ITS
Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR - IS184853

ANALISIS SENTIMEN TEKS BAHASA INDONESIA PADA MEDIA SOSIAL MENGGUNAKAN LONG SHORT TERM MEMORY DAN CONVOLUTIONAL NEURAL NETWORK (STUDI KASUS: OPERATOR TELEKOMUNIKASI)

SENTIMENT ANALYSIS INDONESIAN LANGUAGE IN SOCIAL MEDIA WITH LONG SHORT TERM MEMORY AND CONVOLUTIONAL NEURAL NETWORK (CASE STUDY : TELECOMMUNICATION OPERATOR)

EVIA NANDA IRAWATI
NRP 05211540000056

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

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

TUGAS AKHIR - IS184853

ANALISIS SENTIMEN TEKS BAHASA INDONESIA PADA MEDIA SOSIAL MENGGUNAKAN LONG SHORT TERM MEMORY DAN CONVOLUTIONAL NEURAL NETWORK

(STUDI KASUS : OPERATOR TELEKOMUNIKASI)

EVIA NANDA IRAWATI
NRP 05211540000056

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

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

UNDERGRADUATE THESIS - IS184853

SENTIMENT ANALYSIS INDONESIAN LANGUAGE IN SOCIAL MEDIA WITH LONG SHORT TERM MEMORY AND CONVOLUTIONAL NEURAL NETWORK

(CASE STUDY : TELECOMMUNICATION OPERATOR)

EVIA NANDA IRAWATI
NRP 05211540000056

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

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

LEMBAR PENGESAHAN

ANALISIS SENTIMEN TEKS BAHASA INDONESIA PADA MEDIA SOSIAL MENGGUNAKAN LONG SHORT TERM MEMORY DAN CONVOLUTIONAL NEURAL NETWORK

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:

EVIA NANDA IRAWATI
NRP. 05211540000056

Surabaya, 9 Juli 2019

KEPALA
DEPARTEMEN SISTEM INFORMASI

Mahendrawathi ER. S.T., M.Sc., Ph.D
NIP. 197610112006042001

LEMBAR PERSETUJUAN

ANALISIS SENTIMEN TEKS BAHASA INDONESIA PADA MEDIA SOSIAL MENGGUNAKAN LONG SHORT TERM MEMORY DAN CONVOLUTIONAL NEURAL NETWORK

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:

EVIA NANDA IRAWATI
NRP. 05211540000056

Disetujui Tim Penguji :Tanggal Ujian : 9 Juli 2019
Periode Wisuda : September 2019

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

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

Faizal Johan Atletiko, S.Kom, M.T **(Penguji II)**

ANALISIS SENTIMEN TEKS BAHASA INDONESIA PADA MEDIA SOSIAL MENGGUNAKAN LONG SHORT TERM MEMORY DAN CONVOLUTIONAL NEURAL NETWORK

Nama Mahasiswa : Evia Nanda Irawati
NRP : 05211540000056
Departemen : Sistem Informasi FTIK-ITS
Pembimbing I : Renny Pradina K., S.T, M.T., SCJP

ABSTRAK

Analisis sentimen yang menggunakan algortima neural network mencapai hasil yang sangat baik dalam pemrosesan Bahasa alami dan menunjukkan kemampuan belajar yang baik. Kata dan frasa direpresentasikan menggunakan distributed representation, sehingga memperoleh karakteristik hubungan antara kata sebelum dan sesudah bahasa dan menghindari masalah dimensionalitas tinggi dan penyebaran data. Seiring dengan berjalan waktu, neural network semakin banyak mengalami perkembangan dan menghasilkan banyak macam arsitektur. Pada penelitian ini akan menggunakan Long Short Term Memory dan Convolutional Neural Network untuk menganalisis sentimen publik terhadap layanan operator telekomunikasi di Indonesia yang dimuat di media sosial.

Penggunaan algoritma Long Short Term Memory dan Convolutional Neural Network melalui beberapa tahapan. Diawali dengan membagi dataset kedalam tiga subtask A, B dan C. Subtask A merupakan pengelompokan dataset yang memiliki dua macam label yaitu sentimen positif dan negatif. Kemudian untuk subtask B merupakan pengelompokan dataset yang memiliki tiga macam label yaitu positif, negatif dan netral sedangkan subtask C memiliki 5 macam label yaitu sangat positif, positif, sangat negatif, negatif dan netral.

Selanjutnya dilakukan pengujian tiap subtask terhadap algoritma Long Short Term Memory dan Convolutional Neural Network yang sebelumnya diberi bobot awal dengan model

Word2Vec dengan learning algorithm Skip-gram. Hasil dari pengujian tersebut yang memiliki performa terbaik menjadi masukan pada tahapan menggabungkan model Long Short Term Memory dan Convolutional Neural Network.

Long Short Term Memory menggunakan 2 macam model yaitu bidirectional dan yang non-bidirectional atau lebih disebut dengan LSTM saja. Perbandingan hasil evaluasi pengukuran pada 3 subtask menunjukkan nilai performa model Bidirectional LSTM lebih baik pada 2 subtask yaitu A dan B dibandingkan dengan model LSTM. Proses pelatihan model Bidirectional LSTM pada 3 subtask berbeda didapatkan nilai optimizer paling baik adalah Adadelta dengan nilai learning rate 1. Sedangkan untuk model LSTM pada subtask A dan C didapatkan nilai optimizer paling baik adalah Adadelta dengan nilai learning rate 1, pda subtask B nilai optimizer paling baik adalah Adam dengan learning rate 0.001.

Kemudian untuk model Convolutional Neural Network yang menggunakan multi filter region size performanya mengungguli Long Short Term Memory pada semua subtask. Penggabungan model Convolutional Neural Network dan Long Short Term Memory tidak dapat mengungguli performa model individu dalam melakukan proses klasifikasi.

Kata kunci: Analisis sentimen, Media Sosial, Long Short Term Memory, Convolutional Neural Network.

SENTIMENT ANALYSIS INDONESIAN LANGUAGE IN SOCIAL MEDIA WITH LONG SHORT TERM MEMORY AND CONVOLUTIONAL NEURAL NETWORK

Name : Evia Nanda Irawati
NRP : 05211540000056
Department : Information System FTIK-ITS
Supervisor : Renny Pradina K., S.T, M.T., SCJP

ABSTRACT

Sentiment analysis using neural network algorithms achieves excellent results in natural language processing and shows good learning skills. Words and phrases are represented using distributed representations, thus obtaining the characteristics of the relationship between the words before and after language and avoiding high dimensionality problems and data dissemination. As time goes on, more and more neural networks experience development and produce many kinds of architecture. In this study, we will use Long Short Term Memory and Convolutional Neural Network to analyze public sentiment towards telecommunication operator services in Indonesia that are published on social media.

The use of Long Short Term Memory and Convolutional Neural Network algorithms through several stages. Beginning by dividing the dataset into three subtasks A, B and C. Subtask A is a grouping of datasets that have two kinds of labels, namely positive and negative sentiments. Then for subtask B is a grouping of datasets that have three kinds of labels, namely positive, negative and neutral while subtask C has 5 kinds of labels, namely very positive, positive, very negative, negative and neutral.

Then testing each subtask on the Long Short Term Memory algorithm and Convolutional Neural Network algorithm which was previously given the initial weight with the Word2Vec model with the Skip-gram learning algorithm. The results of

these tests which have the best performance are input to the stages of combining the Long Short Term Memory and Convolutional Neural Network models.

Long Short Term Memory uses 2 types of models, namely bidirectional and non-bidirectional or more so-called LSTM. Comparison of the results of evaluation of measurements in 3 subtasks shows the performance value of the Bidirectional LSTM model is better in 2 subtasks, namely A and B compared to the LSTM model. The training process of the Bidirectional LSTM model in 3 different subtasks found that the best optimizer value was Adadelta with a learning rate 1. As for the LSTM model in the A and C subtask, the best optimizer value was Adadelta with the learning rate 1, in the B subtask the optimizer value was good is Adam with a learning rate of 0.001.

Then for the Convolutional Neural Network model that uses a multi filter region the size performance outperforms the Long Short Term Memory in all subtasks. Combining the Convolutional Neural Network and Long Short Term Memory models cannot outperform the individual models in carrying out the classification process.

Keywords: Analisis sentimen, Media Sosial, Long Short Term Memory, Convolutional Neural Network.

KATA PENGANTAR

Dengan mengucapkan rasa syukur kepada Tuhan Yang Maha Pengasih dan Maha Penyayang atas izin-Nya dan selalu memberi petunjuk dan memudahkan urusan-urusan penulis dalam segala aspek kehidupan penulis dapat menyelesaikan buku ini dengan judul Analisis Sentimen Teks Bahasa Indonesia pada Media Sosial Menggunakan Long Short Term Memory dan Convolutional Neural Network. Dalam penyelesaian Tugas Akhir ini, penulis diiringi oleh pihak-pihak yang selalu memberi dukungan, saran, dan doa sehingga penelitian berlangsung dengan lancar. Secara khusus penulis mengucapkan terima kasih dari lubuk hati terdalam kepada:

1. Bapak Martono dan Ibu Linarsih selaku orang tua penulis yang telah memberi doa dan dukungan dalam banyak aspek, serta kedua saudara penulis yaitu Indra Bagus Prastian dan Eva Calista Putri yang selalu menyemangati penulis.
2. Ibu Renny Pradina K, S.T, M.T. selaku dosen pembimbing yang senantiasa memberikan ilmu, arahan dan motivasi selama mengerjakan penelitian ini.
3. Ibu Nur Aini Rakhmawati, S.Kom, M.Sc.Eng, Ph.D dan Bapak Faizal Johan Atletiko, S.Kom, M.T selaku dosen penguji yang telah memberikan kritik dan saran yang membuat kualitas penelitian ini lebih baik lagi.
4. Segenap dosen Departemen Sistem Informasi yang telah memberi bekal ilmu kepada penulis selama berada Departemen Sistem Informasi.
5. Anggota Laboraturium ADDI yang telah yang menjadikan labrotarium menjadi tempat yang hangat dan menyenangkan untuk mengerjakan penelitian.
6. Untuk Andira, Weny, Supri, Azam, Yoga dan anak bimbing Ibu Renny Pradina K, S.T, M.T lainnya yang menjadi wadah bertukar pikiran dan saling memberi semangat dalam mengerjakan penelitian ini.

7. Regina Cindy, Magrid dan Fia yang menjadi teman berkeluh kesah selama pengerjaan penelitian ini.
8. Ramdhan Febrianto Saputra selaku teman dekat penulis yang senantiasa bersedia membagikan pengalaman dan memberikan semangat dalam pengerjaan tugas akhir ini
9. Adrian dan Alden selaku penulis penelitian terkait yang telah menyediakan waktu untuk memberikan informasi mengenai penelitian ini
10. Teman-teman Lannister 2015 yang telah memberikan warna dan cerita dalam kehidupan kampus penulis
11. Teman - teman Alumni SMA Negeri 1 Kediri yang juga merupakan Lannister yang kiprahnya saat perkuliahan sangat mengispirasi, Salim dan Rahmadani
12. Teman-teman se-karesidenan Kediri yang juga menginspirasi
13. Pihak lainnya yang berkontribusi dalam tugas akhir yang belum dapat penulis sebutkan satu per satu.

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

Surabaya, 4 Juli 2019

Penulis

DAFTAR ISI

LEMBAR PENGESAHAN i
LEMBAR PERSETUJUAN iii
ABSTRAK v
ABSTRACT vii
KATA PENGANTAR ix
DAFTAR ISI xi
DAFTAR GAMBAR xv
DAFTAR TABEL xvii
DAFTAR KODE xix
1 BAB I PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 3
1.3 Batasan Permasalahan 3
1.4 Tujuan 4
1.5 Manfaat 4
1.6 Relevansi 4
2 BAB II TINJAUAN PUSTAKA 5
2.1 Penelitian Sebelumnya 5
2.2 Dasar Teori 8
2.2.1 Analisis Sentimen 8
2.2.2 Machine Learning 8
2.2.3 Natural Language Processing 8
2.2.4 Word Embedding 9
2.2.5 Word2Vec 9
2.2.6 Media Sosial 9
2.2.7 Twitter 10
2.2.8 Crawling 10
2.2.9 Long Short Term Memory 10
2.2.10 Convolutional Neural Network 12
3 BAB III METODOLOGI 17
3.1 Diagram Metodologi 17
3.2 Arsitektur Penelitian 18
3.3 Uraian Metodologi 18
3.2.1 Identifikasi Masalah dan Studi Literatur 18

3.2.2 Pengumpulan Data 19
3.2.3 Pra-pemrosesan Data 19
3.2.4 Training Data 19
3.2.5 Analisis Model 19
3.2.6 Dokumentasi 20
4 BAB IV PERANCANGAN 21
4.1 Perancangan Pengumpulan Data 21
4.2 Perancangan Pra-Pemrosesan Data 22
4.2.1. Perancangan Perubahan URL dan Mention menjadi Token 23
4.2.2. Perancangan Penghapusan Huruf yang Berulang 23
4.2.3. Perancangan Perubahan Emoticon menjadi Token 23
4.2.4. Penghapusan Penghapusan Tanda Baca, Simbol dan Angka 24
4.3 Perancangan Word Embedding 24
4.4 Perancangan Model Convolutional Neural Network ... 25
4.5 Perancangan Model Long Short Term Memory 28
4.6 Perancangan Evaluasi Ensembel Model 30
5 BAB V IMPLEMENTASI 33
5.1 Lingkungan Implementasi 33
5.2 Pra-Pemrosesan Data 34
5.3 Pemuatan Word Embedding 41
5.4 Pembuatan Model Convolutional Neural Network 43
5.5 Pembuatan Model Long Short Term Memory 47
5.6 Pembuatan Ensembel Model 51
6 BAB VI HASIL DAN PEMBAHASAN 63
6.1 Hasil Pengujian Model CNN 63
6.1.1 Konfigurasi Awal Parameter CNN 63
6.1.2 Subtask A 63
6.1.3 Subtask B 65
6.1.4 Subtask C 66
6.2 Hasil Pengujian Model LSTM 67
6.2.1 Subtask A 68
6.2.2 Subtask B 72
6.2.3 Subtask C 78

6.3 Hasil Pengujian Model Ensembel 82
6.3.1 Subtask A 82
6.3.2 Subtask B..... 83
6.3.3 Subtask C..... 83
6.4 Perbandingan Kecepatan Pemrosesan CNN dan LSTM 84
6.5 Hasil Klasifikasi Sentimen tiap Operator Telekomunikasi 84
7 BAB VII KESIMPULAN DAN SARAN..... 91
7.1 Kesimpulan..... 91
7.2 Saran 91
8 DAFTAR PUSTAKA 93
LAMPIRAN A..... 96
9 BIODATA PENULIS 101

Halaman ini sengaja dikosongkan

DAFTAR GAMBAR

Gambar 2.1 Struktur LSTM oleh Jiarui Zhang [15] 11
Gambar 2.2 Arsiterktur Bi-LSTM[7] 12
Gambar 2.3 Proses Konvolusi[19] 13
Gambar 2.4 Proses *Max-Pooling*[19] 14
Gambar 2.5 Grafik ReLu[20] 14
Gambar 3.1 Diagram Metodologi..... 17
Gambar 3.2 Arsitektur Penelitian 18
Gambar 4.1 Proses Word Embedding 24
Gambar 4.2 Arsitektur CNN 26
Gambar 4.3 Arsitektur LSTM 29
Gambar 4.4 Proses Ensembel Model..... 31
Gambar 4.5 Metode *Soft Voting* 31
Gambar 6.1 Performa LSTM dan Bi-LSTM Subtask A 69
Gambar 6.2 Confusion Matrix LSTM Subtask A 70
Gambar 6.3 Normalized Confusion Matrix LSTM Subtask A 70
Gambar 6.4 Confusion Matrix Bi-LSTM Subtask A 71
Gambar 6.5 Normalized Confusion Matrix LSTM Subtask A 72
Gambar 6.6 Performa LSTM dan Bi-LSTM Subtask B 74
Gambar 6.7 Confusion Matrix LSTM Subtask B 75
Gambar 6.8 Normalized Confusion Matrix LSTM Subtask B76
Gambar 6.9 Confusion Matrix Bi-LSTM Subtask B 77
Gambar 6.10 Normalized Confusion MatrixBi- LSTM Subtask B..... 77
Gambar 6.11 Performa LSTM dan Bi-LSTM Subtask C 79
Gambar 6.12 Confusion Matrix LSTM Subtask C 80
Gambar 6.13 Normalized Confusion Matrix LSTM Subtask C 80
Gambar 6.14 Confusion Matrix Bi-LSTM Subtask C 81
Gambar 6.15 Normalized Confusion Matrix Bi-LSTM Subtask C..... 82
Gambar 6.16 Hasil Klasifikasi Sentimen Tiap Operator Telekomunikasi 86
Gambar 6.17 Confusion Matrix Best Model Subtask C 87

