelumnya yaitu akurasi dengan pertimbangan

pengukuran lainnya seperti, *F-Measure*, *Recall* dan *Precision*. Berikut hasil pengaruh *filter region size* untuk model *embedding static*.

**Tabel 6.21 Hasil Training Model**

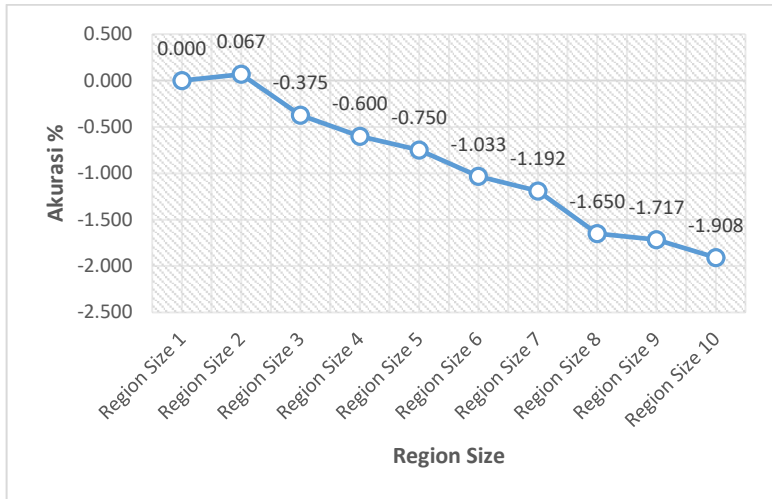
Region Size	Accuracy	F-Measure	Recall	Precision
<b>1</b>	91.49	85.26	79.12	92.63
<b>2</b>	91.56	85.53	81.04	90.73
<b>3</b>	91.12	84.56	79.77	90.26
<b>4</b>	90.89	84.55	79.53	90.37
<b>5</b>	90.74	84.28	79.13	90.28
<b>6</b>	90.46	83.65	78.13	90.15
<b>7</b>	90.30	83.44	77.60	90.35
<b>8</b>	89.84	82.60	76.57	89.77
<b>9</b>	89.78	82.44	76.54	89.52
<b>10</b>	89.58	82.09	76.21	89.11

Hasil dari pengaruh *single filter region size*, akurasi terbaik adalah ukuran *region size 2*. Hasil dari pengukuran *F-Measure* dan *Recall* juga menyatakan hal yang sama. Nilai *recall* yang dimiliki *region size 2* adalah satu satunya yang melebihi nilai 80 % bahkan 81 %. Namun pada pengukuran *precision filter region size* terbaik adalah ukuran 1 dengan hasil *precision* 92.63 %.



**Gambar 6.20 Akurasi Training Model**

Terdapat hasil yang unik dari pengujian berdasarkan *single region size*, dimana *filter region size* ukuran 1 selama proses training memiliki posisi dengan akurasi tertinggi, namun di epoch 10 nilai akurasi turun dibawah *filter region size* ukuran 2. Maka dari itu *filter region size* 2 tetap menjadi hasil terbaik didalam pengaruh *filter region size*.



**Gambar 6.21 Grafik Perubahan Akurasi**

Berdasarkan gambar 6.21 trend akurasi semakin menurun seiring semakin besarnya ukuran *filter region size* yang besar.

### 6.7.3.1.2 Pengaruh *Multi Region Size* dan *Feature Maps*

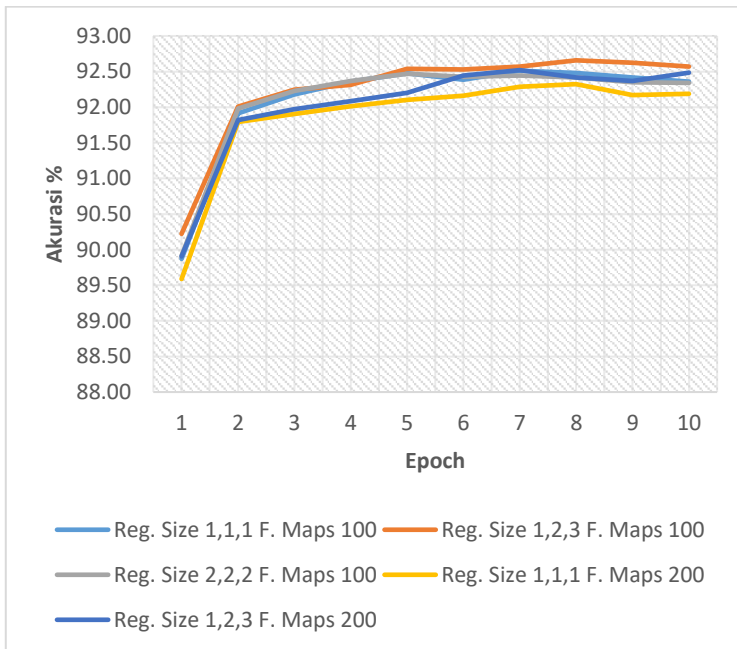
Skenario dua adalah menguji efek perubahan nilai *multi-region size* dan perubahan nilai *feature maps* terhadap akurasi model. Nilai *multi-region size* yang digunakan dalam skenario ini adalah 2,2,2, 2,3,4, 1,2,3 berdasarkan hasil dari skenario sebelumnya, *feature maps* yang digunakan adalah 100 dan 200, berikut hasil percobaanya.

**Tabel 6.22 Hasil Akurasi Model**

Region Size	Feature Maps	Accuracy	F-Measure	Recall	Precision
<b>1,2,3</b>	<b>100</b>	92.36	93.91	94.87	92.98
<b>1,2,3</b>	<b>200</b>	92.57	94.08	95.45	92.75
<b>2,2,2</b>	<b>100</b>	92.34	93.88	95.32	92.49
<b>2,3,4</b>	<b>100</b>	92.19	93.57	95.40	91.82
<b>2,3,4</b>	<b>200</b>	92.49	93.82	95.89	91.85

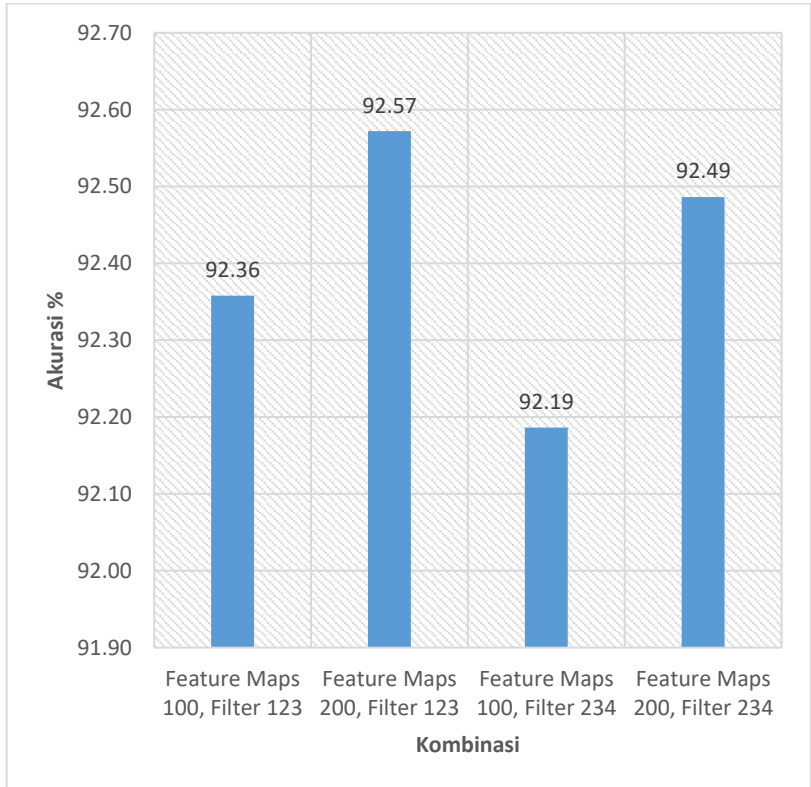


Dari percobaan untuk multi *filter region size*, hasil antara kombinasi *region size* 1,2,3 dan *feature maps* 200 memiliki hasil akurasi paling tinggi. Nilai akurasi yang didapatkan dari kombinasi diatas hanya memiliki perbedaan yang cukup kecil, karena hanya berbeda angka dibelakang koma saja. Terkait hasil pengukuran berdasarkan *F-Measure* menyatakan hal yang sama dengan hasil berdasarkan akurasi sebelumnya, sedangkan hasil berdasarkan *recall* kombinasi *region size* 2,3,4 dan *feature maps* 200 memiliki nilai terbesar dan kombinasi *region size* 1,2,3 dan *feature maps* 200 berada diposisi kedua. Berdasarkan nilai *precision* kombinasi *region size* 1,2,3 dan *feature maps* 100 berada ditingkat pertama sedangkan kombinasi *region size* 1,2,3 dan *feature maps* 200 berada diposisi kedua. Karena kombinasi *region size* 1,2,3 dan *feature maps* 200 berada di posisi kedua baik dari *recall* dan *precision* maka memberikan nilai *F-Measure* paling tinggi.



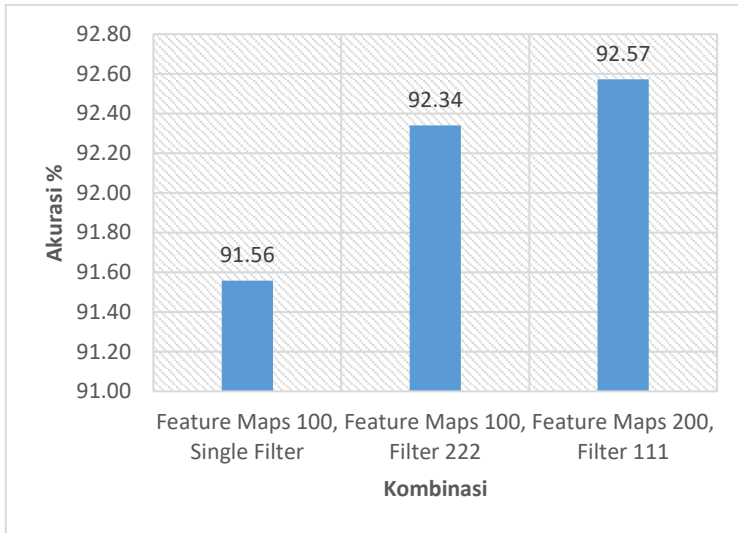
**Gambar 6.22 Perbandingan Akurasi Model**

Berdasarkan akurasi di proses *training* disetiap *epoch*, kombinasi *filter region size* 1,2,3 dan *feature maps* 200 berada diposisi teratas dibanding kombinasi lainnya, terutama di *epoch* 5 hingga *epoch* 10



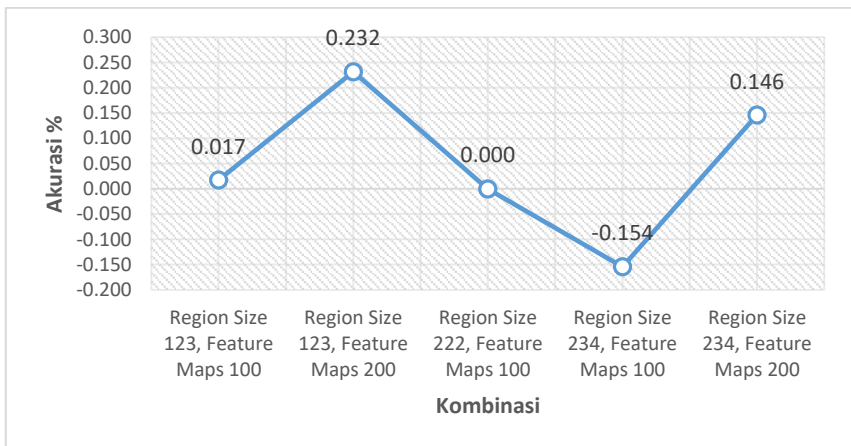
**Gambar 6.23 Perbandingan Akurasi Model**

Pengaruh *feature maps* terhadap akurasi tidak terlalu signifikan untuk *model embedding static* didalam *subtask B* ini. Peningkatan akurasi hanya 0.21 % untuk *region size* 1,2,3 sedangkan *region size* 2,3,4 meningkat sebesar 0.30 %.



**Gambar 6.24 Perbandingan Akurasi Model**

Sedangkan pengaruh jumlah *filter region size* yang diterapkan, meningkatkan akurasi hingga 1.01 % untuk *multi filter region size*.



**Gambar 6.25 Grafik Perbandingan Akurasi**

### 6.7.3.2 Model *Embedding Non-Static*

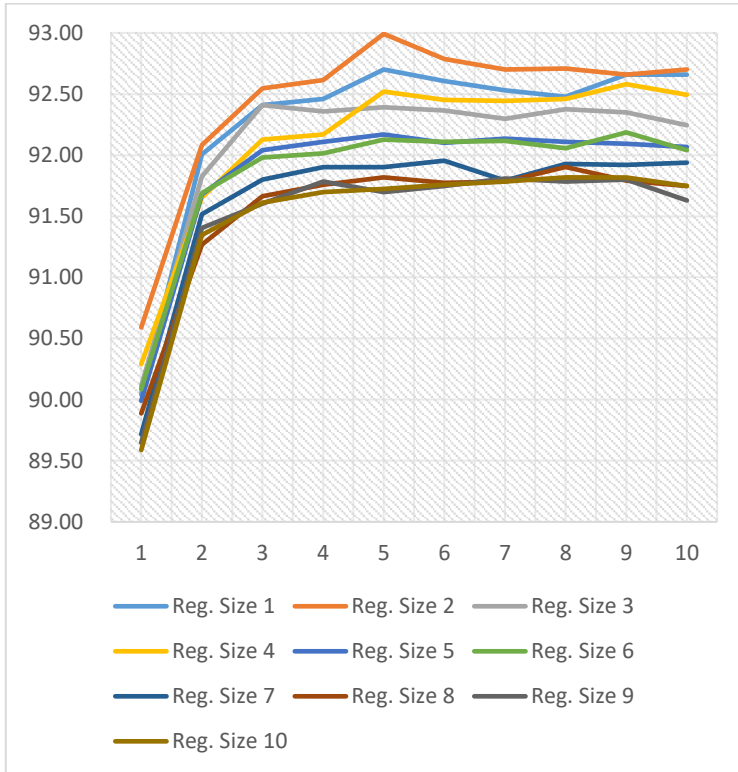
#### 6.7.3.2.1 Pengaruh *Filter Region Size*

Skenario satu adalah menguji efek perubahan nilai *region size* terhadap akurasi model. Nilai *region size* yang digunakan adalah 1,2,3,4,5,6,7,8,9,10 sama seperti percobaan sebelumnya untuk *subtask C*. Pengukuran utama yang digunakan sama seperti *subtask* sebelumnya yaitu akurasi dengan pertimbangan pengukuran lainnya seperti, *F-Measure*, *Recall* dan *Precision*. Berikut hasil pengaruh *filter region size* untuk *model embedding static*.