Halaman ini sengaja dikosongkan

DAFTAR TABEL

Tabel 2.1 Literatur 5
Tabel 4.1 Distribusi Label Dataset dan Sampel Tweet 21
Tabel 4.2 Deskripsi Tiap Subtask 22
Tabel 4.3 Parameter Model CNN 27
Tabel 4.4 Parameter Model LSTM 29
Tabel 5.1 Spesifikasi Komputer 33
Tabel 5.2 Perangkat Lunak yang Dipakai 33
Tabel 6.1 Konfigurasi Awal Parameter CNN 63
Tabel 6.2 Hasil Pengujian Single Filter Region Subtask A ... 64
Tabel 6.3 Hasil Pengujian Multi Filter Region Subtask A 64
Tabel 6.4 Hasil Pengujian Single Filter Region Subtask B ... 65
Tabel 6.5 Hasil Pengujian Multi Filter Region Subtask B 66
Tabel 6.6 Hasil Pengujian Single Filter Region Subtask C ... 66
Tabel 6.7 Hasil Pengujian Multi Filter Region Subtask C 67
Tabel 6.8 Konfigurasi Parameter Awal LSTM 67
Tabel 6.9 Percobaan pada Model LSTM 68
Tabel 6.10 Percobaan pada Model Bi-LSTM 68
Tabel 6.11 Percobaan pada Model LSTM Subtask B 72
Tabel 6.12 Percobaan pada Model Bi-LSTM Subtask B 73
Tabel 6.13 Percobaan pada Model LSTM 78
Tabel 6.14 Percobaan pada Model Bi-LSTM 78
Tabel 6.15 Pengujian Ensembel Subtask A 82
Tabel 6.16 Pengujian Ensembel Subtask B 83
Tabel 6.17 Pengujian Ensembel Subtask C 83
Tabel 6.18 Perbandingan Rata-rata Kecepatan Pemrosesan CNN, LSTM dan Bi-LSTM 84
Tabel 6.19 Sampel Tweet yang Tidak Akurat 87

Halaman ini sengaja dikosongkan

DAFTAR KODE

Kode 5.1 Proses Memanggil Dataset 35
Kode 5.2 Proses Membersihkan Dataset 36
Kode 5. 3 Proses Perubahan Emoticon menjadi String..... 37
Kode 5.4 Proses Perubahan Emoticon menjadi String..... 38
Kode 5.5 Proses Menghapus Simbol dan Filter Huruf 38
Kode 5. 6 Eksekutor Fungsi Pembersihan Dataset 40
Kode 5.7 Konversi Label Numerik ke Kata 40
Kode 5.8 Proses Membaca Model 41
Kode 5. 9 Pemberian Bobot oleh Word Embedding..... 42
Kode 5.10 Parameter Model CNN 44
Kode 5.11 Pendefinisian Model CNN 45
Kode 5.12 Pembuatan Model CNN 46
Kode 5.13 Parameter Model LSTM 48
Kode 5. 14 Pendefinisian Model LSTM 48
Kode 5.15 Pembuatan Model LSTM 50
Kode 5. 16 Fungsi *Cross Validaton* 52
Kode 5.17 Fungsi Lanjutan *Cross Validation* 53
Kode 5.18 Fungsi Lanjutan *Cross Validation* 54
Kode 5.19 Instantiasi Kelas Model CNN dan LSTM 55
Kode 5.20 Proses Training Model 56
Kode 5.21 Proses Training Model Lanjutan 57
Kode 5. 22 Pemanggilan Fungsi Testing dan Statistik 58
Kode 5.23 Proses Testing Model 59
Kode 5.24 Proses Ansembel Model 60

Halaman ini sengaja dikosongkan

BAB I
PENDAHULUAN

Pada bab ini, akan dijelaskan mengenai proses identifikasi masalah meliputi latar belakang masalah, rumusan masalah, batasan masalah, tujuan, manfaat dan relevansi tugas akhir. Berdasarkan uraian pada bab ini diharapkan gambaran umum permasalahan dan pemecahan masalah masalah pada penelitian ini dapat dipahami.

1.1 Latar Belakang

Pengguna internet di Indonesia dari tahun ke tahun meningkat secara pesat. Jumlah pengguna internet di Indonesia pada tahun 1998 baru mencapai 500 ribu jiwa, namun pada tahun 2017 telah menembus angka 143,26 juta jiwa [1]. Perkembangan yang cukup signifikan ini didukung oleh kepemilikan perangkat komputer/laptop sebesar 25,72% dan kepemilikan smartphone/tablet sebesar 50,08% [1]. Pemanfaatan internet oleh masyarakat Indonesia cukup beragam, aplikasi chatting adalah jenis layanan yang paling banyak diakses dengan persentase 89,35% kemudian disusul oleh media sosial sebesar 87,13% [1]. Salah satu media sosial yang banyak digunakan oleh masyarakat Indonesia adalah *Twitter*. Hingga kuartal I – 2017, dari 328 juta pengguna twitter di seluruh dunia, Indonesia masuk lima besar pengguna *twitter*. Sepanjang tahun 2016 sebanyak 4,1 miliar tweet dihasilkan oleh pengguna dari Indonesia, berarti rata-rata tweet yang dihasilkan tiap hari sekitar 11 juta tweet [2].

Waktu yang banyak dihabiskan di media sosial ini menciptakan data yang sangat besar, hal ini tentu dapat menjadikan media sosial sebagai sumber informasi baru. Informasi yang dapat ditangkap melalui sosial media salah satunya adalah pendapat dan sentimen masyarakat. Bahkan dalam bidang bisnis,

mengetahui sentimen pelanggan terhadap barang dan jasa yang ditawarkan sangat memberi dampak yang besar bagi keberlanjutan bisnis. Pengetahuan tentang sentimen terhadap bisnis dapat memberikan masukan untuk meningkatkan kualitas produk, ukuran keberhasilan suatu kampanye marketing, meningkatkan layanan pelanggan dan salah satu sumber ide menciptakan konten baru yang menarik minat pelanggan [3]. Hal tersebut dapat dilakukan dengan mengklasifikasikan teks yang ada dalam sebuah kalimat atau dokumen, kemudian dari teks yang dikelompokkan akan dihitung kecenderungan opini yang disampaikan dalam dokumen tersebut apakah positif, negatif atau netral. Salah satu pengerjaan klasifikasi teks dapat dilakukan dengan menggunakan *machine learning* [4].

Machine learning adalah penerapan kecerdasan buatan yang memberikan kemampuan pada sistem untuk secara otomatis belajar dan meningkatkan dari pengalaman tanpa diprogram secara eksplisit. Pembelajaran mesin berfokus pada pengembangan program komputer yang dapat mengakses data dan menggunakannya untuk belajar sendiri [5].

Teks klasifikasi menggunakan machine learning diantaranya adalah *neural network*. *Neural network* hasil mencapai hasil yang sangat baik dalam pemrosesan Bahasa alami dan menunjukkan kemampuan belajar yang baik. Kata dan frasa direpresentasikan menggunakan *distributed representation*, sehingga memperoleh karakteristik hubungan antara kata sebelum dan sesudah bahasa dan menghindari masalah dimensi tinggi dan penyebaran data [6]. Seiring dengan berjalan waktu, *neural network* semakin banyak mengalami perkembangan dan menghasilkan banyak macam arsitektur. Penelitian yang dikirim ke tantangan analisis sentimen SemEval yang mengkombinasikan dua macam *neural network* yaitu *Convolutional Neural Network* (CNN) dan *Long Sort Term*

Memory dapat mengungguli metode lain pada task analisis sentimen [7].

Berdasarkan permasalahan yang ada serta studi literatur yang telah dilakukan, maka penelitian ini bermaksud untuk melakukan analisis sentimen teks Bahasa Indonesia pada media sosial menggunakan algoritma Convolutional Neural Network dan Long Short Term Memory.

1.2 Rumusan Masalah

Berdasarkan uraian latar belakang diatas, maka perumusan permasalahan yang menjadi fokus dan akan diselesaikan dalam tugas akhir ini adalah sebagai berikut:

1. Bagaimana implementasi algoritma Long Short Term Memory dan Convolutional Neural Network untuk analisis sentimen?.
2. Bagaimana cara mengukur performa analisis sentimen menggunakan algoritma Long Short Term Memory dan Convolutional Neural Network?
3. Bagaimana mengoptimalkan performa model algoritma Long Short Term Memory dan Convolutional Neural Network?

1.3 Batasan Permasalahan

Berikut ini Batasan masalah tugas akhir ini, berdasarkan latar belakang dan perumusan masalah yang telah dijelaskan,

1. Sumber data berasal dari Twitter dengan bahasa Indonesia
2. Dataset yang digunakan merupakan dataset yang berlabel tunggal yang merujuk pada topik Operator Telekomunikasi
3. Pengelompokan sentimen dibagi menjadi tiga subtask 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

1.4 Tujuan

Berdasarkan perumusan dan Batasan masalah yang telah diuraikan sebelumnya, maka tujuan yang ingin dicapai adalah:

1. Mengimplementasikan algoritma Long Short Term Memory dan Convolutional Neural Network untuk analisis sentimen
2. Melakukan uji coba untuk melihat performa analisis sentimen menggunakan algoritma Long Short Term Memory dan Convolutional Neural Network

1.5 Manfaat

1. Bagi penulis, memahami implementasi algoritma Long Short Term Memory dan Convolutional Neural Network untuk analisis sentimen
2. Bagi masyarakat, dapat dijadikan rujukan untuk penelitian-penelitian selanjutnya dan dapat dikembangkan dalam beberapa bentuk seperti bisnis maupun rancang bangun aplikasi yang memanfaatkan algoritma Long Short Term Memory dan Convolutional Neural Network

1.6 Relevansi

Tugas akhir ini dikatakan layak menjadi TA dari penulis karena hasil dari TA ini menerapkan mata kuliah bidang minat laboraturium Akuisisi Data dan Diseminasi Informasi (ADDI). Mata kuliah yang digunakan dalam mengembangkan TA ini adalah Pengolahan Bahasa Alami yang membahas tentang pemrosesan otomatis bahasa alami manusia dengan komputer. Selain itu TA ini juga berkaitan dengan mata kuliah Sistem Cerdas dimana memanfaatkan mesin pembelajaran untuk menyelesaikan berbagai tugas. Sehingga dapat dikatakan bahwa penelitian ini telah mempunyai relevansi sesuai dengan roadmap dari laboraturium Akuisisi Data dan Diseminasi Informasi, Departemen Sistem Informasi.

BAB II
TINJAUAN PUSTAKA

Bab ini menjelaskan mengenai penelitian sebelumnya dan dasar teori yang menjadi acuan dalam pengerjaan tugas akhir ini. Penelitian sebelumnya memberikan gambaran mengenai apa saja yang telah dilakukan dan dasar teori memberikan gambaran secara umum dari tugas ini.

2.1 Penelitian Sebelumnya

Pada penelitian ini, digunakan beberapa penelitian terdahulu yang berguna sebagai pedoman dan referensi dalam melaksanakan proses-proses dalam penelitian, seperti yang terdapat pada Tabel 2.1. Informasi yang disampaikan berisi tentang penelitian sebelumnya, hasil penelitian, dan hubungan penelitian yang akan dilakukan terhadap penelitian sebelumnya dalam rangka tugas akhir ini.

Tabel 2.1 Literatur

Penelitian 1	
Judul Penelitian	SemEval-2017 Task 4: Twitter Sentiment Analysis with CNN's and LSTMs
Nama Peneliti, Tahun Penilitian	Mathieu Cliché, 2017
Metodologi	- *Long Short Term Memory* - *Convolutional Neural Networks*
Relevansi Penelitian	Penelitian ini membahas pembuatan model analisis sentimen menggunakan *Convolutional Neural Network* bersama dengan *Long Short Term Memory*. LSTM yang digunakan adalah LSTM

	bidirectional yang mampu mempertahankan informasi kata kedepan maupun kebelakang sehingga bisa mengatasi kekurangan LSTM *unidirectional* . Kemudian menggunakan metode *distant training* pada CNN dimana embedding di *unfreeze* untuk memisahkan dan memberikan jarak antar kata yang memiliki polaritas sentimen yang berbeda.
Penelitian 2	
Judul Penelitian	Convolutional Neural Networks for Sentence Classification
Nama Peneliti, Tahun Penilitian	Yoon Kim, 2014
Metodologi	*Convolutional Neural Networks*
Relevansi Penelitian	CNN mengklasifikasikan teks dengan baik berbagai variasi, mulai dari CNN *random* yang menginisialisasi kata secara acak, CNN *static* yang memroses dengan word2vec terlebih dahulu, CNN *non static* sama dengan *static* tapi dengan *fine tune*, CNN *multichannel* yang membagi vector kata menjadi 2 set dan

	menjadikan channel, satunya fine tune yang lain dibuat static.
Penelitian 3	
Judul Penelitian	LSTM-CNN Hybrid Model for Text Classification
Nama Peneliti, Tahun Penilitian	J. Zhang, Y. Li, J. Tian, and T. Li, 2018
Metodologi	- *Long Short Term Memory* - *Convolutional Neural Networks*
Relevansi Penelitian	LSTM secara efektif dapat mempertahankan karakteristik informasi historis dalam urutan teks yang panjang dan mengekstraksi fitur teks lokal dengan menggunakan struktur CNN. Vektor keluaran dari model multi layer LSTM sebagai vektor masukan CNN, dan model CNN diatas multi layer LSTM untuk ekstrak teks *sequence* masukan. Hasil dari penelitian ini menunjukkan bahwa LSTM dengan *max pooling* akurasinya lebih tinggi daripada *avg pooling*. Model

	LSTM-CNN hasilnya lebih baik daripda model CNN dan model LSTM. Penggunaan filter *convolution* dengan ukuran berbeda juga berpengaruh dalam meningkatkan akurasi.

2.2 Dasar Teori

2.2.1 Analisis Sentimen

Analisis sentimen atau opinion mining merupakan sebuah alat untuk mendefinisikan sebuah informasi dari sebuah teks, seperti pendapat dan sentimen yang bertujuan untuk menciptakan sebuah pengetahuan yang dapat digunakan untuk sistem pendukung keputusan [8]. Analisis sentimen memiliki fungsi dasar yaitu mengelompokkan teks yang ada dalam sebuah kalimat atau dokumen, kemudian dari teks yang dikelompokkan akan dihitung kecenderungan opini yang disampaikan dalam dokumen tersebut apakah positif, negatif atau netral.

2.2.2 Machine Learning

Machine learning adalah penerapan kecerdasan buatan yang memberikan kemampuan pada sistem untuk secara otomatis belajar dan meningkatkan dari pengalaman tanpa diprogram secara eksplisit. Pembelajaran mesin berfokus pada pengembangan program komputer yang dapat mengakses data dan menggunakannya untuk belajar sendiri [5].