**Tabel 6.23 Hasil Akurasi model**

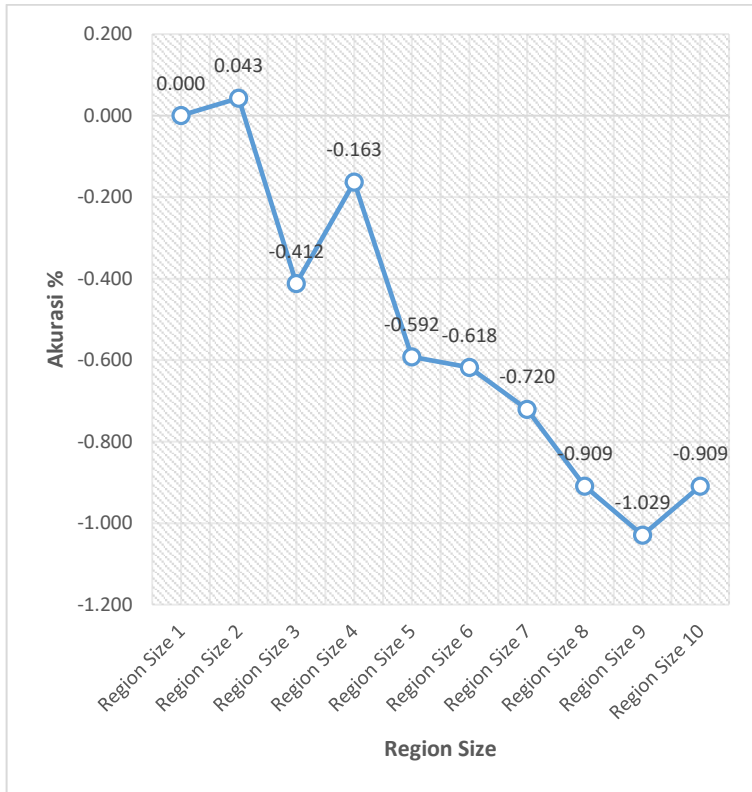
Region Size	Accuracy	F-Measure	Recall	Precision
<b>1</b>	92.66	98.55	98.54	98.57
<b>2</b>	92.70	98.55	98.49	98.61
<b>3</b>	92.25	98.49	98.41	98.57
<b>4</b>	92.50	98.36	98.41	98.32
<b>5</b>	92.07	98.43	98.46	98.40
<b>6</b>	92.04	98.26	98.25	98.27
<b>7</b>	91.94	98.24	98.21	98.27
<b>8</b>	91.75	98.25	98.30	98.22
<b>9</b>	91.63	98.06	98.16	97.97
<b>10</b>	91.75	98.27	98.28	98.27

Dari hasil tersebut perbedaan nilai akurasi dari hasil yang paling rendah dengan yang paling bagus cukup sedikit yaitu sekitar 0.91 %. Sedangkan hasil terbaik sama dengan model *embedding static* sebelumnya yaitu *filter region size 2* dengan nilai akurasi 92.70 %. Nilai *F-Measure* terbaik dalam skenario diatas harus dilihat hingga 5 angka dibelakang koma karena untuk *filter region size 1* dengan 2 terpaut angka yang sangat kecil. Nilai *F-Measure* untuk *filter region size 1* adalah 98.54965 % sedangkan untuk *filter region size 2* adalah 98.54997 %, selisih kedua *filter region size* tersebut adalah 0.00032 %



**Gambar 6.26 Akurasi training model**

Berbeda dengan hasil *model embedding static*, *filter region size* 2 selalu berada diposisi tertinggi untuk akurasi dari *epoch* pertama hingga terakhir.



**Gambar 6.27 Grafik Perubahan Akurasi**

Berdasarkan gambar 6.27 tren akurasi semakin menurun nilainya seiring semakin besarnya ukuran filter *region size*.

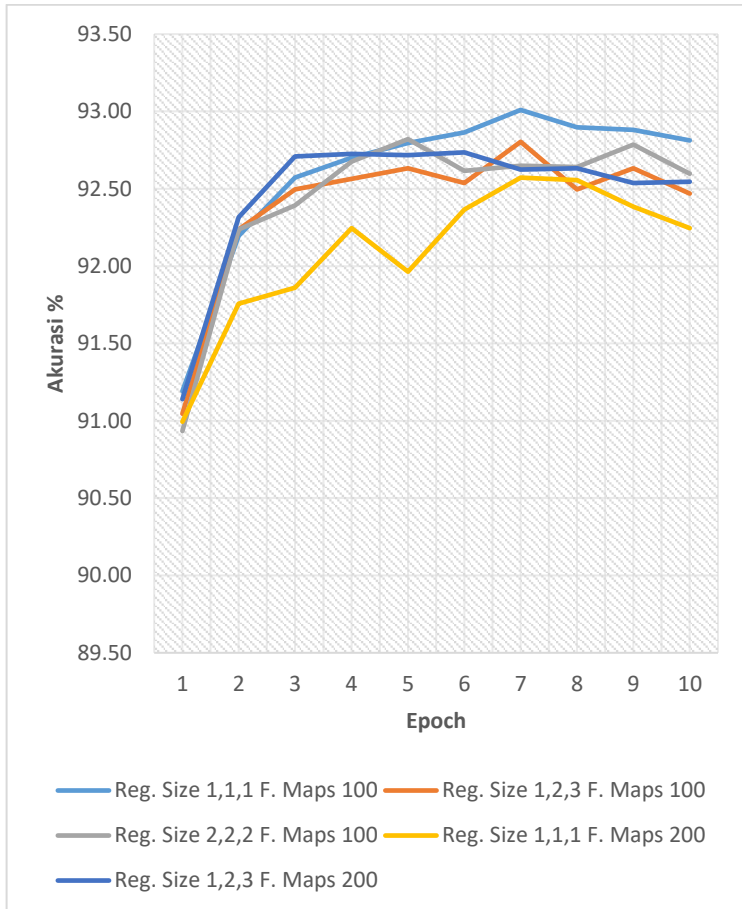
### **6.7.3.2.2 Pengaruh *Multi Region Size* dan *Feature Maps***

Skenario dua adalah menguji efek perubahan nilai *multi-region size* dan perubahan nilai *feature maps* terhadap akurasi model. Nilai *multi-region size* yang digunakan dalam skenario ini adalah 2,2,2, 2,3,4, 1,2,3 berdasarkan hasil dari skenario sebelumnya, *feature maps* yang digunakan adalah 100 dan 200, berikut hasil percobaanya.

Tabel 6.24 Hasil Akurasi Model

Region Size	Feature Maps	Accuracy	F-Measure	Recall	Precision
<b>1,2,3</b>	<b>100</b>	92.47	94.06	94.37	93.78
<b>1,2,3</b>	<b>200</b>	92.81	94.28	94.86	93.73
<b>2,2,2</b>	<b>100</b>	92.60	93.31	93.25	93.41
<b>2,3,4</b>	<b>100</b>	92.25	93.60	93.21	94.00
<b>2,3,4</b>	<b>200</b>	92.55	87.53	86.82	88.31

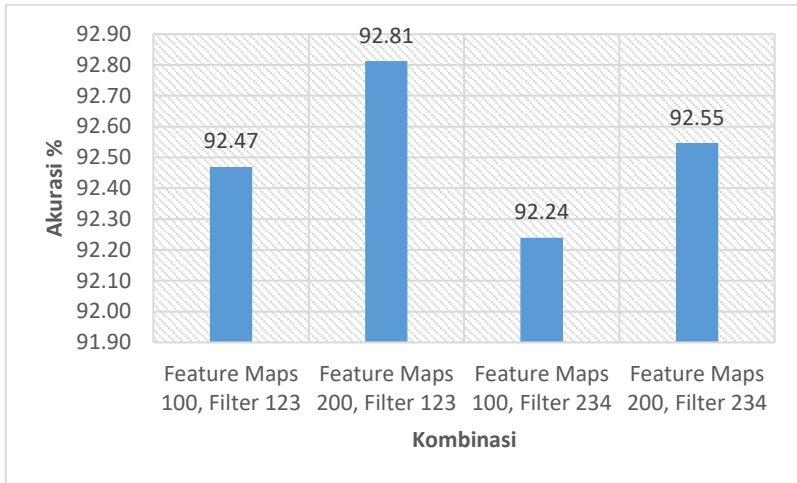
Berdasarkan tabel 6.24 kombinasi *filter region size* 1,2,3 dengan *feature maps* 200 memiliki nilai akurasi tertinggi dengan nilai akurasi 92.81 %. Selisih hasil akurasi tertinggi dengan akurasi terendah cukup kecil yaitu 0.26 %. Berdasarkan nilai *F-Measure* nilai tertinggi adalah kombinasi yang sama yaitu kombinasi *filter region size* 1,2,3 dengan *feature maps* 100 dengan nilai *F-Measure* adalah 94.28 %. Nilai *F-Measure* terendah adalah 87.53 % dengan kombinasi *filter region size* 2,3,4 dengan *feature maps* 200, hal ini cukup jauh dengan selisih 6.75 %. Penyebab rendahnya nilai tersebut adalah nilai *recall* dan *precision* yang rendah.



**Gambar 6.28 Akurasi Training Model**

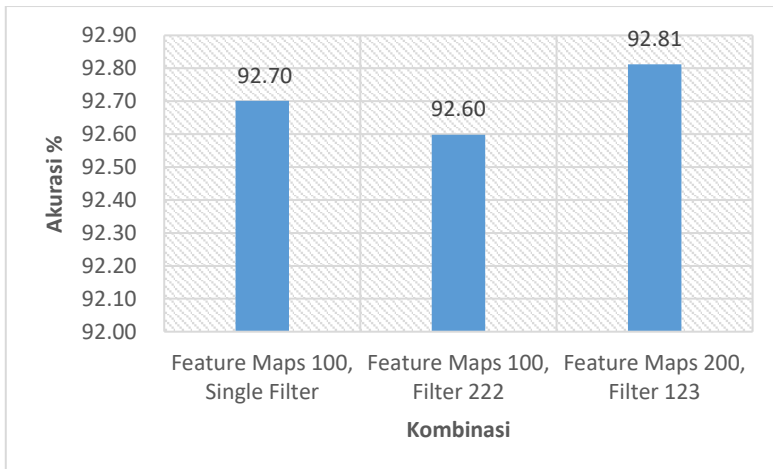
Dari gambar 6.28 kombinasi *filter region size* 1,2,3 dengan *feature maps* 200 pada *epoch* 1 hingga *epoch* 5 kombinasi tersebut memiliki akurasi dibawah kombinasi lainnya, namun akurasi meningkat disaat *epoch* 6 hingga *epoch* 10.





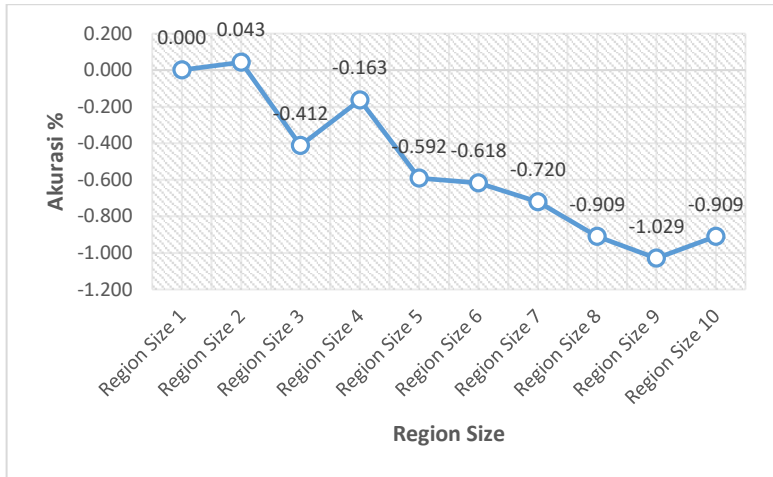
**Gambar 6.29 Perbandingan Akurasi Model**

Berdasarkan pengaruh *feature maps* terhadap akurasi model, sama seperti percobaan sebelumnya dimana *feature maps* lebih besar akan memiliki nilai akurasi yang lebih besar. *Filter region size* 1,2,3 memiliki kenaikan akurasi 0.34 % sedangkan *filter region size* 2,3,4 memiliki kenaikan akurasi 0.31 %.



**Gambar 6.30 Perbandingan Akurasi Model**

Pengaruh jumlah *filter region size* meningkatkan akurasi, namun untuk *multi region size 2,2,2* memiliki akurasi sedikit lebih rendah daripada *single filter region size 2*. Namun mengganti *nilai feature maps* menjadi 200 dan *filter region size* menjadi 1,2,3 dapat meningkatkan akurasi lebih tinggi daripada *single filter region size 2*.



Gambar 6.31 Grafik Perubahan Akurasi

### 6.7.3.3 Pengaruh Model Embedding

Percobaan selanjutnya setelah menguji variasi *single filter region size*, *multiple region size* hingga *feature maps* adalah menguji pengaruh model *word embedding* yang diterapkan untuk proses merubah kata menjadi vektor kata. Sebelum membandingkan pengaruh model *word embedding*, ditentukan terlebih dahulu antara model *embedding static* dengan *non static* yang memiliki akurasi terbaik, berikut hasilnya.

Tabel 6.25 Hasil Akurasi Model

Embedding	Accuracy	F-Measure	Recall	Precision
<b>Static</b>	92.57	94.08	95.45	92.75
<b>Non Static</b>	92.81	94.28	94.86	93.73

Berdasarkan perbandingan diatas nilai akurasi model *embedding non-static* memiliki hasil yang lebih tinggi, dengan silisih peningkatan akurasi 0.24 %. Hasil yang sama juga didapatkan berdasarkan pengukuran *F-Measure* dan *Precision*, tetapi berdasarkan *recall* memberikan hasil yang berbeda yaitu model *embedding static* lebih tinggi dengan nilai *recall* 95.45 %. Maka dari itu untuk melakukan analisis lebih lanjut berikut *confusion matrix* untuk dua model *embedding* tersebut.

A confusion matrix for sentiment classification. The vertical axis is labeled 'PREDICTED' and has three categories: Negatif, Netral, and Positif. The horizontal axis is labeled 'ACTUAL' and also has three categories: Negatif, Netral, and Positif. The matrix cells contain the following counts: (Predicted Negatif, Actual Negatif) = 4630; (Predicted Negatif, Actual Netral) = 57; (Predicted Negatif, Actual Positif) = 138; (Predicted Netral, Actual Negatif) = 253; (Predicted Netral, Actual Netral) = 1865; (Predicted Netral, Actual Positif) = 210; (Predicted Positif, Actual Negatif) = 190; (Predicted Positif, Actual Netral) = 73; (Predicted Positif, Actual Positif) = 4243. The cell for (Predicted Netral, Actual Netral) is highlighted in a darker purple color.

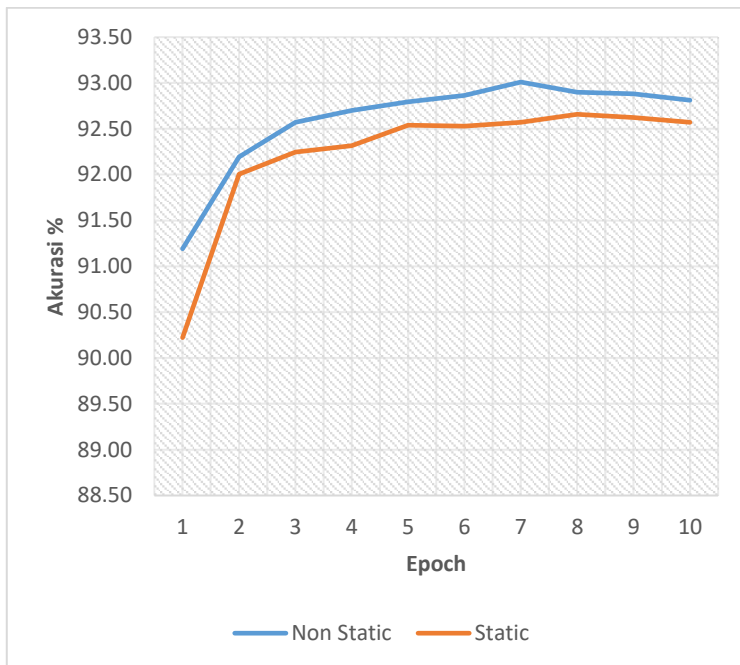
PREDICTED	ACTUAL		
	Negatif	Netral	Positif
Negatif	4630	57	138
Netral	253	1865	210
Positif	190	73	4243

Gambar 6.32 *Confusion Matrix Static*

PREDICTED	Negatif	4627	45	153
	Netral	243	1860	225
	Positif	161	70	4275
		Negatif	Netral	Positif
		ACTUAL		

**Gambar 6.33** *Confusion Matrix Non Static*

Nilai recall yang rendah untuk model embedding non-static dikarenakan nilai recall untuk label netral cukup rendah yaitu 79.89 %. Hal tersebut dipengaruhi karena proporsi data netral yang lebih rendah daripada data positif dan negatif, sedangkan nilai recall label netral untuk model embedding static adalah 80.11 %. Selisih dari keduanya cukup rendah yaitu 0.002 %.



**Gambar 6.34 Perbandingan Akurasi**

Berdasarkan gambar 6.34, model *embedding non-static* memiliki tingkat stabilitas akurasi lebih tinggi disetiap *epoch* nya. Hal tersebut dilihat dari standard deviasi yang lebih rendah yaitu 0.514499 dari *epoch* 1 hingga 10, sedangkan untuk model *embedding static* adalah 0.696116. Perbedaan ini dipengaruhi karena *mode non-static* menggunakan *adadelata optimizer* untuk memperbarui berat vektor kata. Sehingga proses *training* model *embedding non-static* lebih efisien. Hasil ini sama dengan subtask C sebelumnya.

#### 6.7.3.4 Pengaruh Model Word Embedding

Pengaruh model *word embedding* terhadap akurasi cukup signifikan, terutama saat menggunakan *learning alogrithm* yang berbeda. Berikut hasil pengaruh penggunaan model *word embedding* untuk subtask B.

Tabel 6.26 Hasil Akurasi Model

Word Embedding	Accuracy	F-Measure	Recall	Precision
<b>Fasttext</b>	92.30	91.27	90.39	92.66
<b>w2v Skipgram</b>	92.81	94.28	94.86	93.73
<b>w2v CBOW</b>	91.12	89.93	89.01	91.38

Pengaruh penggunaan model *word embedding* terhadap hasil akurasi model cukup tinggi. Selisih yang dihasilkan dari model dengan akurasi terbaik dan akurasi terendah adalah 1.69 %, sedikit lebih kecil dibandingkan *subtask C. learning algorithm skipgram* kembali lebih unggul dibandingkan dengan *learning algorithm cbow*. Algoritma *fasttext* berada diposisi kedua dengan nilai akurasi 92.30 %. Model *word embedding word2vec* dengan *learning algorithm skipgram* memiliki nilai tertinggi berdasarkan pengukuran *f-measure*, *recall* dan *precision* juga.

#### 6.7.4 Subtask A

Percobaan pertama dilakukan untuk *subtask A* dimana memiliki 2 buah kelas.

##### 6.7.4.1 Model Embedding Static

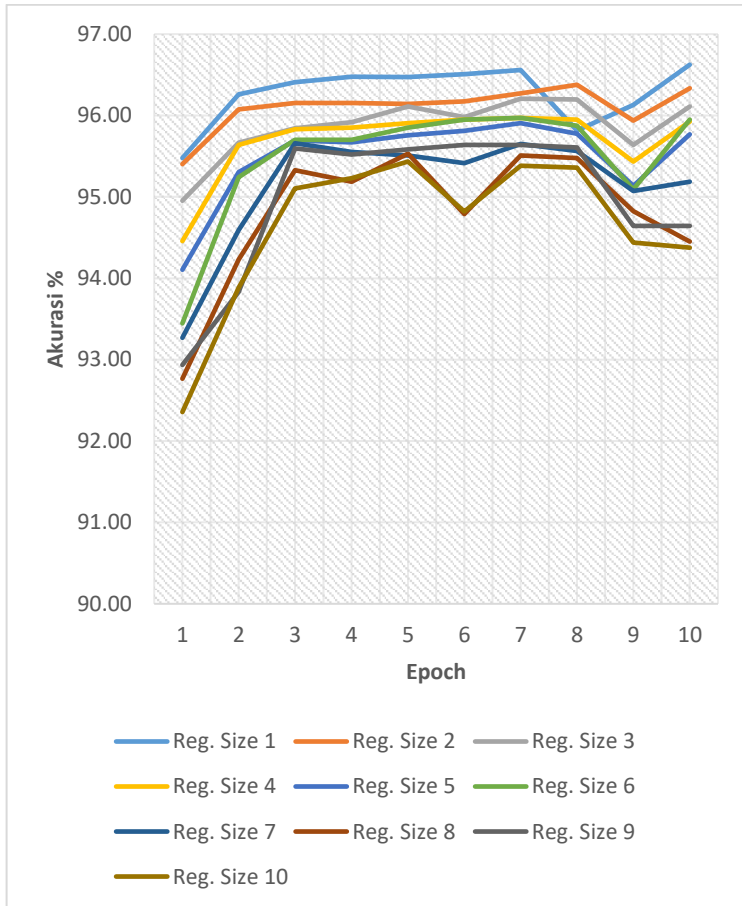
###### 6.7.4.1.1 Pengaruh Filter Region Size

Skenario satu adalah menguji efek perubahan nilai *region size* terhadap akurasi model. Nilai *region size* yang digunakan adalah 1,2,3,4,5,6,7,8,9,10 sama seperti percobaan sebelumnya untuk *subtask B*. Pengukuran utama yang digunakan sama seperti *subtask* sebelumnya yaitu akurasi dengan pertimbangan pengukuran lainya seperti, *F-Measure*, *Recall* dan *Precision*. Berikut hasil pengaruh *filter region size* untuk *model embedding static*.

Tabel 6.27 Hasil Akurasi Model

Region Size	Accuracy	F-Measure	Recall	Precision
<b>1</b>	96.62	96.63	96.85	96.41
<b>2</b>	96.33	96.35	96.83	95.88
<b>3</b>	96.11	96.14	96.87	95.42
<b>4</b>	95.93	95.96	96.62	95.31
<b>5</b>	95.77	95.79	96.37	95.23
<b>6</b>	95.95	95.98	96.59	95.37
<b>7</b>	95.18	95.10	94.74	95.60
<b>8</b>	94.45	94.17	93.22	95.59
<b>9</b>	94.64	94.42	94.10	95.15
<b>10</b>	94.38	94.09	93.64	95.04

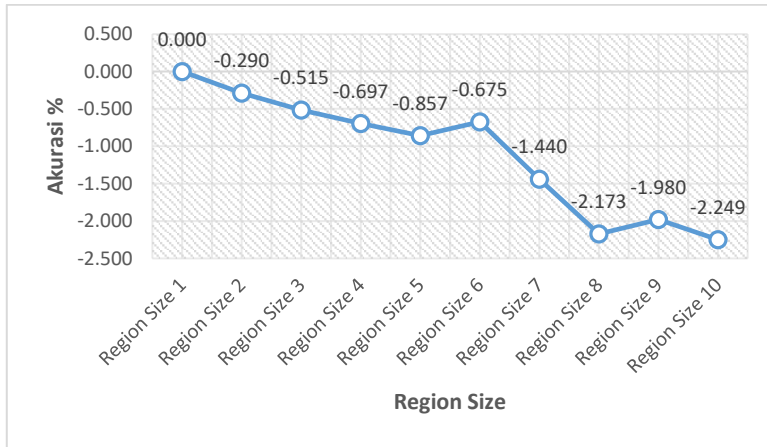
Berdasarkan tabel 6.27 nilai *region size* terbaik untuk *subtask A* dengan model *embedding static* adalah 1 dengan nilai akurasi 96.62 %. Hasil dari pengujian pengaruh *filter region size* selisih akurasi mencapai hingga 2.24 %. Berdasarkan nilai *F-Measure* dan *Precision* menyatakan hasil yang sama dengan pengukuran berdasarkan akurasi, namun berdasarkan berada diposisi kedua dengan nilai *recall* 96.85 %. Nilai tersebut hanya berbeda 0.02 % dengan nilai *recall* terbaik di *filter region size* 3.



**Gambar 6.35 Hasil training model**

Berdasarkan hasil nilai akurasi untuk disetiap *epoch* saat proses training nilai *filter region size* 1 memiliki nilai akurasi tertinggi hingga *epoch* 7 dan pada *epoch* 8 sempat mengalami penurunan akurasi, namun sempat naik hingga mencapai akurasi 96.62 % di *epoch* terakhir. Berdasarkan hasil ini maka kombinasi untuk *multi region size* adalah 1,1,1 dan 1,2,3 untuk *feature maps* 100 dan 200.





**Gambar 6.36 Grafik Perubahan Akurasi**

Berdasarkan gambar 6.36 akurasi model semakin menurun seiring dengan semakin tingginya nilai *filter region size*

#### 6.7.4.1.2 Pengaruh *Multi Region Size* dan *Feature Maps*

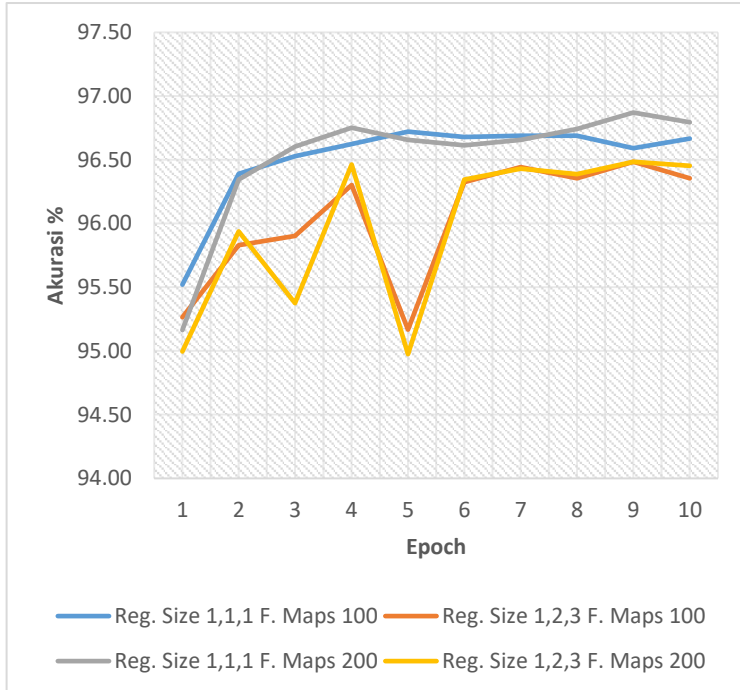
Skenario dua adalah menguji efek perubahan nilai *multi-region size* dan perubahan nilai *feature maps* terhadap akurasi model. Nilai *multi-region size* yang digunakan dalam skenario ini adalah 1,1,1, dan 1,2,3 berdasarkan hasil dari skenario sebelumnya, *feature maps* yang digunakan adalah 100 dan 200, berikut hasil percobaanya.

**Tabel 6.28 Hasil Akurasi Model**

Region Size	Feature Maps	Accuracy	F-Measure	Recall	Precision
<b>1,1,1</b>	<b>100</b>	96.67	96.49	95.44	97.56
<b>1,2,3</b>	<b>100</b>	96.36	96.32	95.55	97.10
<b>1,1,1</b>	<b>200</b>	96.80	96.80	97.16	96.45
<b>1,2,3</b>	<b>200</b>	96.45	96.42	95.73	97.13

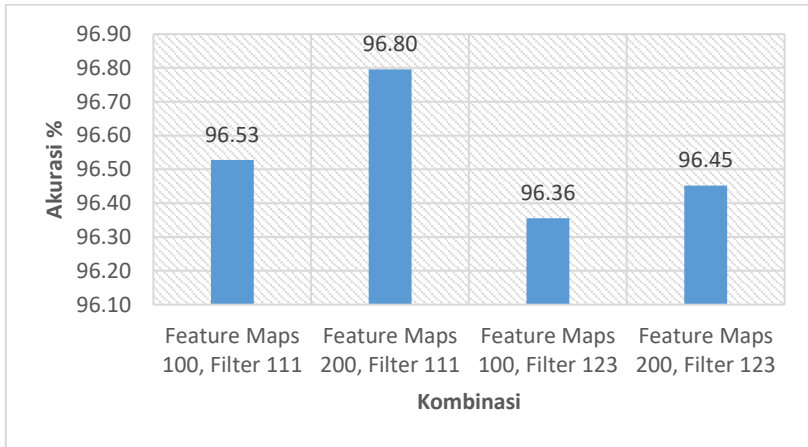
Berdasarkan tabel 6.28 kombinasi *filter region size* 1,1,1 dengan *feature maps* 200 memiliki tingkat akurasi tertinggi

dengan nilai akurasi mencapai 96.80 %. Selisih akurasi tertinggi dengan terendah relatif kecil yaitu 0.44 %. Berdasarkan nilai *F-Measure* kombinasi tersebut berada di posisi pertama juga.



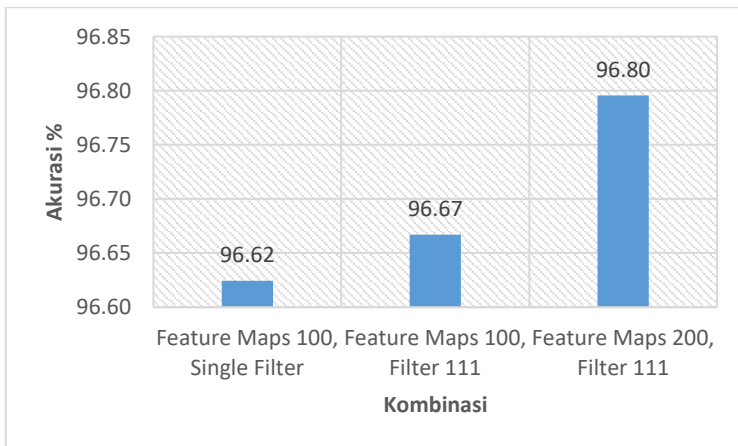
**Gambar 6.37 Akurasi Training Model**

Berdasarkan hasil akurasi disetiap *epoch* Berdasarkan tabel diatas kombinasi *filter region size* 1,1,1 dengan *featue maps* 200 memiliki nilai akurasi tertinggi sejak *epoch* 3. *Filter region size* 1,1,1 memiliki kestabilan nilai akurasi lebih tinggi dibandingkan kombinasi lainnya.