2.2.3 Natural Language Processing

Natural Language Processing adalah bagian dari kecerdasan buatan dan linguistik, dikhususkan untuk membuat komputer mengerti pernyataan dari kata-kata yang ditulis dalam bahasa manusia. Natural language atau bahasa alami yang dimaksud yaitu bahasa yang diucapkan dan ditulis oleh manusia dengan tujuan berkomunikasi pada umumnya[9]. Ada banyak sekali bahasa yang dituturkan manusia seperti Bahasa

Indonesia, Inggris, China maupun Jepang, hal yang sulit dilakukan apabila menginginkan komputer mengerti satu persatu bahasa secara spesifik. Ada dua tugas yang bisa diselesaikan Natural Language Processing, yang pertama adalah Natural Language Understanding untuk mengerti dan menjustifikasi masukan berupa bahasa alami. Yang kedua adalah Natural Language Generation untuk menggenerasi keluaran berupa teks [9]. Natural Language Processing diaplikasikan dalam mesin penerjemah, pengecek ejaan dan tata bahasa, ekstraksi informasi, peringkasan, tanya jawab, sistem dialog dan masih banyak lagi[10].

2.2.4 Word Embedding

Word Embedding merupakan sebutan yang merujuk pada representasi kata untuk menangkap konteks sebuah kata dalam suatu dokumen, kesamaan semantik dan sintak, hubungan dengan kata-kata yang lain dan lain sebagainya. Word embedding juga disebut Distributed Representation. Distributed Representation bersifat rapat, berdimensi rendah, dan bernilai riil[11]. Pada penelitian ini digunakan word embedding yang terbaru dan memiliki performa sangat bagus yaitu Word2Vec.

2.2.5 Word2Vec

Word2Vec adalah salah satu *word embedding* yang dapat merepresentasikan kata-kata kedalam vektor yang rapat dan berdimensi rendah. Word embedding yang dibuat oleh Thomas Mikolov ini memiliki dua model yaitu *Continuous Bag-of-Words* dan *Continuous Skip-gram*[12]. Keunggulan yang dimiliki oleh word embedding ini adalah kecepatan dalam proses training, efisien dan menangani dataset dengan skala besar.

2.2.6 Media Sosial

Media sosial merupakan sebuah media yang digunakan untuk bersosialisasi secara online yang memungkinkan manusia berinteraksi tanpa batas waktu dan ruang. Menurut M. Terry definisi media sosial adalah suatu media komunikasi dimana pengguna dapat mengisi kontennya secara bersama dan menggunakan teknologi penyiaran berbasis internet yang berbeda dari media cetak dan media siaran tradisional. Media sosial memungkinkan penggunanya untuk mengunggah konten

sesuai dengan preferensi dan minatnya kemudian dapat dibaca dan ditanggapi oleh banyak orang. Keterhubungan antar pengguna ini membuat media sosial memiliki proses pengiriman informasi yang sangat cepat.

2.2.7 Twitter

Twitter adalah suatu situs web layanan jaringan sosial dan mikroblog yang memberikan fasilitas bagi pengguna untuk mengirimkan "pembaharuan" berupa tulisan teks dengan jumlah karakter yang dibatasi yaitu sebanyak 140 karakter. Twitter didirikan pada Maret tahun 2006 oleh perusahaan rintisan Obvious Corp. Twitter dapat menjadi sumber yang sangat bermanfaat untuk mengumpulkan data yang digunakan dalam penelusuran informasi yang berkembang. Keragaman dari latar belakang pengguna Twitter membuat bahan informasi tersebut memiliki nilai lebih. Sepanjang tahun 2016, rata-rata didapatkan 11 juta tweet dari masyarakat Indonesia per hari yang membuat Twitter mampu menjadi sumber informasi yang tepat untuk menganalisa informasi dan sentiment Bahasa Indonesia di media sosial [13].

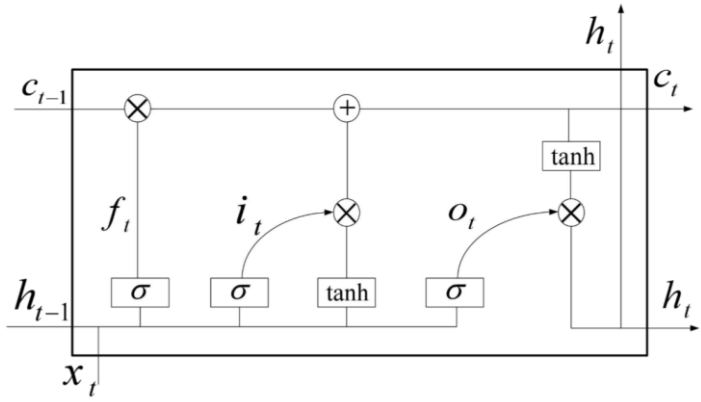
2.2.8 Crawling

Crawling adalah sebuah proses dimana informasi tertentu dari sebuah halaman dari sebuah website didapatkan kemudian disimpan secara offline [14]. Informasi yang didapatkan kemudian disiapkan agar bisa diproses oleh program lain. Dalam penelitian ini dilakukan crawling pada halaman media sosial Twitter untuk mendapatkan data tweet.

2.2.9 Long Short Term Memory

Long Short-Term Memory (LSTM) adalah struktur jaringan yang ditingkatkan dimana ia dirancang berdasarkan Recurrent Neural Networks (RNN)[15]. Namun bedanya, RNN identic dengan jaringan multi layer feed-forward, banyak informasi historis yang dibawa oleh urutan panjang akan menyebabkan hilangnya gradien dan hilangnya informasi [15]. LSTM memberikan solusi dengan struktur internal yang lebih

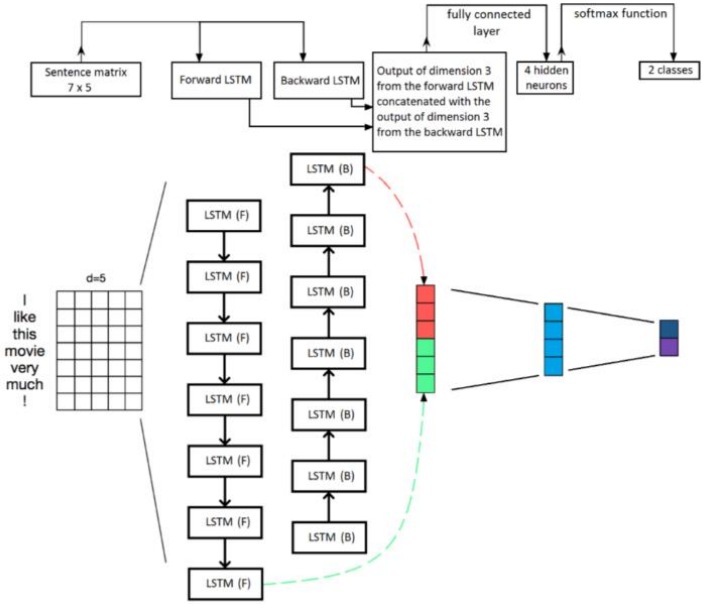
kompleks yang memungkinkan LSTM mengingat informasi baik jangka pendek maupun jangka panjang[16].



Gambar 2.1 Struktur LSTM oleh Jiarui Zhang [15]

Struktur LSTM seperti terlihat pada gambar menambahkan sel memori untuk menyimpan informasi historis, dan pembaruan, penghapusan dan output informasi historis dikontrol oleh masing-masing tiga gate, yaitu input gate, forget gate dan output gate. Input gate digunakan untuk menentukan bagaimana vektor yang masuk mengubah keadaan sel memori. Output gate memungkinkan sel memori memiliki efek pada output. Pada akhirnya, gerbang lupa dapat memungkinkan sel memori untuk mengingat atau melupakan informasi sebelumnya [15].

Kelemahan dari LSTM adalah tidak cukup memperhitungkan informasi setelah kata karena kalimat dibaca hanya satu arah, yaitu kedepan. Solusinya adalah menggunakan bi-directional LSTM, dengan membaca kalimat kedepan dan kebelakang dimana luarannya disusun bersama [7].



Gambar 2.2 Arsiterktur Bi-LSTM[7]

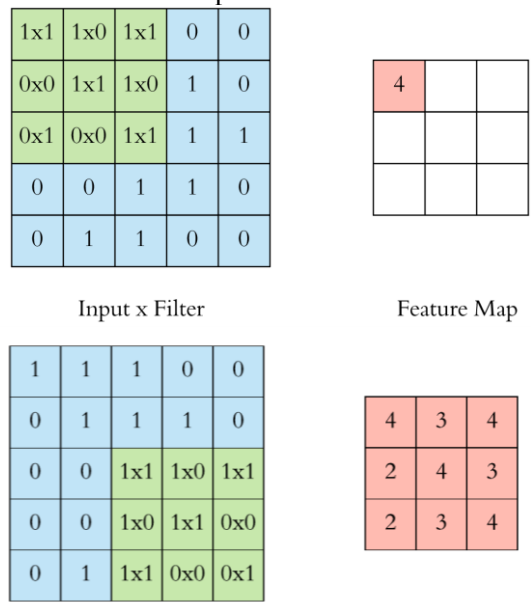
2.2.10 Convolutional Neural Network

Convolutional Neural Network (CNN) adalah golongan Artificial Neural Network (ANN) yang menggunakan layer konvolusional untuk menyaring masukan demi informasi yang berguna. Pada tahun 2012, Alex Krizhevsky dengan penerapan CNN miliknya berhasil menjuarai kompetisi ImageNet Large Scale Visual Recognition Challenge 2012. Dari prestasi tersebut, dapat diketahui bahwa metode Deep Learning, khususnya CNN dapat lebih unggul dari metode Machine Learning lainnya seperti SVM pada kasus klasifikasi citra[17]. Meskipun pada mulanya CNN dibuat untuk mengklasifikasikan citra, namun kemudian model CNN terbukti efektif digunakan untuk Natural Language Processing[18]. Berikut penjabaran arsitektur dari CNN:

2.2.10.1 Convolution Layer

Konvolusi adalah suatu istilah matematis yang berati mengaplikasikan sebuah fungsi pada output fungsi lain secara

berulang [17]. Konvolusi diaplikasikan pada data masukan menggunakan convolution filter atau kernel untuk menghasilkan feature map[19]. Operasi yang dilakukan kernel adalah operasi perkalian matriks dan menjumlahkan hasilnya, kemudian hasil tersebut masuk ke *feature map*. Operasi berhenti dilakukan saat semua nilai pada dimensi vektor terkakulasi.



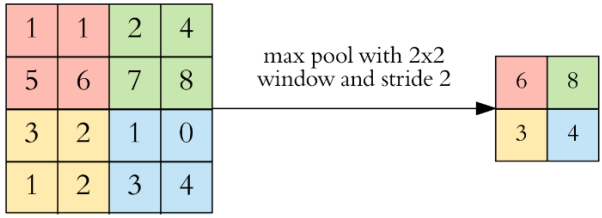
Gambar 2.3 Proses Konvolusi[19]

Tipe data teks biasanya dilakukan beberapa konvolusi pada input, masing-masing menggunakan filter yang berbeda dan menghasilkan *feature map* yang berbeda. Kemudian menumpuk semua *feature map* ini bersama-sama dan itu menjadi hasil akhir dari *convolution layer* [18].

2.2.10.2 Subsampling Layer

Subsampling adalah proses mereduksi ukuran sebuah data, diaplikasikan setelah *convolution layer*. Sebagian besar CNN menggunakan max pooling sebagai metode subsampling.

Max pooling membagi output dari *convolution layer* menjadi beberapa grid kecil lalu mengambil nilai maksimal dari setiap grid untuk menyusun matriks yang telah direduksi [17]. Seperti terlihat pada ilustrasi dibawah, max pooling terhadap *window* 2x2.

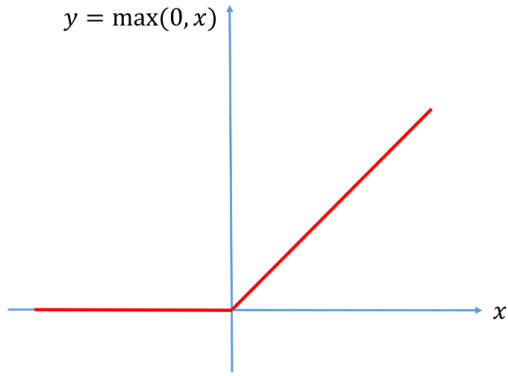


Gambar 2.4 Proses *Max-Pooling*[19]

Operasi max-pooling bertujuan untuk mengekstrak fitur yang paling penting untuk tiap konvolusi [7].

2.2.10.3 ReLu Layer

ReLu memiliki kepanjangan Rectified Linear Unit, memiliki tujuan untuk meningkatkan nonlinearitas dari CNN. Relu Layer mengubah semua nilai negatif menjadi nol [20].



Gambar 2.5 Grafik ReLu[20]

ReLU layer disebut juga fungsi aktivasi pada neural network termasuk CNN. Lapisan ini tidak merubah ukuran dari input vektor hanya merubah nilainya saja. Terlihat pada gambar

2.5 bahwa persamaan fungsi ReLu cukup sederhana, yaitu $y = \max(0, x)$. Yang artinya apabila *input* x kurang dari nol, *output*-nya adalah 0 dan apabila *input* x lebih dari nol maka nilai *output* adalah x. Untuk fitur yang tidak diaktivasi, layer ReLu menetapkan gradien menjadi nol. Untuk fitur yang diaktivasi, gradien dilakukan *backpropagation* tanpa ada perubahan [19].

2.2.10.4 Fully Connected Layer

Fully connected layer merupakan lapisan terakhir pada CNN, setelah setiap neuron pada convolutional layer ditransformasi menjadi data satu dimensi terlebih dahulu. Fully connected layer menggabungkan output dari layer sebelumnya dan bertujuan untuk melakukan transformasi pada dimensi data agar data dapat diklasifikasikan secara linear [16].

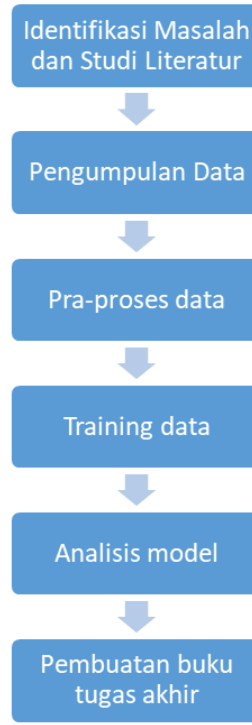
Halaman ini sengaja dikosongkan

BAB III
METODOLOGI

Pada bagian ini dijelaskan metodologi yang akan digunakan sebagai panduan untuk menyelesaikan penelitian tugas akhir ini.

3.1 Diagram Metodologi

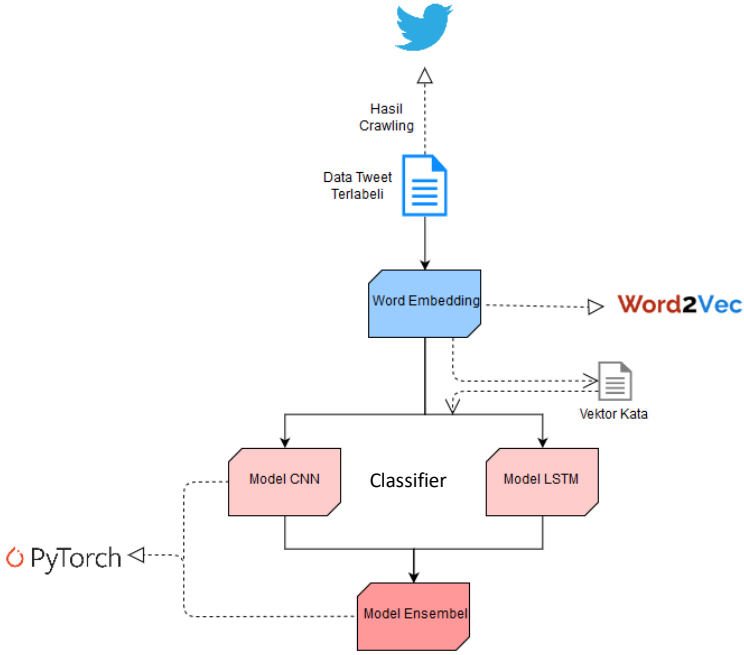
Pada bagian ini akan dijelaskan metodologi dari penelitian yang akan digunakan sebagai panduan sistematis agar pengerjaan dari tugas akhir menjadi terarah dan berjalan sesuai rencana. Berikut ini merupakan metodologi yang digunakan penulis.



Gambar 3.1 Diagram Metodologi

3.2 Arsitektur Penelitian

Pada bagian ini merupakan penjelasan secara garis besar arsitektur pada penelitian ini, mencakup aktivitas dan alur prosesnya. Arsitektur dapat dilihat pada gambar berikut.



Gambar 3.2 Arsitektur Penelitian

3.3 Uraian Metodologi

Berikut merupakan penjelasan dari setiap tahapan yang ada pada metodologi yang digunakan, yaitu:

3.2.1 Identifikasi Masalah dan Studi Literatur

Pada Pada tahap identifikasi masalah merupakan tahap awal untuk memulai penyusunan tugas akhir ini. Proses identifikasi masalah ini didukung dengan dilakukannya studi literatur sesuai permasalahan untuk memahami teori dan metode sehingga dapat memberikan solusi yang dapat digunakan dalam

tugas akhir ini. Adapun literatur utama yang digunakan dalam tugas akhir ini adalah SemEval-2017 Task 4: Twitter Sentiment Analysis with CNN's and LSTMs. Tahap ini akan menghasilkan rumusan masalah dan latar belakang yang dijadikan dasar untuk memulai tugas akhir ini.

3.2.2 Pengumpulan Data

Tahap selanjutnya merupakan pengumpulan data tweet yang digunakan untuk training data. Data didapatkan dari kiriman-kiriman teks di media sosial twitter melalui metode crawling. Data yang dikumpulkan untuk analisis sentimen terkait topik operator telekomunikasi.

3.2.3 Pra-pemrosesan Data

Tahap selanjutnya adalah tahap pra-pemrosesan data, tahap ini bertujuan untuk memproses data agar siap diolah model klasifikasi. Hasil dari tahapan pra-pemrosesan data akan dimanfaatkan dalam proses training data. Di dalam pra-premosesan data sendiri termasuk pemberian bobot awal oleh *word embedding*.

3.2.4 Training Data

Pada tahap training data, data yang sudah melewati pra-pemrosesan data akan berbentuk vektor. Data akan dibagi menjadi dua jenis yaitu data training dan data testing. Parameter akan disesuaikan agar mendapat akurasi terbaik. Pada tahap training data diharapkan mendapatkan model terbaik yang akan digunakan. Pada tahap ini akan dilakukan beberapa iterasi (skenario) berdasarkan subtask, metode word embedding dan konfigurasi parameter pada CNN dan LSTM. Luaran pada tahap ini adalah model dari setiap skenario training yang akan dilakukan.

3.2.5 Analisis Model

Tahap ini dilakukan analisis untuk mengevaluasi seberapa baik performa yang dihasilkan oleh model Long Short Term Memory dan Convolutional Neural Network. Setiap model diuji dengan pengukuran nilai akurasi, recall, presisi dan f-measure. Metrik-metrik tersebut membutuhkan data hasil prediksi dan data aktual yang direpresentasikan dengan istilah :

- True Positive (TP) adalah titik data yang diklasifikasikan model sebagai positif, dan label sesungguhnya memang positif yang berarti prediksi yang dihasilkan betul.
- True Negative (TN) adalah titik data yang diklasifikasikan model sebagai negatif, dan label sesungguhnya memang negatif yang berarti prediksi yang dihasilkan betul.
- False Positive (FP) adalah titik data yang diklasifikasikan model sebagai positif, dan label sesungguhnya adalah negatif yang berarti prediksi yang dihasilkan salah.
- False Negative (FN) adalah titik data yang diklasifikasikan model sebagai negatif, dan label sesungguhnya adalah positif yang berarti prediksi yang dihasilkan salah.

Kemudian data tersebut dimasukkan pada pengukuran yang masing-masing memiliki formula sebagai berikut.

- Akurasi $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$
- Presisi $Precission = \frac{TP}{TP+FP}$
- Recall $Recall = \frac{TP}{TP+FN}$
- F-Measure $= 2 * \frac{Precission*Recall}{Precission+Recall}$
-

3.2.6 Dokumentasi

Tahap terakhir penelitian ini adalah melakukan dokumentasi berbentuk laporan tugas akhir. Dokumentasi dibuata dalam bentuk dokumen yang seudah disesuaikan dengan format yang telah ditentukan. Penyusunan dokumentasi dilakukan secara sistematis sesuai dengan pengerjaan tugas akhir, dari awal hingga berakhirnya proses.

BAB IV
PERANCANGAN

Bab ini menjelaskan tentang perancangan dari luaran tugas akhir ini. Perancangan yang dibuat berupa rancangan pra-pemrosesan data dan rancangan model.

4.1 Perancangan Pengumpulan Data

Data yang akan digunakan untuk proses pemodelan Convolutional Neural Network maupun Long Short Term Memory didapatkan dari penelitian sebelumnya. Data tersebut merupakan kiriman-kiriman teks di media sosial twitter yang diakuisisi dengan metode crawling. Data yang berjumlah 42.720 tweet tersebut sudah melalui tahap pembersihan dan pelabelan data. Berikut adalah distribusi label dataset beserta sampel tweet.

Tabel 4.1 Distribusi Label Dataset dan Sampel Tweet

Label Sentimen	Jumlah Tweet	Sampel Tweet
Sangat Negatif	2325	#Im3 ne Iklan Sampah.. Kreatif dikit Napa.. Ikut2an As Telkomsel.. Maluuuuuuu..
Negatif	2344	Membalas @myXL Makin lama makin jelek, paket dah mahal pula, apaan 4g kek gini
Netral	2384	Membalas @Telkomsel Kalo kayak gini gimana coba aktifinnyaaaaa !!! Kasih tau dong caranyaaaaaa
Positif	2352	Sinyal xl kencang badai, Youtube pun lancar maksimal
Sangat Positif	2310	Smart juga :) set jam download 100MB lewat BB

		RT @tasialalala: Sinyal tri pagi2 emang mantap jaya.

Dataset tarsebut dibagi menjadi skenario atau subtask klasifikasi sentimen yang memiliki perlakuan label yang berbeda. Tabel 4.2 menunjukkan deskripsi tiap subtask.

Tabel 4.2 Deskripsi Tiap Subtask

Subtask	Label	Perlakuan
A	**Postif, Negatif**	Mengubah label sangat positif menjadi positf, mengubah label sangat negatif menjadi negatif dan menghapus data yang memiliki label netral.
B	**Postif, Netral, Negatif**	Mengubah label sangat positif menjadi positf, mengubah label sangat negatif menjadi negatif
C	**Sangat Positif, Postif, Netral, Negatif, Sangat Negatif**	Tidak ada perlakuan.

4.2 Perancangan Pra-Pemrosesan Data

Sebelum data diproses oleh model klasifikasi, data harus diberi perlakuan agar siap diproses atau istilahnya pra-pemrosesan data. Tujuannya untuk efisiensi proses dengan membuat isi kalimat sesedikit mungkin namun mempertahankan token yang bernilai informasi agar

memudahkan dalam proses selanjutnya. Tahapan pra-pemrosesan data adalah sebagai berikut.

4.2.1. Perancangan Perubahan URL dan Mention menjadi Token

URL atau Uniform Resource Locator yang digunakan untuk mereferensi suatu alamat website dan mention sebagai tanda menyebut akun lain seringkali ditemukan pada kiriman-kiriman di media sosisal twitter. Adanya kemungkinan bahwa URL dan mention dalam suatu kiriman tweet mempunyai maksud tertentu dalam suatu struktur kalimat, maka penelitian ini diputuskan untuk mengubah URL dan mention menjadi token. Berikut adalah skenario perubahan URL dan mention.

- URL yang diindikasikan dengan awalan tanda "://" diubah menjadi <url>
- Mention yang diindikasikan dengan awalan tanda "@" diubah menjadi <mention>

4.2.2. Perancangan Penghapusan Huruf yang Berulang

Twitter sebagai media komunikasi sehari-hari berimbas ke penggunaan kata-kata yang tidak baku pada penulisan kiriman tweet agar terkesan santai dan akrab. Kata yang memiliki huruf berulang seperti "maakasiiiii" seringkali ditemui. Kata tersebut akan dibatasi perulangan hurufnya sebanyak maksimal 2 huruf. Maka kata "maakasiiiii" akan diubah menjadi "maakasii".

4.2.3. Perancangan Perubahan Emoticon menjadi Token

Dari dataset yang dikumpulkan didapati banyak simbol emoticon dalam tweet yang merepresentasikan ekspresi wajah seperti marah, senyum dan sedih dalam bentuk tulisan. Adanya emoticon dalam kalimat tweet membantu pengenalan emosi lebih jelas. Sehingga emoticon akan diubah menjadi token yang merepresentasikan emoticon tersebut. Berikut emoticon yang mengalami perubahan.

- <3, :D, :P, :O, :X, :*. :3, =/, XD
- Diubah ke token senyum :)), :), :-)), :-), ((:, (:, ((-:, (-:, =)), =), ^_^

- Diubah ke token sedih :((, :(, :-((, :-(,)):,):,))-:,)-:
- Diubah ke token berkedip ;)), ;)
- Diubah ke token berduka :')), :'), :'((, :'(, ((':, (':
- Diubah ke token terganggu :/, :\\
- Diubah ke token kategori wajah datar :|, :-|

4.2.4. Penghapusan Penghapusan Tanda Baca, Simbol dan Angka

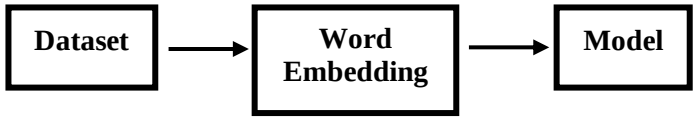
Dari dataset yang dikumpulkan didapati banyak simbol-simbol dan angka yang kurang memiliki makna dan tidak membantu mengenali sentimen dalam proses training. Maka simbol dan angka dihilangkan untuk memperkecil isi kalimat.

Proses menghilangkan tanda baca dan simbol dibantu oleh regex atau regular expression dalam bahasa python. Simbol-simbol yang akan dihilangkan adalah :

- Menghapus semua karakter ASCII, seperti "$", "~", "@"
- Menghapus semua karakter non-ASCII, seperti "é","ö", "ĉ"
- Menghapus semua karakter angka, seperti 1, 2, 3, 4, 5, 6, 7, 8, 9, 0

4.3 Perancangan Word Embedding

Word Embedding yang digunakan sebagai rujukan vektor kata pada proses training model menggunakan Word2Vec yang telah di training pada penelitian sebelumnya. Proses yang dilakukan pertama adalah memuat dataset kemudian dimasukkan word embedding untuk diberi bobot awal.



Gambar 4.1 Proses Word Embedding

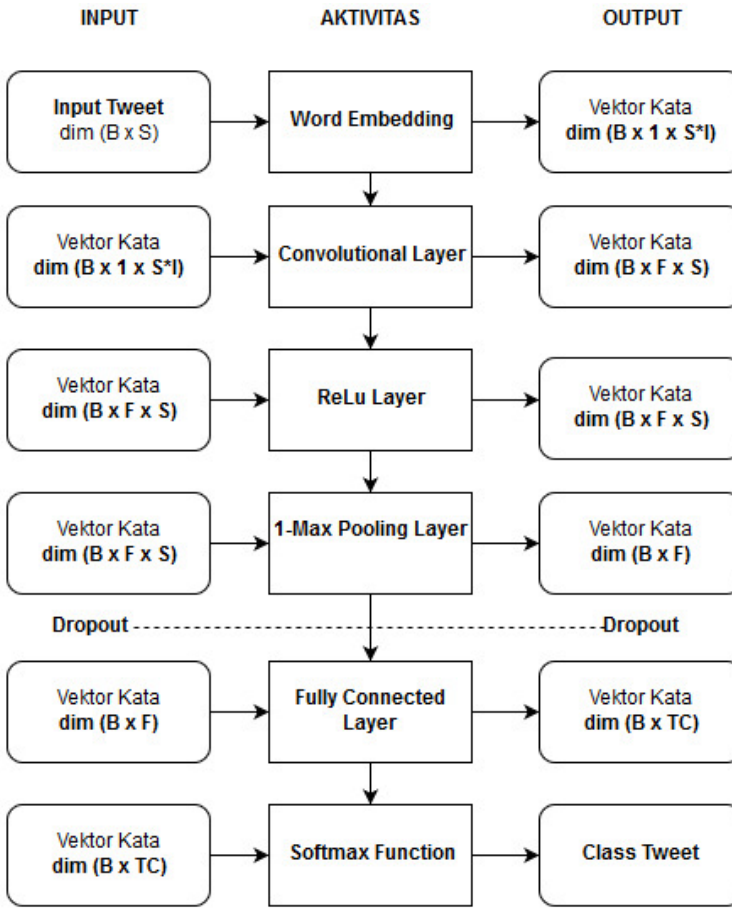
Model Word2Vec telah di training dengan algoritma skip-gram dan menggunakan dataset kiriman-kiriman twitter

berbahasa Indonesia. Model word embedding memiliki ukuran output 300 dimensi vektor setiap kata.

4.4 Perancangan Model Convolutional Neural Network

Pada perancangan model CNN, penulis akan memfokuskan ke arsitektur CNN yang akan dipakai, yang mana hampir serupa dengan CNN yang diterapkan pada penelitian *Convolutional Neural Networks for Sentence Classification* milik Yoon Kim[18]. Sebelum memasuki model CNN, *tweet* sebagai masukan akan transformasi ke bentuk token atau kata. ukuran matriks sebesar *B x S* dimana *B* merupakan *batch size* yang digunakan tiap iterasi epoch dan *S* merupakan *sequence* yang merupakan jumlah rangkaian kata dalam 1 *tweet*. Besarnya ukuran *batch* akan mempengaruhi kecepatan pemrosesan data, semakin kecil *batch* akan semakin panjang pula waktu yang dibutuhkan. Penelitian ini menggunakan *batch size* sebesar 50. Tiap kata akan dipetakan ke representasi vektor kata yang diperoleh dari kamus word embedding.

Word embedding yang dipakai memiliki dimensi matriks 300 (diberi simbol I) dikalikan dengan jumlah sequence, misal didalam satu tweet memiliki 34 kata, maka dimensi luaran proses word embedding adalah (50 x 1 x 11100). Luaran dari proses word embedding menjadi masukan satu proses konvolusi model CNN dan menghasilkan feature maps. Luaran proses konvolusi memiliki ukuran dimensi menyesuaikan dengan ukuran feature maps (diberi simbol F) sehingga ukuran dimensinya adalah (B x F x S). Proses berlanjut setelah melalui layer konvolusi, kemudian memasuki layer ReLu dan 1-Max Pooling. Detail proses yang terjadi didalam layer telah dijelaskan pada bab 2.2.10. Arsitektur CNN dideskripsikan pada gambar dibawah ini.



Gambar 4.2 Arsitektur CNN

Sebelum proses berlanjut ke Fully Connected Layer, data di-dropout untuk mengurangi overfitting. Luran proses Fully Connected Layer berupa matriks yang memiliki dimensi Batch x Target Class (B x TC), besarnya target class berdasarkan banyaknya label pada tiap subtask klasifikasi. matriks telah dirubah menjadi dua dimensi Kemudian mengimplementasikan fungsi softmax untuk mendapatkan nilai prediksi klasifikasi akhir dari label tweet.

Proses training model CNN akan dilakukan berulang kali dengan kombinasi parameter yang berbeda-beda dengan harapan mendapatkan performa CNN yang terbaik. Parameter-parameter ini dipilih berdasarkan penelitian sebelumnya yang menunjukkan hasil peforma model yang terbaik. Merujuk pada penelitian sebelumnya, Tabel 4.3 menunjukkan parameter yang akan digunakan dalam proses training model Convolutional Neural Network.