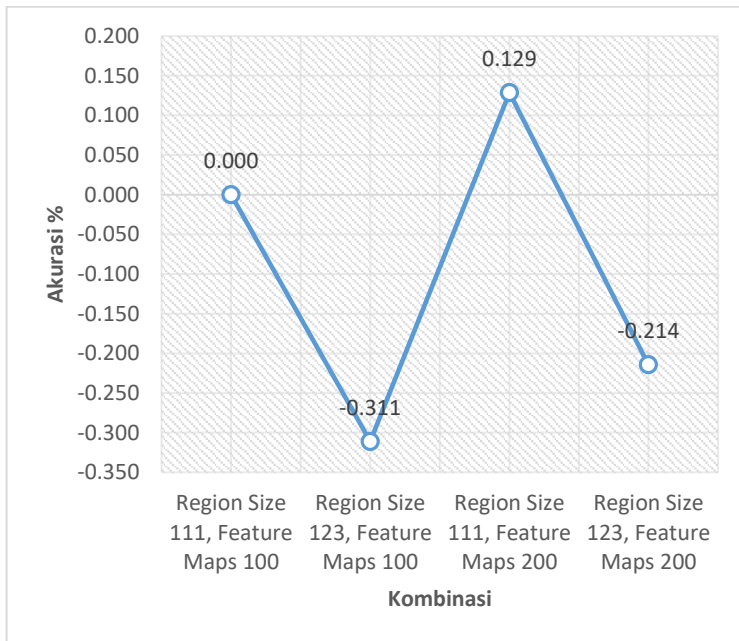
**Gambar 6.38 Perbandingan Akurasi Model**

Berdasarkan pengaruh *feature maps* terhadap akurasi model, sama seperti percobaan sebelumnya dimana *feature maps* lebih besar akan memiliki nilai akurasi yang lebih besar. *Filter region size* 1,1,1 memiliki kenaikan akurasi 0.27 % sedangkan *filter region size* 1,2,3 memiliki kenaikan akurasi 0.1 %.



**Gambar 6.39 Perbandingan Akurasi Model**

Menggunakan multi region size untuk subtask A tampaknya juga dapat meningkatkan akurasi seperti subtask lainnya walaupun hanya sedikit, yaitu 0.18 %.



Gambar 6.40 Grafik Perubahan Akurasi

## 6.7.4.2 Model *Embedding Non-Static*

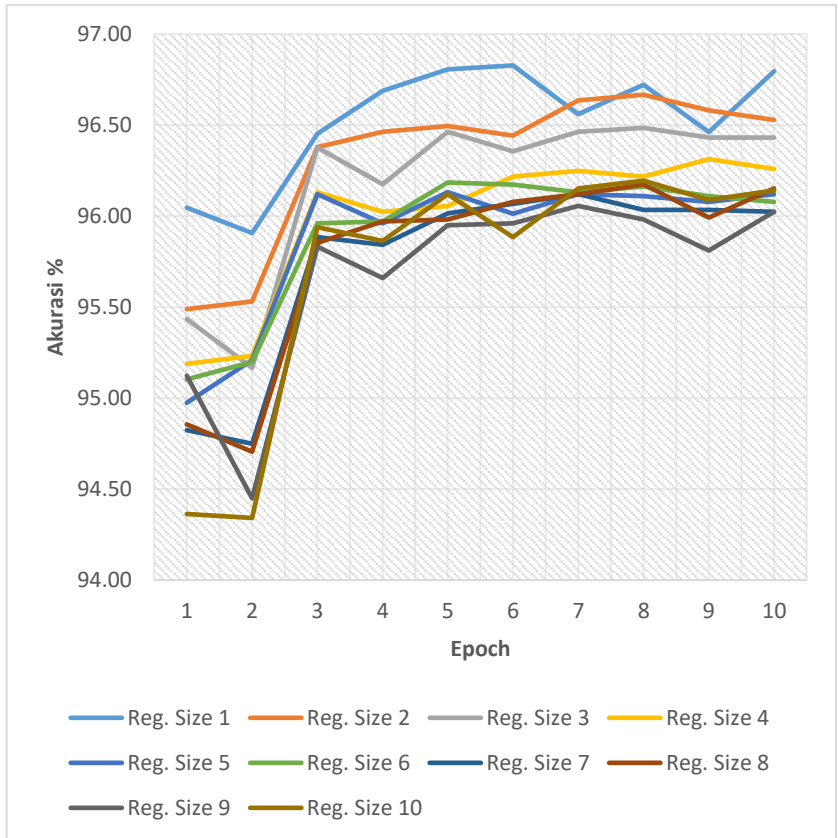
### 6.7.4.2.1 Pengaruh *Filter Region Size*

Skenario satu adalah menguji efek perubahan nilai *region size* terhadap akurasi model. Nilai *region size* yang digunakan adalah 1,2,3,4,5,6,7,8,9,10 sama seperti percobaan sebelumnya untuk *subtask* B. Pengukuran utama yang digunakan sama seperti *subtask* sebelumnya yaitu akurasi dengan pertimbangan pengukuran lainya seperti, *F-Measure*, *Recall* dan *Precision*. Berikut hasil pengaruh *filter region size* untuk *model embedding static*.

**Tabel 6.29 Hasil Akurasi Model**

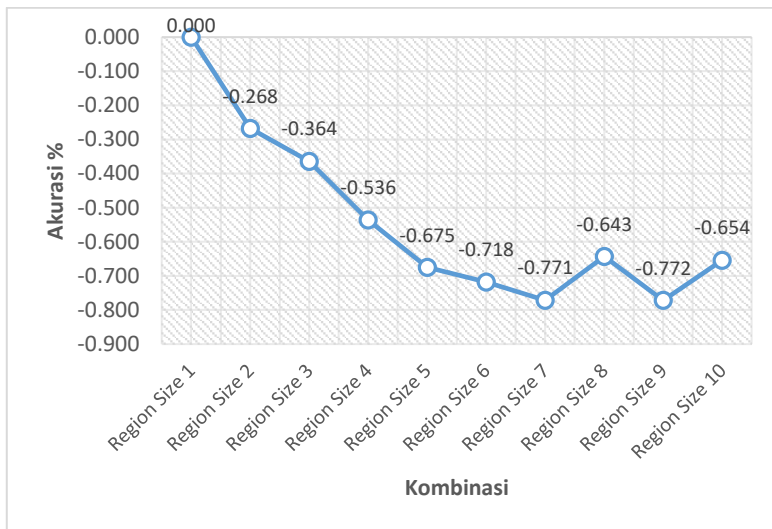
Region Size	Accuracy	F-Measure	Recall	Precision
<b>1</b>	96.80	96.80	97.39	96.22
<b>2</b>	96.53	96.54	96.95	96.13
<b>3</b>	96.43	96.45	97.00	95.90
<b>4</b>	96.26	96.28	96.80	95.76
<b>5</b>	96.12	96.14	96.78	95.52
<b>6</b>	96.08	96.09	96.49	95.70
<b>7</b>	96.02	96.05	96.70	95.43
<b>8</b>	96.15	96.17	96.71	95.65
<b>9</b>	96.02	96.05	96.72	95.39
<b>10</b>	96.14	96.16	96.74	95.59

Dari hasil percobaan berdasarkan *filter region size* untuk model *embedding non-static*, ukuran terbaik berdasarkan akurasi adalah 1. Berdasarkan pengukuran lain menyatakan hal yang sama yaitu 1. Selisih akurasi diantara nilai terendah dengan tertinggi cukup rendah yaitu 0.65 %.



**Gambar 6.41 Akurasi Training Model**

Berdasarkan hasil akurasi disetiap epoch saat proses *training*, *filter region size* 1 memiliki nilai akurasi tertinggi di beberapa epoch. Hasil dari perubahan akurasi, semakin tinggi nilai *filter region size* maka hasil akurasi semakin turun.



**Gambar 6.42 Grafik Perubahan Akurasi**

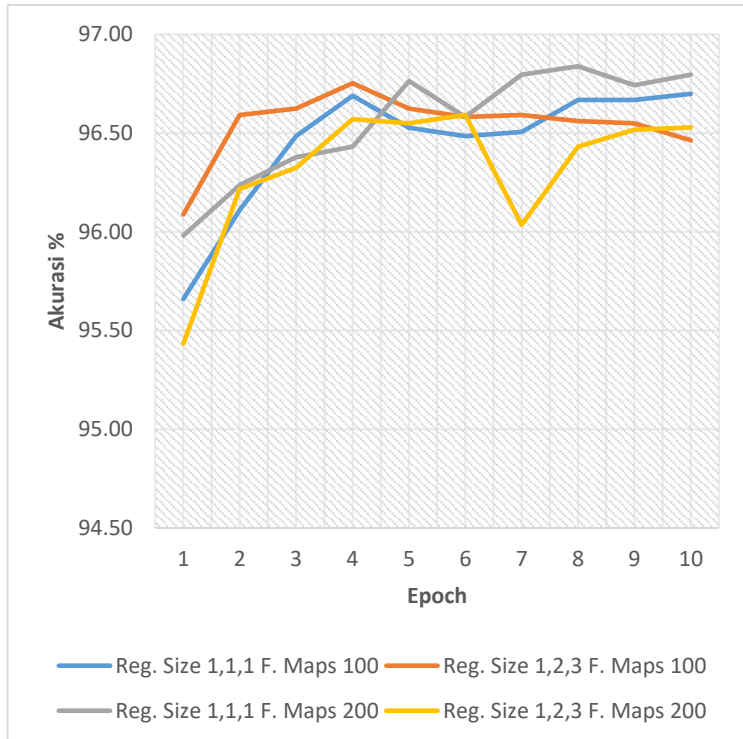
#### 6.7.4.2.2 Pengaruh *Multi Region Size* dan *Feature Maps*

Skenario dua adalah menguji efek perubahan nilai *multi-region size* dan perubahan nilai *feature maps* terhadap akurasi model. Nilai *multi-region size* yang digunakan dalam skenario ini adalah 1,1,1, dan 1,2,3 berdasarkan hasil dari skenario sebelumnya, *feature maps* yang digunakan adalah 100 dan 200, berikut hasil percobaanya.

**Tabel 6.30 Hasil Akurasi Model**

Region Size	Feature Maps	Accuracy	F-Measure	Recall	Precision
<b>1,1,1</b>	<b>100</b>	96.70	96.73	97.99	95.51
<b>1,2,3</b>	<b>100</b>	96.46	96.62	97.37	95.88
<b>1,1,1</b>	<b>200</b>	96.80	96.78	97.95	95.65
<b>1,2,3</b>	<b>200</b>	96.53	96.56	97.64	95.51

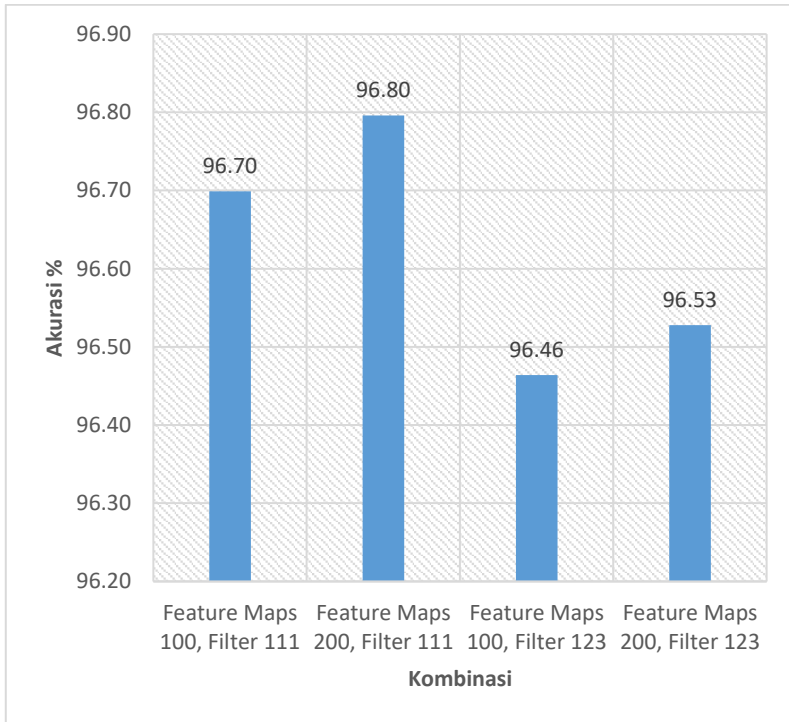
Nilai akurasi tertinggi adalah kombinasi antara filter region size 1,1,1 dan feature maps 200 dimana memiliki nilai akurasi 96.80 %.



**Gambar 6.43 Akurasi Training Model**

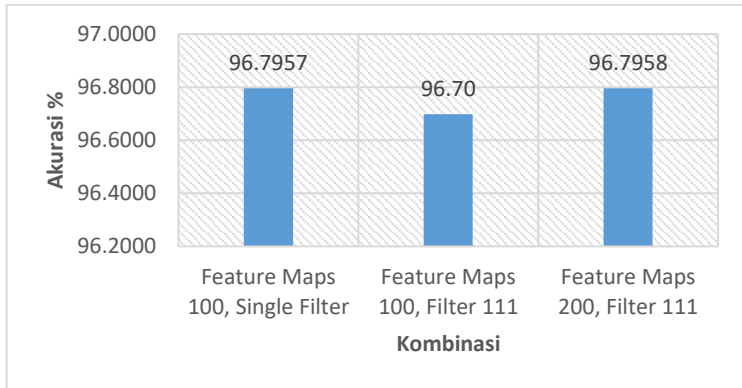
Dari grafik akurasi training disetiap epoch untuk multi filter region size non-static memiliki hasil dengan selisih akurasi yang relatif kecil bahkan untuk disetiap epochnya.