Tabel 4.3 Parameter Model CNN

Parameter	Nilai	Deskripsi
Filter Region Size	1-3	Menentukan jumlah dan ukuran dimensi dari kernel atau filter kata yang digunakan dari *region size* yang dihasilkan pada proses konvolusi
Feature Number	100, 200	Menentukan jumlah dan ukuran dari *feature maps* yang dihasilkan dari proses konvolusi, nilai akan dibatasi hingga 200 karena adanya keterbatasan prosesor
Norm Limit	3	Koefisien reguralisasi (12 regularisasi)
Dropout Rate	0, 5	Nilai dropout saat proses training
Subtask	A, B, C	Jenis subtask yang dipilih pada saat proses training

Tahap perancangan skenario training bertujuan untuk mendapatkan performa model CNN terbaik dengan cara

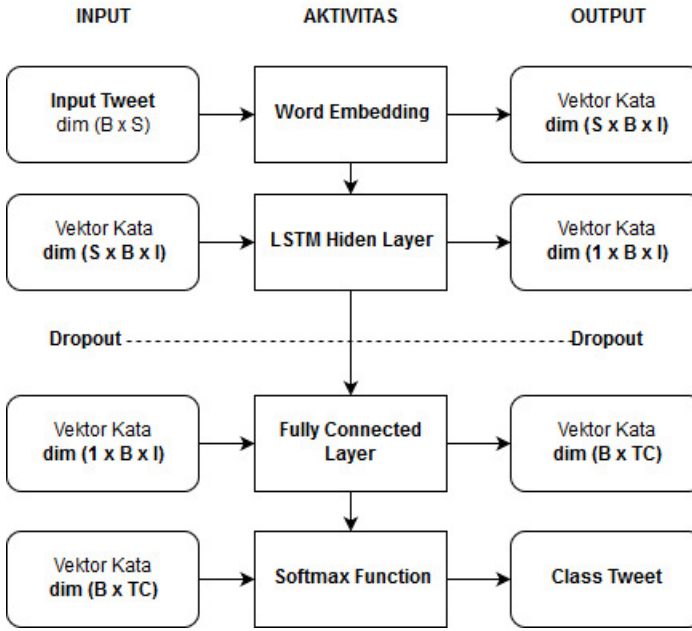
mengatur *hyperparameter* CNN. Dalam proses training dataset, penelitian ini menggunakan metode 10 *fold cross-validation* dimana dataset dibagi menjadi 10 bagian, 9 bagian digunakan untuk data training dan 1 bagian digunakan untuk data testing. Proses training dilakukan sebanyak 10 kali dengan kombinasi bagian yang berbeda untuk data testing dan training sehingga model dapat belajar lebih luas dan efektif.

4.5 Perancangan Model Long Short Term Memory

Perancangan arsitektur model Long Short Term Memory yang digunakan dalam penelitian ini seperti yang terlihat pada gambar 4.3, serupa dengan implementasi Long Short Term Memory pada paper berjudul SemEval-2017 Task 4: Twitter Sentiment Analysis with CNN's and LSTMs[7].

Masukan berupa tweet yang memiliki dimensi Batch size x Sequence akan melewati proses word embedding seperti halnya model CNN. Luaran proses embedding ini memiliki ukuran dimensi S x B x I (Sequence x Batch size x Input Dimensi Embedding). Jika sequence bernilai 34 dan batch size bernilai 50, maka matriks berdimensi 34 x 50 x 300 menjadi input LSTM hidden layer.

Kemudian satu per satu tweet yang berbentuk vektor berdimensi (1 x 50 x 300) akan memasuki Fully Connected Layer untuk diklasifikasikan secara linear. Sehingga luaran dari *Fully Connected Layer* berupa vektor kata yang berdimensi B x TC (Batch Size x Target Class yang sesuai dengan subtask). Agar setara dengan luaran akhir yang dihasilkan oleh model CNN, maka luaran proses Fully Connected Layer model LSTM ini diaplikasikan fungsi softmax untuk mendapatkan nilai prediksi klasifikasi akhir dari label tweet.



Gambar 4.3 Arsitektur LSTM

Proses training model LSTM akan dilakukan berulang kali dengan kombinasi parameter yang berbeda-beda dengan harapan mendapatkan performa LSTM yang terbaik. Berikut merupakan parameter yang akan digunakan dalam proses training model Long Short Term Memory.

Tabel 4.4 Parameter Model LSTM

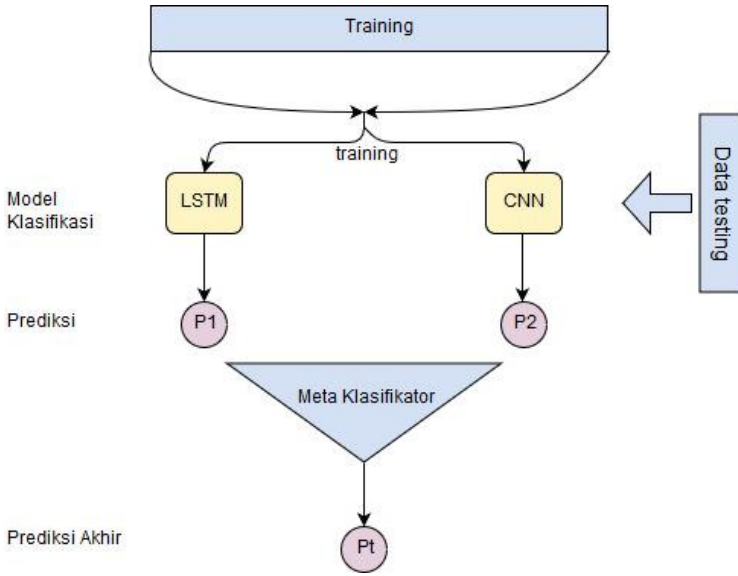
Parameter	Nilai	Deskripsi
Number of Layer	1	Menentukan jumlah lapisan LSTM yang digunakan pada proses klasifikasi

Hidden Dimension	300	Menentukan banyaknya node pada hidden layer LSTM
Bidirectional	True False	Menentukan penggunaan LSTM *bidirectional* atau LSTM *forward* saja.
Dropout Rate		Nilai dropout saat proses training
Subtask		Jenis subtask yang dipilih pada saat proses training

4.6 Perancangan Evaluasi Ensembel Model

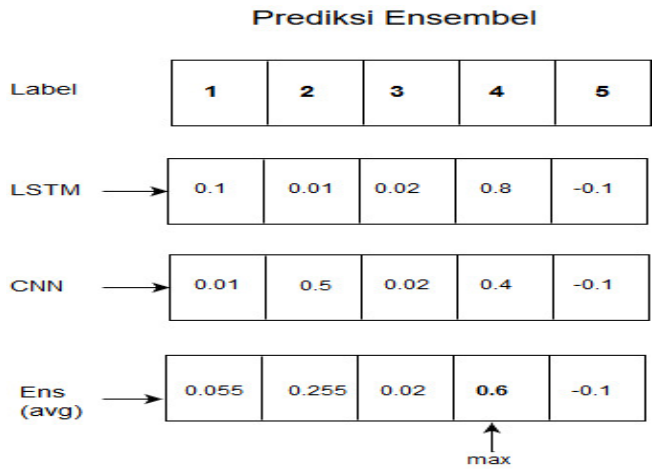
Dalam machine learning, mengansambel model bertujuan untuk meminimalisir varians dan mengurangi overfitting sehingga dapat menigkatkan performa model. Menggabungkan atau mengansambel beberapa model memiliki banyak macam metode. Metode yang dipilih untuk menggabungkan model CNN dan LSTM dalam penelitian ini adalah penjumlahan prediksi dari masing-masing model kemudian dirata-rata untuk mendapatkan prediksi akhir. Berikut adalah arsitektur ensembel model, atau biasa disebut dengan istilah *soft voting*.

Gambar 4.4 mendeskripsikan bagaimana proses data hingga mendapatkan hasil prediksi. Model CNN dan LSTM di *training* dengan dataset yang sama secara *end to end*. Keduanya memberikan hasil hasil prediksi masing-masing, kedua hasil digabungkan untuk menjadi prediksi akhir. tersebut ditambahkan kemudian dirata-rata. Hasil akhir prediksi model ensembel merupakan nilai terbesar dari semua label.



Gambar 4.4 Proses Ensembel Model

Pada gambar 4.5 mendeksripsikan bagaimana proses *soft voting* dalam penelitian ini berlangsung untuk menghasilkan prediksi.



Gambar 4.5 Metode *Soft Voting*

Halaman ini sengaja dikosongkan

BAB V
IMPLEMENTASI

Pada bab ini, akan dijelaskan mengenai implementasi dari perancangan yang telah dilakukan pada bab sebelumnya. Bagian implementasi akan menjelaskan mengenai lingkungan implementasi, pembuatan pra-pemroses data dan pembuatan model dalam bentuk kode.

5.1 Lingkungan Implementasi

Pengerjaan penelitian ini dilakukan pada perangkat keras komputer yang memiliki spesifikasi sebagai berikut.

Tabel 5.1 Spesifikasi Komputer

Processor	Intel i5 8400K
Memory	16GB DDR4 *Memory*
GPU	GTX 1070 8gb
OS	Linux Ubuntu 16.10
Arsitektur	64 bit

Selain itu, beberapa perangkat lunak, bahasa pemrograman dan *library* yang membantu pengerjaan penelitian ini dapat dilihat pada table 5.2.

Tabel 5.2 Perangkat Lunak yang Dipakai

Bahasa Pemrograman	Python
Editor	Sublime Text 3, Visual Studio Code
Library	• Pytorch • Pandas • Tweepy • Torchtext

	• Matplotlib • Emot • Gensm

5.2 Pra-Pemrosesan Data

Pada tahap implementasi, pra-pemrosesan data mencakup aktifitas pemuatan dataset yang sebelumnya disimpan dalam file berformat .txt, pembersihan dataset dan proses memasukan kalimat dan label tweet. Baris kode yang memberi perintah untuk memanggil dataset adalah method read_dataset seperti terpampang pada Kode 5.1.

```
1. def read_dataset(self,type,subtask):
2.
3.     if type == 'indo':
4.         if subtask == 'C':
5.             file_name = 'Data/data_subtaskC.txt'
6.         elif subtask == 'B':
7.             file_name = 'Data/data_subtaskB.txt'
8.         elif subtask == 'A':
9.             file_name = 'Data/data_subtaskA.txt'
10.        else:
11.            file_name = 'Data/subj.txt'
12.
13.        indo_opr_data = ReadDataTopik(file_name)
14.
15.        twt_id_field = torchtext.data.Field(use_vocab
      =False, sequential=False)
16.        label_field = torchtext.data.Field(sequential
      =False)
17.        text_field = torchtext.data.Field()
18.
19.        fields = [('twt_id', twt_id_field), ('label',
       label_field), ('text', text_field)]
20.
21.        self.fields = fields
22.
23.        examples = [
```

```
24.                 torchtext.data.Example.fromlist([twt_i
      d, self.polarity_to_label(polarity), text], field
      s)
25.
26.                         for twt_id, polarity, text in
       indo_opr_data
27.                     ]
28.         self.examples = examples
```

Kode 5.1 Proses Memanggil Dataset

Agar dataset yang diproses sesuai *subtask* yang dipilih, maka dilakukan percabangan yang dimulai dari baris ke-3 kemudian disimpan dalam variabel *file_name*. Kemudian dibaris ke-13 file_name dijadikan nilai yang diberikan untuk *instance* yang dibuat dari kelas *ReadDataTopik()* yang berfungsi untuk membersihkan dataset dan disimpan dalam variabel *instance* indo_opr_data. Setelah itu pada baris ke 15 hingga 17, tiga variabel berjenis field dibuat untuk menampung id unik tiap tweet, label tweet dan kalimat tweet. Ketiga variabel *twt_id_field*, *label_field* dan *text_field* ini memanfaatkan library torchtext. Ketiga variabel tersebut diisi data dari variabel *instance* indo_opr_data menggunakan perulangan kemudian disimpan dalam variabel *examples* yang tertera pada baris kode ke 23-28.

```
1.    class ReadDataTopik:
2.        def __init__(self, file_name):
3.            self.file_name = file_name
4.            self.emoticons = {}
5.            self.emojis    = {}
6.
7.        def replace_URL(self, string):
8.            tokens  = ['<url>' if '://' in token else
       token
9.                       for token in string.split()]
10.           return ' '.join(tokens)
11.
12.       def replace_mention(self, string):
```

```
13.           tokens  = ['<mention>' if token.startswit
      h('@') else token
14.                       for token in string.split()]
15.           return ' '.join(tokens)
16.
17.       def replace_mult_occurences(self, string):
18.           return re.sub(r'(.)\1{2,}', r'\1\1', stri
      ng)
```

Kode 5.2 Proses Membersihkan Dataset

Kode diatas merupakan pendefinisian kelas *ReadDataTopik()* yang memiliki 7 *function* untuk membersihkan dataset. *Method replace_URL()* pada baris ketujuh digunakan untuk mengubah URL menjadi token. *Method* selanjutnya adalah *replace_mention()* yang digunakan untuk mengubah *mention* yang dideteksi dengan simbol '@' pada awal kalimat menjadi token. Kemudian *method replace_mult_occurences()* yang ada pada baris ke-17 untuk menghapus huruf yang berulang dengan maksimal perulangan adalah 2.

```
1.   def replace_token_emoticon(self, token):
2.       if token.startswith('<3'):
3.         return ' <hati> '
4.       if token.startswith(':'):
5.         if token == ':D':
6.           return ' <tertawa> '
7.         if (token == ':P') | (token == ':p'):
8.           return ' <menjulurkan_lidah> '
9.         if (token == ':O') | (token == ':o'):
10.          return ' <terkejut> '
11.        if (token == ':x') | (token == ':*'):
12.          return ' <cium> '
13.        if token == ':3':
14.          return ' <malu-malu_kucing> '
15.      if token.startswith('='):
16.        if (token == '=/') | (token == '=\\'):
17.          return ' <terganggu> '
18.      if token == 'XD':
19.        return ' <tertawa_terbahak-bahak> '
```

```
20.      return token
```

Kode 5. 3 Proses Perubahan Emoticon menjadi String

Method replace_token_emoticon memberi perintah untuk mengubah emoticon yang ada dalam kalimat *tweet* menjadi token menggunakan percabangan bersarang. Emoticon yang telah didefinisikan pada bab 4.2.3 dideteksi oleh *method startswith()* yang ada pada baris kedua, empat dan lima belas.

```
1. def replace_emoticons(self, string):
2.     # Senyum
3.     string = string.replace(':))', ' <senyum_senyum>
       ')
4.     string = string.replace(':-)', ' <senyum> ')
5. . . .
6.     # Sedih
7.     string = string.replace(':((', ' <sedih_sedih> '
       )
8.     string = string.replace(':(', ' <sedih> ')
9. . . .
10.        # Berkedip
11.        string = string.replace(';))', ' <senyum_berk
       edip> ')
12.        string = string.replace(';)', ' <senyum_berke
       dip> ')
13.        # Tears
14.        string = string.replace(":'))", ' <menangis_b
       ahagia> ')
15.        string = string.replace(":')", ' <menangis_ba
       hagia> ')
16.    . . .
17.        string = string.replace(":'(", ' <menangis_se
       dih> ')
18.
19.        string = ' '.join([self.replace_token_emotico
       n(token)
20.                           for token in string.split
       ()])
21.        return string
```

Kode 5.4 Proses Perubahan Emoticon menjadi String

Seperti halnya *method replace_token_emoticon()*, *method replace_emoticons()* memberi perintah untuk mengubah emoticon menjadi token, selain itu juga memanggil *method replace_token_emoticon()* untuk menggabungkan nilai yang di *return* dan string yang dihasilkan oleh *replace_emoticons()* menggunakan *method join()* seperti terlihat pada baris ke-19 yang telah tersedia di Python.

```
1.  def clean_str(self, string):
2.      string = re.sub(r"[^A-Za-z0-
    9(),!?\'\`]", " ", string)
3.      string = re.sub(r"[^A-Za-z]+", " ", string)
4.      string = re.sub(r"\'s", " \'s", string)
5.      string = re.sub(r"\'ve", " \'ve", string)
6.      string = re.sub(r"n\'t", " n\'t", string)
7.      string = re.sub(r"\'re", " \'re", string)
8.      string = re.sub(r"\'d", " \'d", string)
9.      string = re.sub(r"\'ll", " \'ll", string)
10.     string = re.sub(r",", " , ", string)
11.     string = re.sub(r"!", " ! ", string)
12.     string = re.sub(r"\(", " \( ", string)
13.     string = re.sub(r"\)", " \) ", string)
14.     string = re.sub(r"\?", " \? ", string)
15.     string = re.sub(r"\s{2,}", " ", string)
16.     return string.strip()
17.
18. def divide_line(self, line):
19.     tokens      = line.split()
20.     id_tweet    = tokens[0]
21.     topik       = tokens[1]
22.     lbl_sntmnt  = tokens[2]
23.     lbl_topik   = tokens[3]
24.     lbl_sarkas  = tokens[4]
25.     message     = ' '.join(tokens[5:]).strip('"')

26.     return id_tweet, topik, lbl_sntmnt, lbl_topik,
    lbl_sarkas, message
```

Kode 5.5 Proses Menghapus Simbol dan Filter Huruf

Method selanjutnya yang terdapat pada kelas *ReadDataTopik()* adalah *clean_str()* yang bertujuan untuk menghapus simbol dan memastikan karakter yang terfilter hanya huruf. Dengan memanfaatkan modul *re* yang mendukung *regular expression* di Python, *re.sub()* yang dipanggil pada baris kedua hingga ke-15 memanggil fungsi substitusi untuk menghapus string yang muncul sesuai dengan pola pada parameter paling kiri yang kemudian akan diganti dengan parameter kedua. Apabila pola tidak ditemukan pada masukan, maka string tidak berubah. Kemudian *method divide_line()* digunakan untuk memisahkan kata per kata dalam *tweet* dan dijadikan satu oleh *method join()* seperti tertera pada baris ke-25.

```
1.  def __iter__(self):
2.
3.      with open(self.file_name, 'r', encoding='utf8'
    , errors='ignore') as f:
4.
5.          for line in f:
6.              id_tweet, topik, lbl_sntmnt, lbl_topik
    , lbl_sarkas, message = self.divide_line(line)
7.              message = self.replace_URL(message)
8.              message = html.unescape(message)
9.              message = message.replace('"', ' <kut
    ip> ')
10.             message = self.replace_mention(message
    )
11.             message = self.replace_mult_occurences
    (message)
12.             message = message.replace('..', ' <eli
    psis> ')
13.             message = self.replace_emoticons(messa
    ge)
14.             message = self.clean_str(message)
15.             message = self.replace_mention(message
    )
16.             message = message.lower()
17.             yield id_tweet, lbl_sntmnt, message
```

Kode 5. 6 Eksekutor Fungsi Pembersihan Dataset

Kode diatas merupakan *method* yang mengimplementasikan iterator yang disediakan oleh Python dimana ia membersihkan dataset yang dinamai *file_name* (parameter pertama method *open()* pada baris ketiga) dengan memanggil semua *method* yang ada dalam kelas *ReadDataTopik()* yang telah dijelaskan sebelumnya dan menghasilkan id tweet, label sentiment dan kalimat *tweet*. Hasil dari proses ini disimpan dalam variabel *message.*

```
1.  def polarity_to_label(self, polarity):
2.
3.      if int(polarity) == 1:
4.          label = 'strong negative'
5.      elif int(polarity) == 2:
6.          label = 'negative'
7.      elif int(polarity) == 4:
8.          label = 'positive'
9.      elif int(polarity) == 5:
10.         label = 'strong positive'
11.     else:
12.         label = 'neutral'
13.
14.     self.idq += 1
15.     return label
```

Kode 5.7 Konversi Label Numerik ke Kata

Pada saat memasukkan label *tweet* yang ada di dataset dari variabel *instance indo_opr_data* yang ada di *method read_dataset()*, fungsi *polarity_to_label()* dipanggil untuk melakukan konversi dari kode label numerik menjadi bentuk kata. Kondisi label numerik ada pada baris ke-3, kemudian label berbentuk kata ada pada baris ke-4 dan seterusnya.

5.3 Pemuatan Word Embedding

Proses memuat word embedding yang merepresentasikan kata dalam tweet menjadi vektor diimplementasikan dalam method *read_model()* yang didefinisikan dalam kelas Embedding. Word Embedding yang dipakai, dibangun dari library gensim dan dalam file berekstensi *.bin*.

```
1.  def read_model(self,type,cat):
2.
3.      start = time.time()
4.
5.      if type=='indo':
6.          if cat == 'w2v':
7.              print('Loading Word2Vec')
8.              word2vec_model = KeyedVectors.load_wor
    d2vec_format(embeddings_path, binary=True, unicod
    e_errors='ignore')
9.
10.     end = time.time()
11.     print("Word Embedding loading done in {} secon
    ds".format(end - start))
12.
13.     self.word2vec = word2vec_model
14.     self.embed_dim = 300
15.
16.     return word2vec_model
```

Kode 5.8 Proses Membaca Model

Pada *method read_model()* diatas , kondisi *type* dan *cat* harus terpenuhi agar *method load_word2vec_format()* dijalankan. Parameter yang dibutuhkan seperti *path* model *word2vec*, metode format binary dan penyetelan *unicode_errors* menjadi *'ignore'*. Kemudian hasil dari memuat model disiman dalam variabel *word2vec_model*. Variabel *start* dan *end* menggunakan *method time()* untuk menghitung waktu yang dibutuhkan untuk memuat *word embedding*. Ukuran dimensi *word embedding* disimpan dalam variabel *embed_dim*.

```
1.  def create_fold_embeddings(self, embeddings, args,
      key_vector):
2.
3.      emb_init_values = []
4.      unk = []
5.      a = 0
6.      b = 0
7.
8.      if args.embeddings_source == 'none':
9.          for i in range(self.idx_to_vocab.__len__()
    ):
10.             word = self.idx_to_vocab.get(i)
11.             if word == '<unk>':
12.                 emb_init_values.append(np.random.u
    niform(-
    0.25, 0.25, args.embeddings_dim).astype('float32'
    ))
13.
14.             elif word == '<pad>':
15.                 emb_init_values.append(np.zeros(ar
    gs.embeddings_dim).astype('float32'))
16.
17.             elif word in key_vector.wv.vocab:
18.                 emb_init_values.append(key_vector.
    wv.word_vec(word))
19.                 b = b+1
20.             else:
21.                 emb_init_values.append(np.random.u
    niform(-
    0.25, 0.25, args.embeddings_dim).astype('float32'
    ))
22.                 a = a+1
23.                 unk.append(word)
24.
25.     self.emb_init_values = emb_init_values
26.
27.     known_word = b
28.     unknown_word = a
29.
30.     return known_word, unknown_word, emb_init_valu
    es
```

Kode 5. 9 Pemberian Bobot oleh Word Embedding

Kode diatas menunjukkan tahapan pemberian bobot awal oleh *word embedding*. Diawali dengan inisiasi variabel *embed_init_values* dan *unk* pada baris ke-3 dan 4, untuk menjadi nilai awal kondisi perulangan. Terdapat kondisi jika kata dideteksi sebagai *unk* atau kata tidak tersedia di kamus word embedding maka akan diberi nilai random seperti tertera pada baris kode ke-11 dan 12. Selain itu, baris kode ke-14 dan 15 memberi kondisi lain jika kata berupa <pad> atau merupakan hasil padding yang dibuat oleh embedding maka akan diberi nilai nol. Jumlah kata ”unk’ disimpan dalam variabel *unknown_word*. Jumlah kata yang ada didalam kamus *embedding* disimpan dalam variabel *known_word* pada baris ke-27 yang dihasilkan dari proses iterasi yang memenuhi kondisi pada baris ke-17. Ketiga variabel yang disebutkan setelah sintaks *return* adalah nilai yang dikembalikan saat *method create_fold_embedding()* dipanggil.

5.4 Pembuatan Model Convolutional Neural Network

Pembuatan kode untuk model Convolutional Neural Network memanfaatkan library yang memiliki kemampuan akselerasi GPU bernama Pytorch dan didefinisikan dalam satu kelas. Kode dibawah ini mendefinisikan nama kelas model CNN yang di-inherit dari torch.nn.module. Parameter-parameter yang akan dibutuhkan untuk model CNN seperti banyaknya kosa kata, jumlah label yang ada didalam dataset, word embedding, jumlah dan ukuran kernel dan lainnya didefinisikan sebagai berikut.

```
1.  class CNN_Kim2014(nn.Module):
2.
3.     def __init__(self, embed_num, label_num, embeddi
       ngs_dim, embeddings_mode, initial_embeddings,args
       ):
4.
5.       super(CNN_Kim2014, self).__init__()
6.
7.       self.embed_num    = embed_num  # vocab size
```

```
8.       self.label_num    = label_num
9.       self.embed_dim    = 300
10.      self.embed_mode   = embeddings_mode
11.      self.channel_in   = 1
12.      self.feature_num  = args.feature_num
13.      self.kernel_width = args.kernel_width
14.      self.dropout_rate = 0.5
15.      self.norm_limit   = 3
16.  . . .
```

Kode 5.10 Parameter Model CNN

Kode 5.11 yang merupakan lanjutan kode dari kelas CNN_Kim2014 memberi perintah untuk mengecek apakah jumlah *filter region size* berjumlah sama dengan jumlah *feature maps* menggunakan fungsi *assert*. Jika kondisi tersebut terpenuhi maka kode dapat dilanjutkan. Namun jika jumlah *filter region size* tidak sama dengan jumlah *feature maps* maka akan keluar *error* dan kode tidak dapat dilanjutkan.

```
1.  class CNN_Kim2014(nn.Module):
2.
3.    def __init__(self, embed_num, label_num, embeddi
     ngs_dim, embeddings_mode, initial_embeddings,args
     ):
4.
5.  . . .
6.
7.  assert (len(self.feature_num) == len(self.kernel_w
     idth))
8.
9.      self.kernel_num = len(self.kernel_width)
10.     self.embeddings = nn.Embedding(self.embed_num,
      self.embed_dim, padding_idx=1)
11.     self.embeddings.weight.data.copy_(torch.from_n
     umpy(initial_embeddings))
12.
13.     if self.embed_mode == 'static':
14.       self.embeddings.weight.requires_grad = False

15.
```

```
16.      convo = [nn.Conv1d(self.channel_in, self.featu
      re_num[i],self.embed_dim*self.kernel_width[i], st
      ride=self.embed_dim) for i in range(self.kernel_n
      um)]
17.      self.convs = nn.ModuleList(convo)
18.      self.linear = nn.Linear(sum(self.feature_num),
       self.label_num)
```

Kode 5.11 Pendefinisian Model CNN

Kemudian, dengan menggunakan *method nn.Embedding()* dari library pytorch pada baris kode ke-10 kita membuat objek *embedding*. Jumlah kosa kata dan dimensi embedding diberikan sebagai parameter, selain itu ada *padding_idx* yang digunakan untuk menyesuaikan ukuran vektor dengan kata terbanyak dalam satu kalimat di *mini batch*. Nilai yang diberikan pada kalimat yang kurang dari jumlah maksimal kata adalah array yang berisi nol.

Setelah itu, pada baris kode ke-11 ada *method weight.data.copy_()* yang memberikan bobot awal dari proses *word embedding* dan mengubahnya menjadi format array numpy. Dalam penelitian ini, *embedding* difokuskan pada variasi *static* dan *non-static*, perbedaannya embedding *non-static* diberi kesempatan oleh model untuk melakukan pembelajaran terhadap kata dan memperbaharui nilai vektor kata tersebut. Sehingga pada baris selanjutnya ditambahkan percabangan dengan kondisi apabila menggunakan mode *static* maka kode pada baris keempat belas *self.embedding.weight.requires_grad=False* berjalan.

Tahap berikutnya yaitu pada baris ke-16 membuat objek *convo* yang bertugas melakukan proses konvolusi menggunakan method *nn.Conv1d()*. Parameter yang digunakan adalah jumlah *channel*, diberi nilai 1 seperti umumnya CNN digunakan untuk *task* klasifikasi teks. Parameter selanjutnya adalah ukuran *feature maps*, ukuran dimensi embedding

dikalikan ukuran filter, ukuran stride dimana menunjukkan jarak perpindahan setiap pergerakan filter. Jumlah lapisan konvolusi ditentukan oleh jumlah filter yang dipakai, dimana konvolusinya satu buah untuk *single-region* dan *multi-region* untuk lebih dari satu konvolusi. Kemudian pada baris kode ke-17, lapisan konvolusi disimpan dalam method *nn.ModuleList()*. Lapisan *Fully Connected* dibangun dari *method nn.Linear()* dengan ukuran mengikuti jumlah fitur dan jumlah label sentiment seperti tertera pada baris ke-18.

```
1.  def forward(self, input):
2.
3.      batch_width = input.size()[1]
4.
5.      x = self.embeddings(input).view(-
    1, 1, self.embed_dim*batch_width)
6.
7.      conv_results = [
8.          F.max_pool1d(F.relu(self.convs[i](x)), batc
    h_width - self.kernel_width[i] + 1).view(-
    1, self.feature_num[i])
9.
10.         for i in range(len(self.feature_num))
11.     ]
12.     x = torch.cat(conv_results, 1)
13.     x = F.dropout(x, p=self.dropout_rate, training
    =self.training)
14.     x = self.linear(x)
15.     x = F.log_softmax(x, dim=1)
16.
17.     return x
```

Kode 5.12 Pembuatan Model CNN

Kode 5.12 memuat method *forward()* yang terdapat pada kelas CNN_Kim2014() untuk mengkomputasikan *forward pass* dari model CNN. Di *method forward()* ini fungsi-fungsi internal didefinisikan untuk merepresentasikan layer-layer di *Convolutional Neural Network* yang sedang dibangun. Dimulai

dari baris kelima yaitu proses *embedding* untuk merubah input berupa kalimat menjadi vektor disesuaikan dengan jumlah kata yang masuk. Memasuki tahap selanjutnya pada baris ketujuh yaitu proses konvolusi yang diiterasi berdasarkan banyaknya jumlah kernel yang diberikan. Kemudian hasil dari proses konvolusi diterima oleh layer ReLu dimana setiap nilai negatif diubah menjadi menjadi nol.

Luaran dari ReLu diterima oleh layer *Maxpool1d* untuk membagi output dari *convolution layer* menjadi beberapa grid kecil lalu mengambil nilai maksimal dari setiap grid untuk menyusun matriks yang telah direduksi. Setelah melalui layer Maxpool, hasil matriks disimpan dalam variabel *conv_results.* Tahap selanjutnya pada baris ke-13 memanggil fungsi *dropout* dari *library torch.nn.functional()* dengan *rate* yang didefinisikan di argumen. Kemudian diklasifikasikan secara linear dengan memanggil *Fully Connected Layer* pada baris ke-14. Fungsi softmax diimplementasikan pada baris ke-15 untuk mendapatkan nilai prediksi akhir model CNN. Dibawah ini merupakan perubahan ukuran dimensi saat *method forward()* dipanggil yang telah sesuai dengan penjabaran tahap perancangan pada subbab 4.4.1.