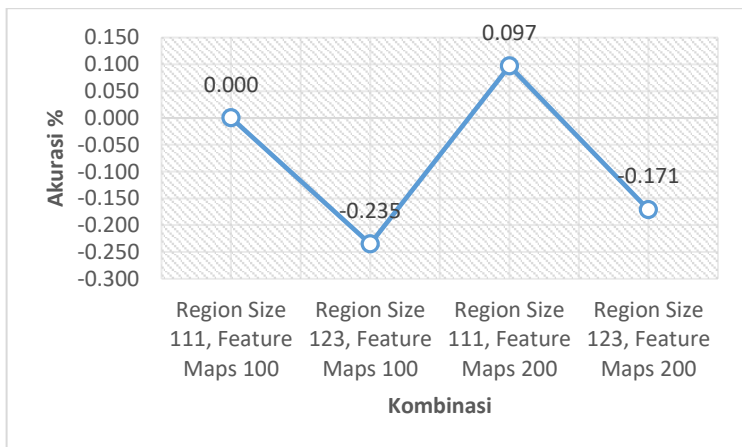
**Gambar 6.44 Perbandingan Akurasi Model**

Berdasarkan pengaruh *feature maps* terhadap akurasi model, sama seperti percobaan sebelumnya dimana *feature maps* lebih besar akan memiliki nilai akurasi yang lebih besar. *Filter region size* 1,1,1 memiliki kenaikan akurasi 0.1 % sedangkan *filter region size* 1,2,3 memiliki kenaikan akurasi 0.06 %. Kenaikan akurasi untuk model *embedding non-static* lebih rendah dibandingkan model *embedding static* yaitu 0.27 %



**Gambar 6.45 Perbandingan Akurasi Model**

Berdasarkan gambar 6.45 terdapat hasil yang cukup berbeda dari percobaan sebelumnya, dimana untuk *feature maps* dan *filter region size* yang sama, *single filter region size* dapat mengalahkan akurasi. Selisih yang dihasilkan tidak terlalu banyak yaitu hanya 0.1 % saja. Apabila menaikkan nilai *feature maps* maka akurasi untuk *multi filter region size* menghasilkan akurasi yang lebih tinggi, namun harus melihat 4 angka dibelakang koma. Multi *filter region size* lebih unggul dengan selisih nilai akurasi 0.0001 %



**Gambar 6.46 Grafik Perubahan Akurasi**

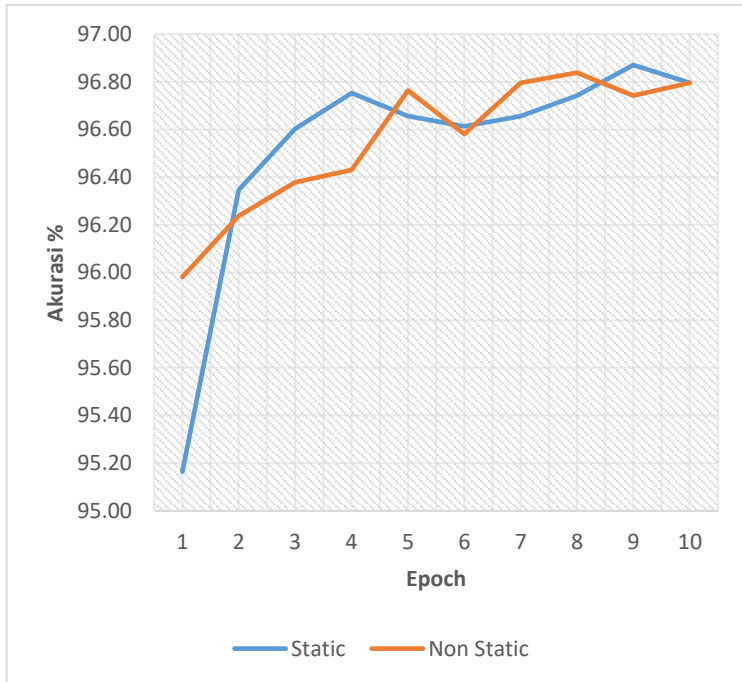
### 6.7.4.3 Pengaruh *Model Embedding*

Percobaan selanjutnya setelah menguji variasi *single filter region size*, *multiple region size* hingga *feature maps* adalah menguji pengaruh model *word embedding* yang diterapkan untuk proses merubah kata menjadi vektor kata. Sebelum membandingkan pengaruh model *word embedding*, ditentukan terlebih dahulu antara model *embedding static* dengan *non static* yang memiliki akurasi terbaik, berikut hasilnya.

**Tabel 6.31 Hasil Akurasi Model**

Embedding	Accuracy	F-Measure	Recall	Precision
<b>Static</b>	96.80	96.80	97.16	96.45
<b>Non Static</b>	96.80	96.78	97.95	95.65

Nilai dari pengukuran akurasi juga sangat tipis dibedakan berdasarkan model *embedding* yang diterapkan, dimana model *embedding non static* unggul dengan selisih akurasi 0.0002 % dibandingkan model akurasi *non static*. Berdasarkan pengukuran lainnya perbedaan sangat kecil diantara model *embedding static* dengan *non static* untuk *subtask A*. sedangkan pengaruh model *embedding* terhadap stabilitas training dapat dilihat didalam grafik berikut ini.



**Gambar 6.47 Akurasi Training Model**

Dari gambar 6.47 terlihat nilai akurasi disetiap *epoch* untuk model *embedding non static* memiliki stabilitas yang lebih baik. Berdasarkan nilai standard deviasi, model *embedding non static* memiliki nilai 0.274257, sedangkan untuk model *embedding static* memiliki nilai 0.471464, sehingga nilai standard deviasi untuk mode *embedding static* dua kali lebih besar daripada model *embedding non-static*. Maka dari itu model *embedding non-static* lebih baik.

#### 6.7.4.4 Pengaruh Model Word Embedding

Pengaruh model *word embedding* terhadap akurasi cukup signifikan, terutama saat menggunakan *learning alogrithm* yang berbeda. Berikut hasil pengaruh penggunaan model *word embedding* untuk subtask B.

Tabel 6.32 Hasil Akurasi Model

Region Size	Accuracy	F-Measure	Recall	Precision
<b>Fasttext</b>	96.62	96.62	96.62	96.63
<b>w2v Skipgram</b>	96.80	96.78	97.95	95.65
<b>w2v CBOW</b>	95.70	95.70	95.70	95.70

Berdasarkan tabel 6.32, pengaruh model *word embedding* untuk *subtask* A sedikit mempengaruhi akurasi model dibandingkan subtask lainnya. Selisih antara model dengan nilai akurasi terbaik dengan nilai akurasi terendah adalah 1.09 %. Hasil dari uji coba ini menghasilkan model dengan *word embedding word2vec skipgram* adalah model *word embedding* dengan nilai akurasi tertinggi.

### 6.7.5 Analisis Antar Subtask

Berdasarkan dari seluruh skenario yang dijalankan, *subtask* A memiliki nilai akurasi tertinggi diantara *subtask* lainnya, dimana model terbaik *subtask* A memiliki nilai akurasi 96.80 %, sedangkan untuk *subtask* B memiliki nilai akurasi 92.81 % dan *subtask* C memiliki nilai akurasi 90.27 %. Hasil akurasi tersebut didapat dari model terbaik disetiap *subtask* dengan konfigurasi yang berbeda. Berikut tabel konfigurasi model terbaik untuk setiap *subtask*.

Tabel 6.33 Perbandingan Konfigurasi Model

Subtask	Word Embed	Learning Algorithm	Model Embed	Region Size	Feature Maps
<b>Subtask A</b>	Word2Vec	Skipgram	Non Static	1,1,1	200
<b>Subtask B</b>	Word2Vec	Skipgram	Non Static	1,2,3	200
<b>Subtask C</b>	Word2Vec	Skipgram	Non Static	1,2,3	200

Dengan masing masing hasil pengukuran sebagai berikut ini.

**Tabel 6.34 Hasil Akurasi Antar Subtask**

Subtask	Accuracy	F-Measure	Recall	Precision
<b>Subtask A</b>	96.80	96.78	97.95	95.65
<b>Subtask B</b>	92.81	94.28	94.86	93.73
<b>Subtask C</b>	90.87	87.88	88.24	87.64

Berdasarkan hasil tersebut, salah satu faktor yang membuat akurasi untuk *subtask A* adalah yang terbaik merupakan sedikitnya jumlah label yang harus ditentukan. *Subtask A* hanya memprediksi *tweet* apakah sentimen yang dimiliki merupakan positif atau negatif, sehingga peluang untuk melakukan klasifikasi di kelas yang benar semakin besar yaitu 50 %. Selain itu dengan jumlah kelas yang sedikit maka model akan lebih mudah mendapatkan ‘fitur’ untuk menentukan apakah itu merupakan kelas negatif atau kelas positif dari *dataset* yang diberikan. Jumlah *dataset* yang berbeda mungkin mempengaruhi, namun tidak dapat dipastikan didalam penelitian ini karena *subtask A* menggunakan *dataset* dengan jumlah label negatif 4669 dan label positif 4662 dengan total data 9331 data, dimana *subtask* lain menggunakan jumlah data 11659.

Berdasarkan peningkatan akurasi disetiap subtask, *subtask C* merupakan subtask dengan peningkatan akurasi terbaik, meningkat sebesar 5.09 %. Peningkatan akurasi didapatkan dari pengurangan akurasi model dengan nilai akurasi tertinggi dengan model akurasi terendah. Sedangkan untuk *subtask B* mengalami peningkatan akurasi sebesar 3.23 % dan subtask A mengalami peningkatan akurasi 2.42 %. Berdasarkan hasil peningkatan akurasi semakin tinggi akurasi model maka semakin susah untuk ditingkatkan akurasinya. Hal tersebut dibuktikan dari *subtask A* dimana model dengan akurasi

terendah adalah 94.38 % sedangkan model dengan akurasi terbaik adalah 96.80 %, sedangkan *subtask C* model dengan akurasi terendah merupakan 85.78 % dan dapat ditingkatkan menjadi 90.87 %.

**Tabel 6.35 Hasil Perbandingan Akurasi *Single Filter Size* Per Subtask**

Reg. Size	Akurasi					
	Static			Non Static		
	Subtask			Subtask		
	A	B	C	A	B	C
1	96.80	91.49	88.11	96.62	92.66	89.92
2	96.53	91.56	87.70	96.33	92.70	89.83
3	96.43	91.12	87.47	96.11	92.25	89.74
4	96.26	90.89	87.13	95.93	92.50	89.23
5	96.12	90.74	86.52	95.77	92.07	89.30
6	96.08	90.46	86.46	95.95	92.04	88.73
7	96.02	90.30	86.44	95.18	91.94	88.62
8	96.15	89.84	85.93	94.45	91.75	88.59
9	96.02	89.78	85.51	94.64	91.63	88.22
10	96.14	89.58	85.78	94.38	91.75	88.33

Penggunaan *filter region size* memiliki pengaruh besar terhadap akurasi model yang dihasilkan, untuk *single filter region size* yang tepat dapat meningkatkan akurasi hingga 2.33 % untuk *subtask C*, sedangkan *subtask B* dapat meningkatkan akurasi 1.97 % dan *subtask A* meningkatkan sebesar 2.25 %. *Single filter region size* paling optimal berdasarkan ketiga *subtask* tersebut adalah 1 dan 2. Salah satu faktor yang mempengaruhi mengapa ukuran *filter region size* yang optimal relatif rendah adalah *dataset* yang digunakan berdasarkan dari twitter yang memiliki jumlah *tweet* 280 karakter, sehingga kalimat yang dihasilkan relatif singkat.

Tabel 6.36 Tabel Perbandingan Akurasi Multi Region Size antar Subtask

Reg. Size	F. Maps	Akurasi					
		Static			Non Static		
		Subtask			Subtask		
		A	B	C	A	B	C
1,1,1	100	96.67	-	89.62	96.70	-	90.54
1,2,3	100	96.36	92.36	89.57	96.46	92.47	90.67
1,1,1	200	96.80	-	89.95	96.80	-	90.87
1,2,3	200	96.45	92.57	89.72	96.53	92.81	90.76
2,2,2	100	-	92.34	-	-	92.60	-
2,3,4	100	-	92.19	-	-	92.25	-
2,3,4	200	-	92.49	-	-	92.55	-

Penggunaan *multi filter region size* terbukti dapat meningkatkan performa model. Akurasi dapat meningkat hingga 1.84 % untuk *subtask C*. Penggunaan kombinasi *multi filter region size* didapatkan dari hasil terbaik *single filter region size* dimana terdapat dua jenis variasi yaitu ukuran *filter* sejenis dan ukuran filter yang meningkat, contoh hasil terbaik adalah 2 maka kombinasi *multi filter region size* sejenis adalah 2,2,2, untuk kombinasi yang meningkat adalah 1,2,3 dan 2,3,4. Berdasarkan percobaan diatas penggunaan variasi tersebut tidak menentukan akurasi terbaik, diantara ketiga *subtask* memberikan hasil yang berbeda dimana *subtask A* kombinasi *multi filter region size* terbaik adalah 1,1,1 sedangkan di *subtask B* dan *C* adalah 1,2,3. Terdapat anomali pada hasil pengaruh jumlah *filter region size* yang digunakan, tepatnya di *subtask A*, ketika *single filter region size* dapat mengalahkan *multi filter region size* dengan selisih akurasi yang sangat kecil. Namun dengan menggunakan kombinasi lain *multi filter region size* dengan *feature maps* yang berbeda dapat mengalahkan nilai akurasi dari *single filter region size* tersebut.



Berdasarkan hasil skenario pengaruh *filter region size*, hasil optimal untuk masing masing subtask A, B dan C adalah 1, 2 dan 1. Salah satu faktor yang membuat *filter region size* optimal relatif kecil adalah rata rata jumlah kata per *tweet* adalah 19. Hal tersebut dipengaruhi oleh batas maksimal karakter yang diberikan oleh twitter adalah 280 karakter. Berbeda dengan penelitian sebelumnya yang menggunakan data relatif lebih panjang dengan rata rata jumlah kata adalah 24.

Penggunaan *feature maps* dapat mempengaruhi nilai akurasi, namun tidak terlalu signifikan dibandingkan pengaruh *filter region size*. Dari percobaan diatas ukuran *feature maps* yang lebih besar dapat meningkatkan nilai akurasi. Perbedaan nilai akurasi dapat mencapai 0.33 % untuk *subtask C*. Ukuran *feature maps* dibatasi hingga 200 karena keterbatasan kemampuan hardware. Berdasarkan hasil yang didapat *feature maps* lebih besar berarti memiliki ukuran dimensionalitas vektor yang lebih besar. Hal ini dapat menyimpan informasi yang lebih banyak sehingga dapat meningkatkan akurasi model yang dihasilkan.

**Tabel 6.37 Perbandingan Akurasi Berdasarkan Model *Embedding***

Model	Akurasi			Std Deviasi		
	Subtask			Subtask		
	A	B	C	A	B	C
<b>Static</b>	96.80	92.57	89.95	0.471	0.696	1.286
<b>Non Static</b>	96.80	92.81	90.87	0.274	0.514	1.228

Penggunaan *model embedding* mempengaruhi akurasi namun, lebih mempengaruhi stabilitas nilai akurasi dalam proses *training*. Penggunaan *model embedding non-static* selalu memiliki hasil akurasi yang lebih tinggi dibandingkan *static*, dikarenakan penggunaan *optimizer adadelta* yang selalu memperbarui nilai vektor kata disetiap *epoch* nya. hal tersebut terbukti efektif meningkatkan stabilitas akurasi didalam fase training dan meningkatkan nilai akurasi akhir walaupun hanya

sedikit. Pengukuran stabilitas diukur menggunakan standard deviasi dimana semakin kecil maka semakin stabil nilainya. Model *embedding non static* dapat menurunkan standard deviasi hingga 0.2 untuk *subtask A* dan dapat meningkatkan akurasi sebesar 1.24 % untuk *subtask B*.