5.5 Pembuatan Model Long Short Term Memory

Dengan memanfaatkan *library* Pytorch, kode untuk model *Long Short Term Memory* dibuat dan didefinisikan dalam satu kelas. Kode dibawah ini mendefinisikan nama kelas model LSTM yang di-*inherit* dari torch.nn.module.

```
1.  class Net(nn.Module):
2.      def __init__(self, embed_num, label_num, embed
    ding_dim, embeddings_mode, initial_embeddings, ar
    gs):
3.
4.          super(Net, self).__init__()
```

```
5.          self.embed_num    = embed_num
6.          self.label_num    = label_num
7.          self.embed_dim    = 300
8.          self.embed_mode   = embeddings_mode
9.          self.hidden_size  = 300
10.         self.n_layers = 1
11.         self.dropout_rate = 0.5
12. . . .
```

Kode 5.13 Parameter Model LSTM

Parameter-parameter yang dibutuhkan untuk model LSTM seperti banyaknya kosa kata, jumlah label yang ada didalam dataset, mode *word embedding*, ukuran *hidden node* dan lainnya didefinisikan dalam *method init().*

```
1.  class Net(nn.Module):
2.
3.  . . .
4.
5.          self.embeddings = nn.Embedding(self.embed_
    num, self.embed_dim, padding_idx=1)
6.          self.embeddings.weight.data.copy_(torch.fr
    om_numpy(initial_embeddings))
7.
8.          self.lstm = nn.LSTM(self.hidden_size, self
    .hidden_size, self.n_layers, bidirectional=True)
9.
10.         self.fc = nn.Linear(self.hidden_size, self
    .label_num)
11.
12.     def _init_hidden(self, batch_width):
13.         hidden = torch.zeros(self.n_layers, batch_
    width, self.hidden_size)
14.         return Variable(hidden)
```

Kode 5. 14 Pendefinisian Model LSTM

Kode diatas merupakan lanjutan dari kelas Net() model LSTM. Pada baris kelima dengan menggunakan *method nn.Embedding()* dari library Pytorch untuk membuat objek *embedding*. Serupa dengan implementasi pada model CNN,

jumlah kosa kata dan dimensi embedding diberikan sebagai parameter, selain itu ada *padding_idx* yang digunakan untuk menyesuaikan ukuran vektor dengan kata terbanyak dalam satu kalimat di *mini batch*. Nilai yang diberikan pada kalimat yang kurang dari jumlah maksimal kata adalah array yang berisi nol. Kemudian, bobot awal dari proses *word embedding* diberikan dengan menjalankan *method weight.data.copy_()* dan mengubahnya menjadi format array numpy seperti tertera pada baris keenam.

Langkah selanjutnya adalah membuat objek yang merupakan *instance* dari *nn.LSTM() library* Pytorch yang nantinya mendefinisikan sel jaringan LSTM seperti tertera pada baris ke-8. Parameter yang digunakan adalah ukuran *hidden node*, banyaknya lapisan LSTM yang diberi nilai 1 dan parameter untuk opsi LSTM Bidirectional. Jika nilainya 'True' maka LSTM yang menjalankan proses *forward* dan *backward* seperti Bi-LSTM semestinya. Variabel self.fc pada baris ke-10 merupakan lapisan *Fully Connected* yang dibuat dari *method nn.Linear()* dengan ukuran mengikuti jumlah fitur dan jumlah label sentimen.

```
1.  class Net(nn.Module):
2.
3.  . . .
4.
5.      def forward(self, input):
6.
7.          batch_width = input.size(0)
8.          input = input.t()
9.
10.         embed = self.embeddings(input)
11.
12.         lstm_out, (ht, ct) = self.lstm(embed)
13.
14.         ht2 = F.dropout(ht, p=self.dropout_rate, t
    raining=self.training)
15.
```

```
16.         fc_output = self.fc(ht2[-1])
17.
18.         tag_scores = F.log_softmax(fc_output, dim=
    1)
19.
20.         return tag_scores
```

Kode 5.15 Pembuatan Model LSTM

Kode 5.15 memuat method *forward()* yang terdapat pada kelas *Net()* untuk mengkomputasikan *forward pass* LSTM. Di *method forward()* ini layer-layer di *Long Short Term Memory* disusun. Dimulai dari baris kode ke-8 dimana ia men-*transpose* inputan model kemudian baru dilanjutkan proses embedding. Proses embedding pada baris ke-10 untuk mengubah input berupa kalimat menjadi vektor disesuaikan dengan jumlah kata yang masuk. Pada baris ke-12, jaringan LSTM yahg telah didefinisikan pada method init() dipanggil untuk melakukan *forward pass* dari hasil tahap *embedding*.

Tahap selanjutnya memanggil fungsi *dropout* dari *library torch.nn.functional()* yang disimpan dalam variabel ht2 seperti tertera pada baris ke-14. Kemudian variabel *tag_scores* yang ada pada baris ke-16 mengklasifikasikan masukan secara linear dengan memanggil *Fully Connected Layer*. Fungsi softmax yang dipanggil pada baris ke-18 diimplementasikan untuk mendapatkan nilai prediksi akhir model LSTM yang sepadan dengan nilai prediksi akhir model CNN agar memudahkan untuk mengevaluasi model ensembel. Dibawah ini merupakan perubahan ukuran dimensi saat *method forward()* dipanggil yang telah sesuai dengan penjabaran tahap perancangan pada subbab 4.5.1.

```
Ukuran Dimensi Input torch.Size([50, 47])

Ukuran Dimensi pada Embedding torch.Size([47, 50, 300])

Ukuran Dimensi pada Hidden torch.Size([2, 50, 300])

Ukuran Dimensi Akhir  torch.Size([50, 2])
```

Gambar 5.1 Ukuran Dimensi Layer LSTM

5.6 Pembuatan Ensembel Model

Pada tahap perancangan model ensembel telah dijelaskan bahwa sebelum mendapatkan prediksi akhir hasil ensembel model, baik model LSTM maupun CNN harus di *training* dengan dataset yang telah dibagi 9 dari 10 bagian untuk *training*. Baru kemudian setelah masing-masing model mendapatkan pembelajaran, model di evaluasi dengan dataset testing.

```
1. def cross_validate(fold, data, embeddings, args, key
      _vector, subtask):
2.
3.   actual_counts     = defaultdict(int)
4.   predicted_counts  = defaultdict(int)
5.   match_counts      = defaultdict(int)
6.
7.   split_width = int(ceil(len(data.examples)/fold))
8.
9.   for i in range(fold):
10.
11.       train_examples  = data.examples[:]
12.
13.       del train_examples[i*split_width:min(len(data
      .examples), (i+1)*split_width)]
14.
15.       test_examples   = data.examples[i*split_width
      :min(len(data.examples), (i+1)*split_width)]
16.
17.       total_len = len(data.examples)
18.       train_len = len(train_examples)
19.       test_len  = len(test_examples)
20.
```

```
21.       train_counts = defaultdict(int)
22.       test_counts  = defaultdict(int)
23.
24.       for example in train_examples:
25.
26.         train_counts[example.label] += 1
27.
28.       for example in test_examples:
29.
30.         test_counts[example.label] += 1
31.
32.   . . .
```

Kode 5. 16 Fungsi *Cross Validaton*

Method cross_validate() memberi perintah folding data seperti skenario yang telah dijelaskan menggunakan perulangan sejumlah *fold* yang ditentukan (penelitian ini menggunakan 10 fold). Didalam perulangan tersebut, *method* ini memberi nilai pada variabel *train_examples* dengan data *training*, kemudian *test_examples* diisi dengan data testing. Kemudian untuk menyimpan jumlah data tiap label dalam dataset, dibuatkan variabel *train_counts* untuk *training set* dan variabel *test_counts* untuk *testing set* yang dibangun bertipe *defaultdict* seperti pada baris kode ke-21 dan 22.

```
1. def cross_validate(fold, data, embeddings, args, key
      _vector, subtask):
2.
3. . . .
4.
5.     fields = data.fields
6.
7.     train_set = torchtext.data.Dataset(examples=trai
      n_examples, fields=fields)
8.     test_set  = torchtext.data.Dataset(examples=test
      _examples, fields=fields)
9.     all_set = torchtext.data.Dataset(examples=data.e
      xamples, fields=fields)
10.
11.       text_field  = None
```

```
12.       label_field = None
13.
14.       for field_name, field_object in fields:
15.         if field_name == 'text':
16.           text_field = field_object
17.         elif field_name == 'label':
18.           label_field = field_object
19.
20.       text_field.build_vocab(train_set)
21.       label_field.build_vocab(train_set)
22.
23.       data.vocab_to_idx = dict(text_field.vocab.sto
      i)
24.       data.idx_to_vocab = {v: k for k, v in data.vo
      cab_to_idx.items()}
25.
26.       data.label_to_idx = dict(label_field.vocab.st
      oi)
27.       data.idx_to_label = {v: k for k, v in data.la
      bel_to_idx.items()}
28.
29.       embed_num = len(text_field.vocab)
30.       label_num = len(label_field.vocab)
31.   . . .
```

Kode 5.17 Fungsi Lanjutan *Cross Validation*

Kode lanjutan dari *cross_validate()* diatas bertujuan untuk membuat field yang dipergunakan sebagai wadah teks dan label sentimen. Parameter bernama data akan di-*assign* ke variabel fields yang mengalami perulangan pada baris ke-14 untuk mengisi teks dan field. Untuk menghapus kata yang berulang dari dataset, penulis memanfaatkan *method build_vocab()* seperti pada baris ke-20 dan 21, kemudian pada baris ke-23, *method dict()* dimanfaatkan untuk memberi id setiap kata yang disimpan dalam variabel *vocab_to_idx.* Variabel *idx_to_vocab* digunakan untuk mengubah kata menjadi id.

```
1. def cross_validate(fold, data, embeddings, args, key
     _vector, subtask):
```

```
2.
3. . . .
4.
5.     known_word, unknown_word, emb_init_values = data
   .create_fold_embeddings(embeddings, args, key_vect
   or)
6.     emb_init_values = np.array(emb_init_values)
7.
8.     train_iter, test_iter = torchtext.data.Iterator.
   splits((train_set, test_set),batch_sizes=(args.bat
   ch_size, len(test_set)),device=None,repeat=False,s
   huffle=False)
9.
10.      train_bulk_dataset = train_set,
11.      train_bulk__size   = len(train_set),
12.
13.      train_bulk_iter    = torchtext.data.Iterator.s
   plits(datasets=train_bulk_dataset, batch_sizes=tra
   in_bulk__size,device=None, repeat=False)[0]
14.
15.      train_bulk_iter.sort_key = lambda x: len(x.tex
   t)
16. . . .
```

Kode 5.18 Fungsi Lanjutan *Cross Validation*

Tiba saatnya *method cross_validate()* memanggil method *create_fold_embedding()* pada baris kelima yang telah dijelaskan fungsinya seperti pada sub bab 5.3. Method *create_fold_embedding()* me-*return* nilai berupa jumlah kata yang diketahui kamus *embedding*, jumlah kata yang tidak diketahui dan bobot awal dari kamus *embedding*. Kemudian baris keenam nilai bobot awal di emb_init_values diubah tipe variabelnya menjadi array menggunakan *library* Numpy. Variabel *train_iter* menyimpan dataset untuk keperluan *training* dan *test_iter* menyimpan data untuk *testing* seperti tertera pada baris kode ke-8.

```
1.  def cross_validate(fold, data, embeddings, args, k
      ey_vector, subtask):
2.
```

```
3.  . . .
4.      lstm = lstmmodel.Net(embed_num, label_num - 1,
       args.embeddings_dim, args.embeddings_mode, emb_i
      nit_values, args)
5.      kim2014 = model.CNN_Kim2014(embed_num, label_n
      um - 1,args.embeddings_dim, args.embeddings_mode,
      emb_init_values, args)
6.
7.      if args.cuda:
8.          lstm.cuda()
9.          kim2014.cuda()
```

Kode 5.19 Instantiasi Kelas Model CNN dan LSTM

Diakhir *method cross_validate*, pada baris ke-4 penulis membuat instans dari kelas model LSTM yang disimpan dalam variabel ‘lstm’ dan kelas model CNN yang disimpan dalam variabel ‘kim_2014’ pada baris ke-5. Kedua variabel tersebut memanggil *method cuda()* apabila dalam pemrosesan memilih menggunakan GPU. Setelah membuat instans dari kelas model, *cross_validate()* memanggil fungsi *train()* dan *evaluate_ens()* yang masih di dalam *for-loop* sebanyak fold. Kode dan kegunaan dari dari kedua fungsi tersebut akan dibahas dibawah ini.

```
1.  def train(model, train_iter, test_iter, label_to_i
      dx, idx_to_label, train_bulk_iter,fold,subtask):
2.
3.     parameters = filter(lambda p: p.requires_grad, m
      odel.parameters())
4.
5.     optimizer = torch.optim.Adam(parameters)
6.
7.     if args.cuda:
8.       model.cuda()
9.
10.    model.train()
11.
12.    start_epoch = 0
13.
14.    for epoch in range(1, args.epoch_num+1):
15.
```

```
16.      print("###__________FOLD/EPOCH [{}/{}]________
      __###".format(fold+1,epoch))
17.      steps = 0
18.      corrects_sum  = 0
19.      go = time.time()
20.
21.      for batch in train_iter:
22.        text_numerical, target = batch.text, batch.l
      abel
23.
24.        if args.cuda:
25.  . . .
```

Kode 5.20 Proses Training Model

Kode diatas merupakan fungsi *train()* memiliki parameter model yang akan di training, dataset *train_iter* dan *test_iter*, label sentimen, fold dan subtask. Seperti fungsi training pada umumnya, dalam perulangan epoch pada baris ke-14, sebelumnya kita akan mendefinisikan *optimizer* pada baris ketiga, jumlah hasil prediksi dan mode *model.train()* pada baris ke-10 agar library mengijinkan adanya fungsi *dropout* pada model.

```
1.   def train(model, train_iter, test_iter, label_to_i
       dx, idx_to_label, train_bulk_iter,fold,subtask):
2.
3.   . . .
4.           text_numerical, target = text_numerical.cu
      da(), target.cuda()
5.
6.         text_numerical.data.t_()
7.         target.data.sub_(1)
8.
9.         optimizer.zero_grad()
10.
11.        forward = model(text_numerical)
12.        loss = F.nll_loss(forward, target)
13.        loss.backward()
14.        optimizer.step()
15.        steps += 1
16.
```

```
17.         corrects = (torch.max(forward, 1)[1].view(ta
      rget.size()).data == target.data).sum()
18.         accuracy = 100.0 * corrects / batch.batch_si
      ze
19.
20.  . . .
```

Kode 5.21 Proses Training Model Lanjutan

Kode lanjutan dari fungsi *train()* ini memanggil fungsi *zero_grad()* pada baris kode ke-9 untuk mengatur gradien menjadi nol sebelum memulai *backpropagation* dikarenakan *library* Pytorch mengakumulasi gradien pada *backward pass*. Pada baris ke-11 variabel *forward* memanggil model untuk melakukan *forward pass* dengan masukan *text_numerical.* Perhitungan loss menggunakan NLLLoss dari *library torch.nn.Functional()*. Variabel *Optimizer.step()* pada baris ke-14 akan mengkalkulasikan gradien pada setiap loss saat *loss.backward()* dipanggil. Kemudian hasil prediksi yang benar sesuai dengan label akan dihitung pada baris ke-17 menggunakan *method torch.max()* untuk memilih label yang memiliki nilai prediksi yang paling besar diantara yang lain. Gambar dibawah ini merupakan *print out* dari variabel *forward* empat item teratas. Kemudain 'torch max' pada *print out* adalah forward yang telah diberi fungsi *torch.max()*.

```
forward :
 tensor([[-0.4381, -1.0364],
        [-1.2774, -0.3268],
        [-0.2738, -1.4290],
        [-0.7964, -0.5995]])
torch max :
 (tensor([-0.4381, -0.3268, -0.2738, -0.5995, -0.6746, -0.3721, -0.6195, -0.4705,
        -0.6416, -0.4564, -0.5273, -0.3055, -0.4290, -0.3247, -0.4054, -0.3029,
        -0.2521, -0.3750, -0.5396, -0.6437, -0.5588, -0.2916, -0.3108, -0.3743,
        -0.3126, -0.6454, -0.6256, -0.4585, -0.2975, -0.4261, -0.5022, -0.6493,
        -0.4368, -0.2996, -0.6129, -0.6731, -0.6683, -0.3840, -0.2448, -0.3573,
        -0.6686, -0.3304, -0.6907, -0.5434, -0.3288, -0.5543, -0.5104, -0.5461,
        -0.4115, -0.5042], grad_fn=<MaxBackward0>), tensor([0, 1, 0, 1, 0, 1, 0, 1, 0,
        1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0,
        1, 1]))
torch max[1] :
 tensor([0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1,
        1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0,
        1, 1])
target data :
 tensor([0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0,
        1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1,
```

Gambar 5.2 Matriks Hasil Training Model

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

. . .

    print('\n###____Performance on Test Data____###')

    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,'testing',epoch,fold,go,subtask)

  return model
```

Kode 5. 22 Pemanggilan Fungsi Testing dan Statistik

Masih dalam perulangan tahap training, *method evaluate()*, *calculate_fold_counts()* dan *display_measure()* untuk kepentingan analisis model dipanggil pada baris ketujuh, kesembilan dan sebelas. *Method evaluate()* digunakan untuk melakukan testing tiap model, baris kode seperti dibawah ini.