Penggunaan model *word embedding* memiliki pengaruh besar setelah *filter region size* terhadap akurasi, dimana penentuan algoritma dan *learning algorithm* berpengaruh. Dalam menentukan performa *word embedding* sedikit sulit karena *word embedding* termasuk dalam *unsupervised learning*, sehingga dibantu dengan menerapkannya dalam *supervised training* yaitu analisis sentimen. *Word embedding* terbaik didapatkan dari tipe *word2vec* dengan *learning algorithm skipgram*. Berdasarkan dari ketiga *subtask* menunjukkan urutan akurasi model dari akurasi tertinggi menuju terendah yang sama yaitu *word2vec skipgram*, *fasttext* dan *word2vec cbow*. Salah satu faktor yang mempengaruhi juga adalah jumlah kata pada *dataset* yang terkandung didalam model *word embedding*, dimana *fasttext* memiliki jumlah kata yang tidak terkandung lebih banyak. Salah satu penyebab model *word2vec* memiliki kata yang terdapat pada *dataset* yaitu sumber data untuk training model *word2vec* adalah twitter. Twitter merupakan sumber data yang digunakan untuk *training* model CNN begitu juga model *word2vec*. Sedangkan model *fasttext* menggunakan sumber data dari wikipedia bahasa indonesia. Dengan menerapkannya model *word embedding word2vec skipgram* dapat meningkatkan akurasi hingga 3.06 %

## 6.8 Hasil Perbandingan Dengan Algoritma Lain

Hasil akurasi analisis sentimen menggunakan CNN dibandingkan dengan algoritma *naïve bayes* dapat dilihat di tabel berikut ini.

Tabel 6.38 Perbandingan Akurasi Dengan Antar Algoritma

Algoritma Klasifikasi	Akurasi		
	Subtask		
	A	B	C
CNN	96.80	92.81	90.87
Naïve Bayes	91.68	84.42	77.44

Berdasarkan tabel diatas algoritma CNN dapat mengalahkan nilai akurasi yang dihasilkan dari algoritma *Naïve bayes*. Hasil perbandingan yang dilakukan menggunakan CNN dengan parameter terbaik yang telah di *tuning* didalam penelitian ini. Berdasarkan hasil tersebut CNN dapat mengungguli akurasi yang dihasilkan dari *Naïve Bayes* hingga 14.43 % untuk *subtask* C, 8.39 % untuk *subtask* B dan 5.12 % untuk *subtask* A.

## 6.9 Uji Signifikansi

Hasil analisis yang didapatkan pada setiap *subtask* dapat diperkuat dengan melakukan uji signifikansi di beberapa skenario. Berikut uji signifikansi terhadap beberapa skenario.

### 6.9.1 Pengaruh Model Embedding Subtask A

Hasil yang didapatkan dalam skenario perbandingan model *embedding* di *subtask* A memiliki hasil yang sangat tipis, maka dari itu uji signifikansi dapat dilakukan. Langkah pertama dengan mengulang proses training sebanyak 15 kali untuk model *embedding static* dan 15 kali untuk *non static*. Berikut hasil akurasinya.

Tabel 6.39 Hasil Akurasi

Non Static	Static
96.69	96.60
96.81	96.72
96.78	96.70

96.76	96.74
96.73	96.66
96.74	96.65
96.69	96.76
96.79	96.68
96.72	96.58
96.79	96.61
96.70	96.75
96.88	96.66
96.83	96.62
96.82	96.65
96.77	96.75

Langkah selanjutnya menentukan hipotesis awal dimana hipotesis awal adalah “Nilai akurasi *non-static* tidak terdapat perbedaan dengan akurasi *static*” dan hipotesis satu adalah “Nilai akurasi *non-static* terdapat perbedaan dengan akurasi *static*”. Langkah selanjutnya adalah menentukan nilai *alpha* yaitu 0.05 dan menghitung df dimana didapatkan dari total data – 2. Nilai df yang didapatkan adalah 28. Selanjutnya menghitung beberapa nilai sebagai berikut.

**Tabel 6.40 Hasil Perhitungan**

	<b>Non Static</b>	<b>Static</b>
Rata Rata	96.767	96.676
Stdev	0.05532	0.05962
Stdev <sup>2</sup>	0.0030598	0.0035539
stdev/15	0.000203992	0.000236931
Total	0.000440923	
Akar	0.020998178	

Nilai P asli didapatkan dari selisih rata rata dan dibagi dengan akar, sehingga mendapatkan nilai 4.33. Berdasarkan dari tabel T nilai P didalam tabel T adalah 2.04941, hal ini berarti apabila nilai P asli diantara -2.04941 dan 2.04941 maka hipotesis awal diterima, sedangkan apabila diluar batas tersebut maka hipotesis awal ditolak dan hipotesis satu diterima. Berdasarkan aturan tersebut maka dalam uji signifikan ini hipotesis satu diterima yaitu “Nilai akurasi *non-static* terdapat perbedaan dengan akurasi *static*”

## BAB VII KESIMPULAN DAN SARAN

Pada bab ini dibahas mengenai kesimpulan dari semua proses yang telah dilakukan dan saran yang dapat diberikan untuk pengembangan yang lebih baik..

### 7.1 Kesimpulan

Kesimpulan yang didapatkan dari proses pengerjaan tugas akhir ini antara lain:

1. Penggunaan metode *crawling* dengan *API* twitter dapat mengumpulkan tweet hingga 62.834.464 dalam waktu 5 bulan, sehingga model *word embedding* dengan data tersebut memiliki akurasi yang lebih tinggi.
2. Metode *pre-processing* dilakukan mulai melakukan *filtering* bahasa untuk data tanpa topik dan dilanjutkan dengan melakukan *cleansing* data untuk kedua macam *dataset* ( tanpa topik dan bertopik ).
3. Pembuatan model analisis sentimen dengan algoritma CNN dapat memanfaatkan *library* berbasis *python* yaitu *pytorch*, dimana dalam implementasinya dapat menggunakan GPU untuk proses *training* model.
4. Menggunakan parameter terbaik dalam pembuatan model analisis sentimen dapat meningkatkan akurasi hingga 11.07 % ( tanpa memperhatikan jumlah label ) dan 5.09 % jika memperhatikan jumlah label.
5. Semakin sedikit jumlah kelas yang harus ditentukan oleh model, maka akurasi dari model akan semakin meningkat. Hal ini dibuktikan dari nilai akurasi *subtask* A yang paling tinggi diantara *subtask* lainnya.
6. Dalam proses pembuatan model *word embedding* parameter terbaik model terbaik adalah *word2vec* dengan *learning algorithm skipgram*. *Learning algorithm skipgram* memiliki nilai akurasi terbaik untuk semua *subtask*.

7. *Filter region size* memiliki pengaruh besar terhadap akurasi, sehingga harus diperhatikan dengan baik.
8. *Feature maps* memiliki pengaruh juga terhadap akurasi, namun tidak sebesar *filter region size*. Menggunakan *feature maps* dengan ukuran besar membutuhkan durasi *training* data lebih lama dan membutuhkan spesifikasi *hardware* yang lebih tinggi.
9. Menggunakan model *embedding non-static* terbukti dapat membuat akurasi selama proses *training* lebih stabil dan dapat meningkatkan akurasi walaupun hanya sedikit.

## 7.2 Saran

Dari pengerjaan tugas akhir ini, adapun beberapa saran untuk pengembangan penelitian kedepan.

1. Menggunakan kombinasi multi filter region size yang lebih variatif, sehingga mendapatkan penemuan yang lebih baru.
2. Memperbanyak pengujian terhadap feature maps yang digunakan, contoh 100 – 800. Pengujian feature maps tetap menggunakan kelipatan 100
3. Melihat pengaruh fungsi aktivasi yang berbeda, seperti menggunakan TanH atau yang lainnya.
4. Memperbanyak variasi word embedding dengan bermain di parameter training word embedding.
5. Memperbanyak variasi model embedding lainnya selain static dan non-static, seperti random dan multichannel
6. Memperbanyak variasi alur konvolusi yang diterapkan pada model CNN.
7. Melihat pengaruh text preprocessing seperti stemming, stopword removal atau regularisasi lainnya terhadap akurasi model.

8. Melihat pengaruh penggunaan epoch yang berbeda selama proses training untuk mengetahui epoch optimal untuk training model.
9. Melihat pengaruh penggunaan folding data yang berbeda selama proses training untuk mengetahui epoch optimal untuk training model.



## DAFTAR PUSTAKA

- [1] Yudhianto, "132 Juta Pengguna Internet Indonesia, 40% Penggila Medsos," 2017. [Online]. Available: <https://inet.detik.com/cyberlife/d-3659956/132-juta-pengguna-internet-indonesia-40-penggila-medsos>. [Accessed: 24-Jan-2018].
- [2] I. N. Solechah, "Pengguna Internet di awal Tahun 2017 meningkat 51%," 2017. [Online]. Available: <https://www.herosoftmedia.co.id/pengguna-internet-di-awal-tahun-2017-meningkat-51/>. [Accessed: 24-Jan-2018].
- [3] APJII, "Penetrasi dan Perilaku Pengguna Internet Indonesia," 2016.
- [4] katadata, "Siapa Operator Seluler yang Mempunyai Pelanggan Terbanyak?," 2017. [Online]. Available: <https://databoks.katadata.co.id/datapublish/2017/05/03/siapa-operator-seluler-yang-mempunyai-pelanggan-terbanyak>. [Accessed: 24-Jan-2018].
- [5] Z. Ye and B. Wallace, "A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification."
- [6] Y. Kim, "Convolutional Neural Networks for Sentence Classification."
- [7] P. Nakov, A. Ritler, S. Rosenthal, F. Sebastiani, and V. Stoyanov, "SemEval-2016 Task 4: Sentiment Analysis in Twitter," 2016.
- [8] F. R. Imam, S. H. Pramono, and E. A. Dahlan, "Implementasi Opinion Mining (Analisis Sentimen) untuk Ekstraksi Data Opini Publik pada Perguruan Tinggi," *J. EECCIS*, vol. 6, 2012.
- [9] L. B, "Processing, Handbook of Natural Language chapter Sentiment Analysis and Analysis, 2nd Edition," 2010.
- [10] V. Chandani, R. Satria, and Purwanto, "Komparasi Algoritma Klasifikasi Machine Learning Dan Feature Selection pada Analisis Sentimen Review Film," *J. Intell. Syst.*, vol. 1, 2015.

- [11] L. Claesson and B. Hansson, “Deep Learning Methods and Applications,” 2017.
- [12] A. Turing, “computing machinery and intelligence,” 1950.
- [13] W. I. Suartika, A. Y. Wijaya, and R. Solaiman, “Klasifikasi Citra Menggunakan Convolutional Neural Network (Cnn) pada Caltech 101.”
- [14] E. D. Liddy, “Natural Language Processing,” 2001.
- [15] R. Sandhu, “Applications of Natural Language Processing.” [Online]. Available: <https://www.lifewire.com/applications-of-natural-language-processing-technology-2495544>. [Accessed: 08-Feb-2018].
- [16] J. Turian, L. Ratinov, and Y. Bengio, “Word representations: A simple and general method for semi-supervised learning,” pp. 384–394, 2010.
- [17] A. Gelbukh, “International Journal of Computational Linguistics and Applications,” vol. 5, 2014.
- [18] R. Rahmanda, “RANCANG BANGUN APLIKASI BERBASIS MICROSERVICE UNTUK KLASIFIKASI SENTIMEN.”
- [19] K. Fukushima, “A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position.,” *Biol Cybern*, vol. 36, no. 4, pp. 193–202, 1980.
- [20] A. heru Kuncoro and R. Dalimi, “Aplikasi Jaringan Syaraf Tiruan Untuk Peramalan Beban Tenaga Listrik Jangka Panjang Pada Sistem Kelistrikan Di Indonesia.”
- [21] J. Wu, “Introduction to Convolutional Neural Networks,” 2017.
- [22] S. Priansya, “NORMALISASI TEKS MEDIA SOSIAL MENGGUNAKAN WORD2VEC, LEVENSHTAIN DISTANCE, DAN JARO-WINKLER DISTANCE.”
- [23] H. Noviyarto, “Pengaruh Perilaku Konsumen Mobile Internet Terhadap Keputusan Pembelian Paket Layanan Data Unlimited Internet CDMA di DKI Jakarta.”
- [24] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “5021-Distributed-Representations-of-Words-and-Phrases-

- and-Their-Compositionality,” pp. 1–9.
- [25] “Implementing a CNN for Text Classification in TensorFlow – WildML.” [Online]. Available: <http://www.wildml.com/2015/12/implementing-a-cnn-for-text-classification-in-tensorflow/>. [Accessed: 26-Feb-2018].

## BIODATA PENULIS



Penulis lahir di Jember pada tanggal 28 Februari 1997. Merupakan anak pertama dari 2 bersaudara. Penulis telah menempuh beberapa pendidikan formal yaitu; SD Al-Furqan Jember, SMPN 3 Jember dan SMAN 1 Jember.

Pada tahun 2014 pasca kelulusan SMA, penulis melanjutkan pendidikan dengan jalur SBMPTN ( tulis ) di Jurusan Sistem Informasi FTIf – Institut Teknologi Sepuluh Nopember (ITS) Surabaya dan terdaftar sebagai mahasiswa dengan NRP 5214100134. Selama menjadi mahasiswa, penulis mengikuti berbagai kegiatan kemahasiswaan seperti beberapa kepanitiaan serta pernah menjabat sebagai ketua divisi aplikasi teknologi HMSI ITS. Selain itu, kegiatan seperti Latihan Ketrampilan Manajemen Mahasiswa pun pernah diikuti hingga Tingkat Dasar. Di bidang akademik, penulis aktif menjadi asisten praktikum desain manajemen jaringan komputer ( DMJK ). Selain itu, penulis sempat meraih posisi sebagai semi finalis kompetisi GEMASTIK yang diselenggarakan oleh Universitas Indonesia. Penulis juga sempat menjadi staf magang di PT. Pertamina Persero.

Pada tahun keempat, karena penulis memiliki ketertarikan di bidang pengolahan data, maka penulis mengambil bidang minat Akuisisi Data dan Diseminasi Informasi (ADDI). Penulis dapat dihubungi melalui email di [adrianafnandika@gmail.com](mailto:adrianafnandika@gmail.com).

*Halaman ini sengaja dikosongkan*

## LAMPIRAN A

### Pengaruh Filter Region Size, Subtask ABC

Subtask	Embedding	Region Size	Akurasi / Epoch									
			1	2	3	4	5	6	7	8	9	10
C	Static	1	85.54	87.76	88.45	88.27	88.96	88.68	89.08	89.12	88.95	88.11
C	Static	2	85.39	87.57	88.18	87.46	88.58	88.51	88.38	88.36	88.32	87.7
C	Static	3	84.48	86.93	88.15	86.57	88	87.88	87.74	88.01	87.79	87.47
C	Static	4	83.68	86.48	87.48	86.33	87.15	87.38	87.59	87.82	87.13	87.13
C	Static	5	82.82	85.68	86.58	85.51	86.77	86.75	86.92	86.8	86.6	86.52
C	Static	6	82.26	85.7	86.5	85.13	86.4	86.57	86.91	86.94	86.33	86.46
C	Static	7	82.14	84.9	86.22	84.54	86.15	86.37	86.62	86.51	86.35	86.44
C	Static	8	81.32	84.88	85.82	84.72	86.06	86.08	85.94	86.22	86.05	85.93
C	Static	9	81.75	84.34	85.43	83.74	85.52	85.66	85.98	85.78	85.64	85.51
C	Static	10	81.08	84.5	85.51	83.89	85.61	85.34	85.97	85.85	85.79	85.78

C	Non - Static	1	85	88.41	89.51	89.72	90.01	90.22	90.08	90.27	90.28	89.92
C	Non - Static	2	85.65	88.7	89.55	89.72	90.03	89.91	89.9	89.89	90.13	89.83
C	Non - Static	3	85.62	88.71	89.14	89.53	89.65	89.72	89.82	89.92	89.88	89.74
C	Non - Static	4	85.37	88.35	88.93	89.06	89.29	89.14	89.25	89.37	89.3	89.23
C	Non - Static	5	85.87	88.51	89.07	89.02	89.38	89.26	89.29	89.37	89.4	89.3
C	Non - Static	6	85.14	88.21	88.71	88.68	88.81	88.77	88.84	88.77	88.8	88.73
C	Non - Static	7	84.99	87.8	88.37	88.39	88.68	88.5	88.57	88.57	88.52	88.62
C	Non - Static	8	84.73	88.1	88.53	88.44	88.78	88.78	88.69	88.67	88.64	88.59
C	Non - Static	9	84.96	87.73	88.08	87.9	88.16	88.24	88.16	88.11	88.15	88.22
C	Non - Static	10	84.6	87.92	88.09	88.23	88.4	88.4	88.33	88.39	88.45	88.33
B	Static	1	90.14	91.48	91.85	91.17	91.88	91.82	92.01	92.07	91.98	91.49
B	Static	2	90.05	91.06	91.49	89.8	91.48	91.43	91.68	91.7	91.46	91.56
B	Static	3	89.34	90.17	91.16	89.19	91.07	91.13	91.39	91.33	91.31	91.12
B	Static	4	89.31	89.48	91.03	88.47	90.22	90.82	90.98	90.91	90.73	90.89

B	Static	5	88.68	88.98	90.29	88.17	89.43	90.25	90.88	90.54	90.61	90.74
B	Static	6	88.8	88.45	90.14	88.38	89.43	90.25	90.6	90.56	90.6	90.46
B	Static	7	87.94	88.24	90.03	88.84	89.15	90.1	90.52	90.25	90.33	90.3
B	Static	8	87.33	88.07	89.66	88.03	88.58	89.57	90.1	89.89	90.07	89.84
B	Static	9	87.69	87.74	89.91	88.08	88.01	89.73	90.22	89.85	90.04	89.78
B	Static	10	87.66	87.67	89.03	87.77	87.8	89.64	90.21	89.69	89.95	89.58
B	Non - Static	1	90.04	92.01	92.41	92.46	92.7	92.61	92.53	92.48	92.66	92.66
B	Non - Static	2	90.59	92.08	92.55	92.62	92.99	92.79	92.7	92.71	92.66	92.7
B	Non - Static	3	90.11	91.83	92.41	92.36	92.39	92.37	92.3	92.38	92.35	92.25
B	Non - Static	4	90.29	91.65	92.13	92.17	92.52	92.45	92.44	92.46	92.58	92.5
B	Non - Static	5	89.99	91.68	92.04	92.11	92.17	92.1	92.13	92.11	92.09	92.07
B	Non - Static	6	90.08	91.69	91.98	92.01	92.13	92.11	92.12	92.06	92.19	92.04
B	Non - Static	7	89.72	91.52	91.8	91.9	91.9	91.95	91.79	91.93	91.92	91.94
B	Non - Static	8	89.89	91.27	91.66	91.76	91.82	91.77	91.78	91.9	91.79	91.75



B	Non - Static	9	89.65	91.41	91.6	91.78	91.7	91.75	91.81	91.78	91.8	91.63
B	Non - Static	10	89.59	91.35	91.61	91.7	91.72	91.76	91.78	91.82	91.82	91.75
A	Static	1	95.48	96.26	96.41	96.47	96.47	96.51	96.56	95.82	96.13	96.62
A	Static	2	95.4	96.08	96.15	96.15	96.14	96.17	96.27	96.38	95.94	96.33
A	Static	3	94.95	95.67	95.84	95.92	96.11	95.98	96.21	96.2	95.64	96.11
A	Static	4	94.46	95.64	95.83	95.85	95.91	95.95	95.97	95.95	95.43	95.93
A	Static	5	94.1	95.31	95.69	95.67	95.76	95.81	95.91	95.78	95.13	95.77
A	Static	6	93.45	95.24	95.7	95.7	95.85	95.95	95.97	95.87	95.09	95.95
A	Static	7	93.27	94.59	95.66	95.55	95.51	95.41	95.65	95.56	95.07	95.18
A	Static	8	92.77	94.23	95.33	95.19	95.53	94.79	95.51	95.48	94.82	94.45
A	Static	9	92.93	93.83	95.59	95.52	95.58	95.64	95.64	95.61	94.64	94.64
A	Static	10	92.36	93.89	95.1	95.23	95.43	94.82	95.38	95.36	94.44	94.38
A	Non - Static	1	96.05	95.91	96.45	96.69	96.81	96.83	96.56	96.72	96.46	96.8
A	Non - Static	2	95.49	95.53	96.38	96.46	96.5	96.44	96.64	96.67	96.58	96.53

A	Non - Static	3	95.44	95.17	96.38	96.17	96.46	96.36	96.46	96.48	96.43	96.43
A	Non - Static	4	95.19	95.23	96.13	96.02	96.06	96.22	96.25	96.22	96.31	96.26
A	Non - Static	5	94.97	95.21	96.12	95.96	96.13	96.01	96.12	96.11	96.08	96.12
A	Non - Static	6	95.1	95.2	95.96	95.97	96.18	96.17	96.13	96.16	96.11	96.08
A	Non - Static	7	94.82	94.75	95.89	95.84	96.01	96.07	96.12	96.04	96.03	96.02
A	Non - Static	8	94.86	94.71	95.85	95.97	95.98	96.08	96.12	96.17	95.99	96.15
A	Non - Static	9	95.12	94.45	95.83	95.66	95.95	95.96	96.06	95.98	95.81	96.02
A	Non - Static	10	94.36	94.34	95.94	95.86	96.12	95.89	96.15	96.19	96.09	96.14

**Pengaruh Filter Region Size, Subtask ABC**

Subtask	Embedding	Region Size	Feature Maps	Akurasi / Epoch									
				1	2	3	4	5	6	7	8	9	10
C	Static	1,1,1	100	84.91	88.28	89	89.44	89.78	89.86	89.79	89.92	89.85	89.62
C	Static	1,2,3	100	85.06	88.21	89.06	89.52	89.52	89.61	89.62	89.59	89.56	89.57
C	Static	1,1,1	200	85.75	88.66	89.08	89.9	89.82	90.24	89.96	89.96	90.14	89.95
C	Static	1,2,3	200	85.41	88.64	89.07	89.36	89.69	89.76	89.72	89.78	89.83	89.72
C	Non-Static	1,1,1	100	86.11	88.81	89.47	90.06	89.97	90.28	90.15	90.59	90.43	90.54
C	Non-Static	1,2,3	100	86.63	89.21	89.7	90.11	90.2	90.2	90.36	90.63	90.38	90.67
C	Non-Static	1,1,1	200	86.89	89.05	89.61	90.43	90.54	90.53	90.52	90.39	90.6	90.76
C	Non-Static	1,2,3	200	86.71	89	89.78	90.44	90.46	90.76	90.81	90.51	90.76	90.87
B	Static	1,2,3	100	89.87	91.91	92.18	92.36	92.48	92.38	92.51	92.48	92.42	92.36

B	Static	1,2,3	200	90.22	92.01	92.25	92.31	92.54	92.53	92.57	92.66	92.62	92.57
B	Static	2,2,2	100	89.91	91.98	92.23	92.37	92.47	92.43	92.44	92.42	92.35	92.34
B	Static	2,3,4	100	89.59	91.79	91.9	92.01	92.1	92.16	92.29	92.32	92.17	92.19
B	Static	2,3,4	200	89.9	91.82	91.97	92.08	92.2	92.44	92.52	92.42	92.37	92.49
B	Non-Static	1,2,3	100	91.05	92.24	92.5	92.56	92.63	92.54	92.8	92.5	92.63	92.47
B	Non-Static	1,2,3	200	91.19	92.19	92.57	92.7	92.8	92.86	93.01	92.9	92.88	92.81
B	Non-Static	2,2,2	100	90.93	92.24	92.39	92.68	92.82	92.62	92.65	92.64	92.79	92.6
B	Non-Static	2,3,4	100	90.99	91.76	91.86	92.25	91.96	92.37	92.57	92.56	92.38	92.25
B	Non-Static	2,3,4	200	91.14	92.31	92.71	92.73	92.72	92.74	92.62	92.63	92.54	92.55
A	Static	1,1,1	100	95.52	96.39	96.53	96.62	96.72	96.68	96.69	96.69	96.59	96.67
A	Static	1,2,3	100	95.26	95.83	95.9	96.3	95.17	96.32	96.44	96.36	96.48	96.36
A	Static	1,1,1	200	95.16	96.35	96.6	96.75	96.66	96.61	96.66	96.74	96.87	96.8
A	Static	1,2,3	200	95	95.94	95.38	96.46	94.97	96.35	96.43	96.39	96.48	96.45
A	Non-Static	1,1,1	100	95.66	96.11	96.48	96.69	96.53	96.48	96.51	96.67	96.67	96.7

A	Non-Static	1,2,3	100	96.09	96.59	96.62	96.75	96.62	96.58	96.59	96.56	96.55	96.46
A	Non-Static	1,1,1	200	95.98	96.24	96.38	96.43	96.76	96.58	96.8	96.84	96.74	96.8
A	Non-Static	1,2,3	200	95.43	96.22	96.32	96.57	96.55	96.59	96.04	96.43	96.52	96.53

**Pengaruh Model Word Embedding, Subtask ABC**

Subtask	Word Embedding	Akurasi / Epoch									
		1	2	3	4	5	6	7	8	9	10
C	Word2Vec CBOW	84.64	85.32	86.19	86.98	86.91	87.01	87.19	87.61	87.62	87.81
C	Word2Vec Skipgram	86.71	89	89.78	90.44	90.46	90.76	90.81	90.51	90.76	90.87
C	Fasttext	86.19	88.66	89.36	89.41	89.97	89.72	89.81	89.84	89.48	89.27
B	Word2Vec CBOW	88.61	89.24	90.3	90.45	90.43	90.77	90.88	91.3	91.15	91.12
B	Word2Vec Skipgram	91.19	92.19	92.57	92.7	92.8	92.86	93.01	92.9	92.88	92.81
B	Fasttext	90.41	91.98	91.95	92.02	92.5	92.38	92.23	92.58	92.49	92.3

A	Word2Vec CBOW	93.97	93.04	94.21	95.73	95.57	95.26	95.1	95.4	95.53	95.7
A	Word2Vec Skipgram	95.98	96.24	96.38	96.43	96.76	96.58	96.8	96.84	96.74	96.8
A	Fasttext	95.24	96.14	96.46	96.66	96.67	96.66	96.62	96.6	96.66	96.62

**Contoh Data untuk Analisis Sentimen**

<b>Kalimat Tweet</b>	<b>Label</b>
@indosatcare Kenapa kartu diregistrasi nggak bisa pak? Padahal nomor kk & ktp nya sudah benar? Sampai 5x gagal, tolong di daftarkan segera, demi kenyamanan pengguna kartu indosat. Terima kasih	Netral
indosat mantap djaja koneksinya , ngebut euy	Sangat Positif
Membalas @IndosatCare @IndosatCare mau nangis deh rasanya pdahl saya percaya betul dengan indosat soal koneksi di wil Karimun tp sudah 3 hari koneksi kacau balau	Negatif
Sinyal Indosat asu ari musim hujan, hadeuh	Sangat Negatif
Membalas @triindonesia Cek dm woi , layananan keluhan semua ga berfungsi...	Negatif

Ini kenapa di bale endah sinyal XL tiba2 bagus yah? Pindah kali yah tower nya ke dekat rumah.. *aneh liat hp sendiri	Sangat Positif
Hotsale 8GB hanya gratis nelfon aja ga ada gratis sms nya tah min? @triindonesia	Positif
Membalas @andikapram_ @IndosatCare dan 2 lainnya Hai kak Andika, Ada yang bisa saya bantu. Kami lihat akhir akhir ini anda sering sambat dengan kualitas jaringan dari @IndosatCare , saya sarankan gunakan Paket Internet dari @Telkomsel agar anda tidak buang tenaga untuk marah marah.	Negatif
Membalas @tsel_malang @tsel_malang iya min. Udah bisa kok. Hihhi. Tengkyu yaaaa :) seneng banget pake telkomsel. Sinyal lancaaar { }	Sangat Positif
@IndosatCare 30 ribu saya beneran atau bohongan saya coba buat internet lngsung tanpa daftar paket internet dan alhamdulillah bisa buat internet trus saya buka halaman indosat.. Baru kebuka saya dpt sms dari indosat "Kamu internetan dg tarif perKB,pemakai	Sangat Positif
Sinyal kencang XL mantabbbzz :D	Positif

<p>@IndosatCare tolong kasih penjelasan yg bisa saya terima dong.. kenapa 3 hari ini sinyal Indosat sangat mengecewakan di daerah Mampang Depok</p>	<p>Netral</p>
<p>yang lagi mau liburan di The Jungle Bogor Indosat Super WIFI juga ada di sana looh, ngenet asik sambil liburan #Mantap @indosatbogor</p>	<p>Positif</p>
<p>Membalas @rapdodge @triindonesia Saya juga cukup heran mengapa 4G 3 speednya bisa naik dirumah akan saya SS sisa paket saya, semoga aja di sekolah cepat membaik (6)</p>	<p>Positif</p>
<p>JIKALAU MASALAH SDM YG MENGAKIBATKAN KELUHAN SAYA TIDAK DIBALAS, COBALAH UNTUK MENAMBAH CS NYA LAGI @triindonesia Maulana Muldan menambahkan, Maulana Muldan @UdanDanski Min balas dm saya @triindonesia</p>	<p>Negatif</p>
<p>@triindonesia operator 3 tolong yah sinyal nya jangan jelek terus tiap hari saya kecewa sekali dengan 3 sekarang.. padahal sdh 3tahun pakai</p>	<p>Negatif</p>



Membalas @Telkomsel mantap jiwa nih min harga paketnya	Sangat Positif
Terima kasih kepada sponsor yg sudah mendukung acara kita, Indosat dan Nafigo. Sehingga acara kita dapat berjalan dengan lancar ?	Positif
@myXL sinyal dimana2 cuma H+ padahal paket internet 4G. Makin mahal makin bobrok aja xl. Kecewa saiah ?	Negatif
Membalas @myXL @Feriza07 woy @myXLCare anjing bukan dikota jakarya saja semua kota. apa lo gak baca twett pengguna lain? makasar aja uda seminggu. sialan lo anjing kampret	Sangat Negatif
- ____ - RT @DianSaaputra: Aseek.. "NICK0MINAJ: Kalo hujan dan malam tri pasti lemot @triindonesia .. Bangsat"	Sangat Negatif
Frontal cuk :)RT @dwikyoutSIDers: Bangsat,bbm+sms pending @triindonesia gatel!	Sangat Negatif
Memang terBAIK sakali iye teh TELKOMSEL , sehari semalam GSM wae ..	Sangat Positif

Ya ampun,, Sinyal @myXL sangat kencang sekali. Sampai <sup>2</sup> gak bisa MaBar Mobile Legend. Berjamaah pula. 55nya. XL. Lag semua. Hebat. Tingkatkan lagi.	Negatif
Indosat sinyale koyo NGELEC.... Asu	Sangat Negatif
@triindonesia gimana nich,internet untuk wilayah serang baru kok lambat,bbm pending terus,buka facebook lama,terima kasih	Netral
Haloo.. @myXL @myXLCare bagaimana nasib keluhan saya tiket C19402752? Apakah masih di tanggapi atau di anggap angin lalu? Sejak Januari masih belum selesaikah? Cc : @YLKI_ID	Netral
Makassar lg seru banget nih guys sore ini, ada event keren, "Celebes Jambore jeep" & XL senang banget bs jd bagian dr event ini pastinya :)	Netral
kecewa berat saiah dengan @XLcare, lebih baik beralih ke @indosat lah...kualitas sinyal nya pasti!!!	Negatif
@IndosatCare min, jaringan indosat lemot bgt dan tolong jangan kasih solusi "update jaringan manual lewat hp" dan sejenisnya. karena yang bermasalah kan bukan hpnya tapi	Netral

indosatnya. jadi tolong kasih solusi yang bener2 bisa dilakukan dan menyelesaikan mas	
17th pakai XL,lancar,cepat pula,tp XL klo udah ngambek gak kira2,3 hari ini @myXL menyiksaku! Inet Lemot,Signal down,Voice angin-anginan!	Sangat Positif
Kebiasaan pake indosat ngecek pulsa salah,tapi makasih telkomsel,internet mu cepat sekaliii :*.	Sangat Positif
Sudah 2 bulan sejak Januari 2018 keluhan saya no tiket C19402752 tidak ada respon dan solusi @myXL @myXLCare Cc: @YLKI_ID	Netral
@Telkomsel sinyal jelek bgt sih, kecewa nih pakai simpati daerah buaran, serpong	Negatif
@triindonesia maju tak gentar membela yang benar, maju serentak menenangkan yang marah. #3BangkitIndonesia	Negatif
Membalas @Telkomsel Min mau nanya ni.. knp sim card telkomsel saya tdk bisa buat panggilan keluar sedangkan kemarin bisa . Sudah dua minggu koit min sdh berhasil registrasi pula gimana saran dan tanggapan anda mewakili telkomsel ?	Netral

Setelah Telkomsel... Ini Indosat memakai jasa Felix Kwetiau org HTI ormas terlarang utk isi acara Ramadhan... Emangnya gak ada kah Ulama yg lain? Mohon klarifikasinya utk @IndosatCare @IndosatBusiness Apa benar flyer yg saya terima ini? *D. ??	Netral
@undzhira @myXLCare @myXL coba cek ke *1,2,3*4*3# zhir... #bukananorangdalemXL	Netral
Alhamdulillah kartu indosat pintar laris manis..bisa free call 1000 menit 1000 sms ke semua indosat dan 512 data... <a href="http://fb.me/5UnVZP7qR">http://fb.me/5UnVZP7qR</a>	Sangat Positif
Sinyal xl kencang badai, Youtube pun lancar maksimal	Positif
Membalas @triindonesia saya sedang mempertanyakan informasi kuota yg berbeda antara bima dan web 3 saya pinta penjelasan	Netral
@triindonesia sinyal Tri di area Cikarang delta mas dan Jababeka hilang sinyal mulai jam 16.00 tadi Mohon perbaikannya. Terima kasih	Positif

Sekarang udah jamannya 4G LTE, udah ngga ada lagi yang namanya lemot. Kata temen 4G LTEnya Telkomsel dahsyat banget .. Eh pas dites, bener!	Negatif
makasih @indosatcare keluhanku sudah di tangani dengan cepat, i love indosat	Sangat Positif
Sekarang pake indosat pasca bayar, rada gimana gitu skrg sama indosat semenjak gw nangis di kantor indosat, gegara 9177 gw gak bisa balik	Negatif
Ogah RT @febrysara: beralihalah ke xl..sinyal kencang dan lari pulsanya juga kencang RT@veliieii Indosat kampring!!	Positif
di buat nangis sama jaringan indosat :/ gara* jaringan nya jelek nyangka nya mz dari dia ga di lz hampura sayank :* @egi_pribadi	Negatif
Membalas @adetruna nambahin wacana, selain live streaming di youtube... ngapain lagi ya biar sinyal kencang XL bisa dimaksimalkan? #XL4GLTEJatengDIY	Positif
Membalas @triindonesia Tolong perbaiki sinyal 3 di Aceh, khusus nya Banda Aceh.Sudah 4 hari kehilangan sinyal tanpa ada informasi apa pun dari triindonesia, kuota baru	Netral

diisi malah bikin kecewa dengan sinyal tiba-tiba menghilang gini. Registrasi kartu juga sudah dilakukan, tapi masih tak berguna kartunya	
(^^)/ keluarga telkomsel memang terbaik.. RT @MessyBae: @GalihSecos kayaknya aq jg akan kembali ke kartu AS qu...?	Sangat Positif
Membalas @Telkomsel Saya mau nangis kejer rasanya mas. Kalau saya main mobile legend tuh tiba tiba jaringan terputus tanpa sebab. Tau gak telkomsel sering banget kayak gini. Padahal untuk streaming apapun itu lancar lancar aja. Mas saya udah 2x turun credit score mobile legend gara2 koneksi terputus	Negatif
Kepada para pemenang Kuis #3BangkitIndonesia bisa dicek DM-nya ya, hadiah sudah mimin kirimkan :D	Sangat Positif
mantap..sinyal 3G stabil skrg dirumah...thx telkomsel...	Positif
@myXL kualitas sinyalnya makin hari makin ancur. tolong dong diperbaiki, hidup saya udah hancur masa sinyal saya mau diancurin juga? kalo pelanggan kecewa kaya gini wajar	Negatif

kan ya? emang ga cape liat timeline situ dipenuhi sama ocehan para konsumen dulu?	
#Im3 ne Iklan Sampah.. Kreatif dikit Napa.. Ikut2an As Telkomsel.. Maluuuuuu..	Sangat Negatif
Ready pisan :)) RT @triindonesia: Sudah siap untuk kuis #3BangkitIndonesia? Tes suara dulu ah ;)	Sangat Positif
Membalas @Telkomsel Kalo kayak gini gimana coba aktifinnyaaaaa !!! Kasih tau dong caranyaaaaa	Netral
Nah ini ! RT @triindonesia: Cari Tri Store terdekat di kota kamu? Klik link ini untuk informasi lengkapnya yaa :) <a href="http://bit.ly/1la0SLo">http://bit.ly/1la0SLo</a>	Netral
@mahendraekky_ @myXL @Telkomsel Pake antena PF Goceng brad, siapa tau tambah bagus sinyalnya... #saran	Netral
Membalas @iwnkurniawn @Telkomsel Klo bisa nangis, nangis hp sy di restart trus dr kmarin!	Negatif
Gw yakin @Telkomsel ga bisa baca komen2 ini.. Lemot.. #TelkomselProRadikalis #TelkomselProRadikalis #TelkomselProRadikalis #TelkomselProRadikalis #TelkomselProRadikalis #TelkomselProRadikalis	Negatif

<p>#TelkomselProRadikalis #TelkomselProRadikalisJoe® ? menambahkan, Richardo Tio @Richardotio Min @Telkomsel udah tahu belum hestek #TelkomselProRadikalis jadi trending topik? Makanya kalau memilih penceramah agama hati-hati min.. Jangan sampai gara-gara begini ada aksi boikot..</p>	
<p>monyet koneksi sampah, telkomsel nepu mulu dan gw mau aja ditipu terus. jadi yang bego siapa?</p>	<p>Sangat Negatif</p>
<p>Ir.Soekarno Hatta RT @triindonesia: Kuis #3BangkitIndonesia : Tebak tokoh yang ada di gambar ini! <a href="http://twitpic.com/9ifjt6">http://twitpic.com/9ifjt6</a></p>	<p>Sangat Positif</p>
<p>Membalas @Telkomsel @Telkomsel anyway, sangat puas dgn pelayanan Grapari Tsel Bogor cs Amel sabtu kmren. Thx</p>	<p>Sangat Positif</p>
<p>Pengalaman gue sangat mantap pakek indosat</p>	<p>Sangat Positif</p>
<p>@Telkomsel min saya mau nanya, ngga sengaja saya lewat jatuh tempo tagihan kartu halo, gimana ya mengurusnya? Apakah denda atau bagaimana?</p>	<p>Netral</p>



Kumpul mahadelta dapet rejeki dari indosat lumayan 11gb free lancar abis. Terima kasih indosat	Positif
Membalas @triindonesia Baik, Ada yang bisa kami bantu mengenai layanan Smartfrennya. Terima kasih :) – Randi	Sangat Positif
Joss @Othonk_00uye: Makany klo beli no sklian bli sinyal. Ahhaha RT @stevaniefebrian: Plis xl jgn balak2 donk! Knp skr signal nya ababil sih	Sangat Positif
@Telkomsel anjing lu!kalau gk suka mobile legend jangan dilenotin dong sinyalnya asu! AOV itu asu kyk telkomsel	Sangat Negatif
Udah senin nih. upload foto" pas weekend kmrn ayo pakai XL. Internet Super Cepat & Stabil #LovePalembang #Hotrod	Sangat Positif
Lagu favoritku Kecewa karna org itu sukanya bikin hati org kecewa baper bgt lagunya @triindonesia @kampungseleb #KONSERROSSA3	Negatif
Merah sinyalnya, jadi noob gua gila @triindonesia edan sinyalnya, pada molor apa operatornya tai lah	Sangat Negatif
nih provider @IndosatCare harus banyak belajar sama @Telkomsel keluhan cepat di tanggapi ,mungkin late	Negatif

respon terlalu banyak yg complain ke indosat ,bertahun2 pake im3 sama telkomsel baru indosat yg sering pulsa raib pake tuyul apa sih	
apakah ini kemajuan? apa indosat makin maju? dari kemarin-kemarin internetnya cepet terusssss selamat !	Sangat Positif
CS nya lelet dan lemot, @Telkomsel harus merubah sistem pelayanan, agar tidak ada korban lagi setelah saya. Kecewa dg peleyananya..	Negatif
SUMPAH XL @myXL SAMPAH BANGET SINYALNYA. BENERIN DONG JARINGANNYA JANGAN CUMA BERANI NGASIH PROMO MURAH.????????	Sangat Negatif
Indosat sinyal nya palang merah. Okeee. Xl lumayan lah ada dikit2...	Positif
@triindonesia min , ini jaringan di malang (lowokwaru ,dinoyo) lagi maintenance ya ? kok drtd jaringan data gamau konek kecewa banget ....	Negatif
Membalas @putrachndrg Terima kasih Indosat	Positif

Membalas @DanielMilagross @DanielMilagross @XL1,2,3 memang gitu sinyal XL terkadang kencang tiba2 lambat.Sama ditempat q jga gitu.tapi jgn ganti kartu semuanya Sama.	Positif
@triindonesia #089523359172 #Jl.PHH Musofa (Jln Suci dpn SMA YAS) "knpa ya skarang sinyal tri EDGE 3 terus ga pernah H di area rumah saya?"	Negatif
Sama! Tolong kenapa ini XL di rumah yang biasanya streaming ga buffer ini buka twitter aja gabisa. Hadeh @myXLCare @myXL Kile. menambahkan, ?nnk? ?nnis? k?nti @annkannisa udah seminggu belakangan ini sinyal bangke bgt deh xl. lte full tp gabisa akses inet. plis deh @myXLCare @myXL	Netral
Membalas @IndosatCare Anjing indosat goblok fakk maen moba aja leg nya minta di tampil kontrol	Sangat Negatif
Asu @indosat jaringane kek tai percuma pkt banyak tp nge lag gbsa ngapa2in sumpah payah jancuk	Sangat Negatif

Membalas @a_damnthing @InfoTwitwor dan 4 lainnya Iya nih kontrol indosat pulsa gua dipotong 50k anjing babi, pulsa cepe beli paket telpon 20 sisa 30 gajelas assu	Sangat Negatif
satelit telkomsel meledak pa gimana hng #SEHUNxLV #LVCruise @weareoneEXO	Netral
Pertama kalinya XL dibawa ke luar rumah. Signal internetnya bagus! bye bye Indosat. cukup lah ya bertahan 6 tahun. :)	Positif
Membalas @TsamaraDKI Semua program infrastruktur warisan dari pemerintahan sebelumnya... kita tunggu yg murni dari jokowi seperti listrik 35k megawatt, kereta cepat jkt bdg, buyback indosat, 10jt tenaga kerja	Negatif
Susah sama @Telkomsel smpai skrg cuma di tanya, di tanggapinya & perbaikannya ga ada padahal setia menggunakan tsel cuma skrg kecewa brt	Negatif
Kecewa sama pelayanan Telkomsel, harga Paket makin naik tapi koneksi abal.	Negatif
Membalas @myXL Makin lama makin jelek, paket dah mahal pula, apaan 4g kek gini	Negatif

Membalas @IndosatCare waktu itu kartu udah hangus karena lupa isi pulsa, alhamdulillah pas di urus ke gallery indosat pas hari itu juga kartu bisa di reaktifasi lagi meskipun harus nunggu 2 bulan, thanks indosat	Sangat Positif
@triindonesia harusnya lihat kualitasnya dulu baru naikkin harga,banyak org2 tidak nyaman dengan kebijakan ini	Negatif
@IndosatCare halo indosat, saya sudah kirim keluhan via DM. tolong cek dan follow up, ya. mohon bantuannya. terima kasih.	Netral
Membalas @myXL @XLCare ini xl knapa sih? gangguan mulu signal ada tapi no internet connection.. sampah	Sangat Negatif
Oke thanks info.nya RT @triindonesia: @AnaasKhoironi informasi paket sms bisa km cek di <a href="http://bit.ly/1mbtTqC">http://bit.ly/1mbtTqC</a>	Netral
Membalas @myXL Saya tidak pilih semuanya. Karena saat ini jaringan internet XL mbyarpet alis MATI-HIDUP plus LELET. Sangat..sangat..sangat mengecewakan. Kecewa karena sudah terjadi sejak Sabtu kemarin, dari kampung saya Pasuruan, sampe saya balik ke Bali hingga DETIK INI. PARAH!!!!	Negatif