```
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)
```

```
11.      forward = model(text_numerical)
12.      loss = F.cross_entropy(forward, target, size_a
    verage=False)
13.      avg_loss += loss.data.item()
14.      corrects += (torch.max(forward, 1)[1].view(tar
    get.size()).data == target.data).sum()
15.    size = len(data_iter.dataset)
16.    avg_loss = avg_loss/size
17.    accuracy = 100.0 * corrects/size
18.    cor = corrects.item()
19.    acc = 100 * (cor/size)
```

Kode 5.23 Proses Testing Model

Fungsi evaluasi diawali dengan mengatur mode menjadi *model.eval()* pada baris kedua agar library tidak mengijinkan adanya fungsi *dropout* pada model. Perulangan sebanyak ukuran *batch* yang dimulai dari baris kode kelima dilakukan. Kemudian prosesnya serupa dengan method train() melakukan *forward pass* dan menghitung loss, hanya saja di *method evaluate* ini tidak melakukan *backward pass.* Untuk model ensembel, fungsi *evaluate()* dibuat tersendiri untuk menghitung *forward pass* dua model secara sekuensial. Perbedaan keduanya terlihat pada kode dibawah.

```
1.  def evaluate_ens(modela, modelb, data_iter, type,
    label_to_idx, idx_to_label,fold,subtask):
2.
3.    modela.eval()
4.    modelb.eval()
5.
6.    for epoch in range(1, args.epoch_num+1):
7.
8.  . . .
9.
10.       text_numerical.data.t_()
11.       target.data.sub_(1)
12.
13.       out_lstm = modela(text_numerical)
14.       out_cnn = modelb(text_numerical)
15.
16.       forward = out_lstm + out_cnn
17.       forward1 = forward/2
```

```
18.
19.         loss = F.cross_entropy(forward, target, size
    _average=False)
20.
21.         avg_loss += loss.data.item()
22.         steps += 1
23.         corrects += (torch.max((forward1), 1)[1].vie
    w(target.size()).data == target.data).sum()
24.
25.       size = len(data_iter.dataset)
26.       avg_loss = avg_loss/size
27.       cor = corrects.item()
28.       acc = 100 * (cor/size)
```

Kode 5.24 Proses Ansembel Model

Pada *method evaluate_ens()*, model LSTM dan CNN diberi masukan secara berurutan pada baris ke-13 dan 14. Kemudian hasil forward kedua model ditambahkan dan disimpan dalam variabel *forward_1* seperti terlihat pada baris ke-16. Kemudian pada baris ke-17 untuk mendapatkan nilai rata-rata prediksi tiap model maka dibagi 2 (disesuaikan banyak model yang digabung). Untuk menentukan hasil prediksi yang benar dari gabungan kedua model, pada baris ke-23 *method torch.max()* dipanggil dan disimpan dalam variabel corrects. Dibawah ini merupakan gambar hasil dari forward pass model LSTM dan CNN, diambil 4 data teratas.

```
out_lstm tensor([[-0.5715, -0.8316],
        [-0.0099, -4.6188],
        [-0.3314, -1.2656],
        [-0.0334, -3.4164]])
out_cnn tensor([[-1.4663, -0.2624],
        [-0.2018, -1.6997],
        [-1.3216, -0.3102],
        [-0.3038, -1.3394]])
forward tensor([[-2.0378, -1.0940],
        [-0.2117, -6.3185],
        [-1.6530, -1.5758],
        [-0.3372, -4.7558]])
forward1 tensor([[-1.0189, -0.5470],
        [-0.1059, -3.1592],
        [-0.8265, -0.7879],
        [-0.1686, -2.3779]])
torch max forward1 :  (tensor([-0.5470, -0.1059, -0.7879, -0.1686, -0.1013, -0.13
        -0.2241, -0.6008, -0.3013, -0.7795, -0.5660, -0.1085, -0.4506, -0.1481,
        -0.1906, -0.4235, -0.1675, -0.5307, -0.3843, -0.2053, -0.2539, -0.8301,
```

Gambar 5.3 Matriks Hasil Ansembel Model

Terlihat pada data ketiga *out_lstm* menyimpan nilai prediksi sentimen LSTM ke label 1, karena -0,03314 adalah nilai yang lebih besar daripada -1.2656. Berbeda halnya dengan CNN yang memprediksi sentimen ke label 2 dengan nilai item *out_cnn* urutan ketiga adalah -0.3102. Variabel forward1 merata-rata nilai prediksi kedua model dan dihasilkan nilai -0.8265 dan -0.78879. Maka keputusan akhir prediksi oleh model gabungan CNN dan LSTM adalah label 2 (bisa dilihat pada hasil output torch max forward1 item ke 3.

Halaman ini sengaja dikosongkan

BAB VI
HASIL DAN PEMBAHASAN

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

6.1 Hasil Pengujian Model CNN

6.1.1 Konfigurasi Awal Parameter CNN

Sebelum melakukan proses training, model CNN ditentukan konfigurasi model dahulu agar dapat dibandingkan hasilnya dengan jelas. Hyperparameter akan ditentukan berdasarkan penelitian sebelumnya, berikut tabel terkait konfigurasi awal.

Tabel 6.1 Konfigurasi Awal Parameter CNN

Parameter	Nilai
Folding Data	10
Epoch	20
Mini batch	50
Word Embedding Model	Word2Vec Skip-gram non static
Dropout Rate	0,5
Feature Maps	100
Pooling	1-Max Pooling
Optimizer	Adadelta
Learning Rate	1

6.1.2 Subtask A

1) Pengaruh Single Filter Region Size pada Subtask A

Skenario pertama pada model CNN untuk menguji efek perubahan nilai *region size* terhadap akurasi model yang

nantinya akan diambil region size yang memiliki nilai akurasi terbaik. Region size tersebut akan mejadi landasan multi region size. Nilai region size yang digunakan adalah 1 hingga 3. Pengukuran utama yang digunakan dalam skenario ini adalah akurasi, F-Measure, recall dan precision. Berikut hasil dari skenario pertama untuk subtask A.

Tabel 6.2 Hasil Pengujian Single Filter Region Subtask A

Region Size	Akurasi	F1	Recall	Presisi
1	96,80	96,40	96,30	96,22
2	96,67	96,52	96,96	96,40
3	96,40	96,19	96,66	96,07

Dari percobaan diatas yang berdasarkan *single filter region size* untuk subtask A, dihasilkan akurasi terbaik pada region size 1. Namun untuk pengukuran F-Measure, presisi dan recall memiliki nilai tebaik pada region ke-2. Maka dari itu, region size 1 menjadi *baseline* pertimbangan dalam menentukan ukuran *multiple filter region* dengan melakukan kombinasi angka region size.

2) Pengaruh Multi Filter Region Size pada Subtask A

Memasuki skenario kedua dengan percobaan multi filter region size, hasil kombinasi dari nilai terbaik *single filter region size* yaitu 1,1,1 dan 1,2,3. Kemudian diberi nilai dua nilai *feature maps* untuk melihat pengaruh penambahan *feature maps.*

Tabel 6.3 Hasil Pengujian Multi Filter Region Subtask A

Region Size	Feature Maps	Akurasi	F1	Recall	Presisi
1,2,3	100	96,80	96,73	97,40	96,25
1,2,3	200	96,58	96,11	97,07	97,07
1,1,1	100	96,84	96,68	97,23	96,47
1,1,1	200	96,85	96,73	97,18	96,53

Percobaan diatas berdasarkan *multi filter region size* untuk subtask A, dihasilkan akurasi terbaik pada region size 1,1,1 dengan *feature maps* 200. Selain akurasi, pengukuran F-Measure, F1 dan presisi memiliki nilai tebaik pada region yang sama. Sedangkan nilai recall tertinggi ada pada region sze 1,1,1 dengan nilai *feature_maps* 100.

6.1.3 Subtask B

Skenario yang dilakukan seperti pada subtask A juga dilakukan pada subtask B. Dimana Subtask B memiliki jumlah label yang lebih banyak, yaitu sebanyak 3 (positif, negatif dan netral).

1) Pengaruh Single Filter Region Size pada Subtask B

Tabel 6.4 Hasil Pengujian Single Filter Region Subtask B

Region Size	Akurasi	F1	Recall	Presisi
1	92,45	93,74	93,6	93,95
2	92,21	86,06	84,45	88,60
3	92,25	86,39	84,53	89,74

Dari percobaan diatas yang berdasarkan *single filter region size* untuk subtask B, dihasilkan akurasi terbaik pada region size 1, yaitu sebesar 92,45. Begitu pula dengan pengukuran F-Measure, presisi dan recall. Karena region size 1 memiliki nilai tebaik pada region yang sama, maka region size 1 menjadi *baseline* dalam menentukan ukuran *multiple filter region* dengan melakukan kombinasi angka region size.

2) Pengaruh Multi Filter Region Size pada Subtask B

Memasuki skenario kedua dengan percobaan multi filter region size, hasil kombinasi dari nilai terbaik single filter region size yaitu 1,2,3 dan 1,1,1. Kemudian diberi kombinasi dua nilai *feature maps* untuk melihat pengaruh penambahan *feature maps.*

Tabel 6.5 Hasil Pengujian Multi Filter Region Subtask B

Region Size	Feature Maps	Akurasi	F1	Recall	Presisi
1,2,3	100	92,57	93,96	95,00	93,31
1,2,3	200	92,56	94,00	95,56	92,98
1,1,1	100	92,56	93,95	96,00	92,17
1,1,1	200	92,67	93,95	95,81	92,55

Percobaan diatas berdasarkan *multi filter region size* untuk subtask B, dihasilkan akurasi terbaik pada region size 1,1,1 dengan *feature maps* 200.

3) Pengaruh Feature Maps pada Subtask B

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 karean 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 %.

6.1.4 Subtask C

Skenario yang dilakukan seperti pada subtask A diulang pada subtask B. Dimana Subtask B memiliki jumlah label yang lebih banyak, yaitu sebanyak 5 (sangat positif, positif, negatif, sangat negatif dan netral).

1) Pengaruh Single Filter Region Size pada Subtask C

Tabel 6.6 Hasil Pengujian Single Filter Region Subtask C

Region Size	Akurasi	F1	Recall	Presisi
1	89,07	98,68	98,75	98,7
2	89,08	86,42	86,53	86,53
3	88,86	85,55	86,90	83,30

Dari percobaan diatas yang berdasarkan *single filter region size* untuk subtask B, dihasilkan akurasi terbaik pada region size 2, yaitu sebesar 89,08. Namun untuk hasil pengukuran terbaik FMeasure, presisi dan recall terletak pada *region size* 1.

2) Pengaruh Multi Filter Region Size pada Subtask C

Memasuki skenario kedua dengan percobaan multi filter region size, hasil kombinasi dari nilai terbaik single filter region size yaitu 1,1,1 dan 1,2,3. Kemudian diberi nilai dua nilai feature maps untuk melihat pengaruh penambahan *feature maps.*

Tabel 6.7 Hasil Pengujian Multi Filter Region Subtask C

Region Size	Feature Maps	Akurasi	F1	Recall	Presisi
1,2,3	100	89,51	87,97	87,44	88,84
1,2,3	200	89,70	86,78	85,82	87,90
1,1,1	100	89,51	87,97	87,44	88,84
1,1,1	200	89,54	98,55	98,50	98,50

Percobaan diatas berdasarkan *multi filter region size* untuk subtask B, dihasilkan akurasi terbaik pada region size 1,2,3 dengan *feature maps* 200. Selain akurasi, pengukuran seperti nilai F1 recall dan presisi tertinggi ada pada region sze 1,1,1 dengan nilai *feature_maps* 200.

6.2 Hasil Pengujian Model LSTM

Sebelum melakukan *training*, model LSTM dilakukan konfigurasi parameter awal. Berikut merupakan parameter awal yang konfigurasinya dibuat sama untuk membandingkan akurasi model LSTM.

Tabel 6.8 Konfigurasi Parameter Awal LSTM

Parameter	Nilai
Folding Data	10

Epoch	20
Mini batch	50
Word Embedding Model	Word2Vec Skip-gram non static
Dropout Rate	0,5

6.2.1 Subtask A

Tabel 6.9 Percobaan pada Model LSTM

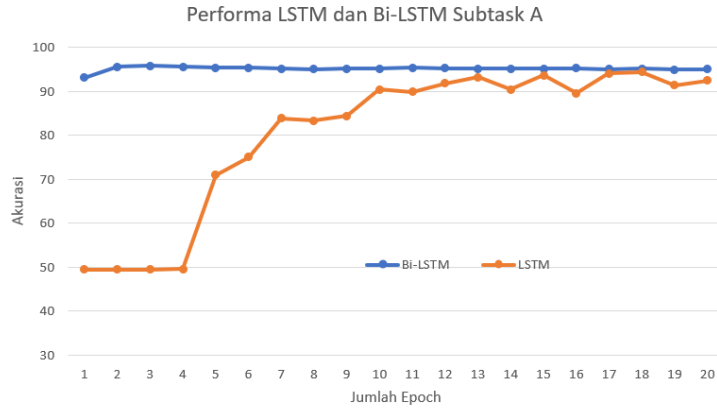
Optimizer	L. Rate	Akurasi	F1	Recall	Presisi
Adam	**0.001**	93,27	92,73	91,36	95,00
Adam	**0.01**	90,77	90,69	90,42	91,07
Adadelta	**0.01**	49,38	39,70	59,98	29,68
Adadelta	**1**	94,40	92,14	94,44	94,56

Tabel 6.10 Percobaan pada Model Bi-LSTM

Optimizer	L. Rate	Akurasi	F1	Recall	Presisi
Adam	**0.001**	94.96	94.94	94.57	95.41
Adam	**0.01**	95,26	95,2	94,5	95,97
Adadelta	**0.01**	82,05	82,02	82,22	81,89
Adadelta	**1**	95,77	95,63	95,72	95,82

Dari percobaan diatas yang divariasikan *optimizer* dan *learning rate*, baik model LSTM maupun Bi-LSTM dihasilkan akurasi terbaik pada *optimizer* Adadelta dan *learning rate* 1. Jika dilihat dari keseluruhan percobaan pada subtask A ini, Bi-LSTM konsisten memiliki performa akurasi yang lebih baik dibandingkan dengan LSTM.

Kemudian dari model terbaik LSTM dan Bi-LSTM, kita bandingkan performanya seiring bertambahnya epoch pada subtask A menggunakan grafik dibawah ini.

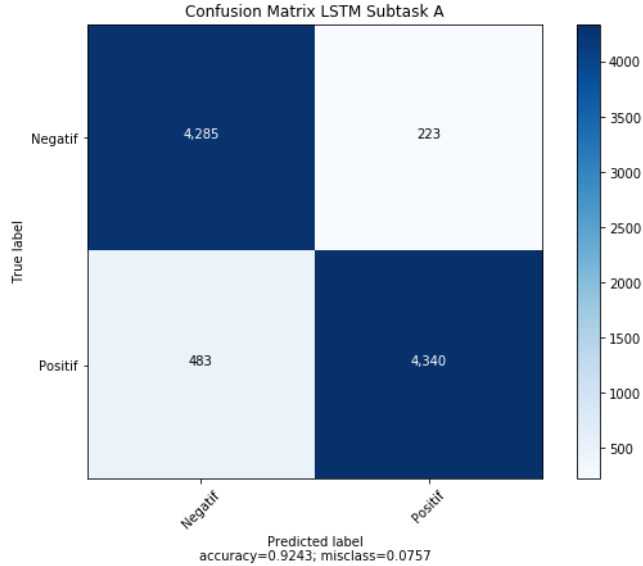


Gambar 6.1 Performa LSTM dan Bi-LSTM Subtask A

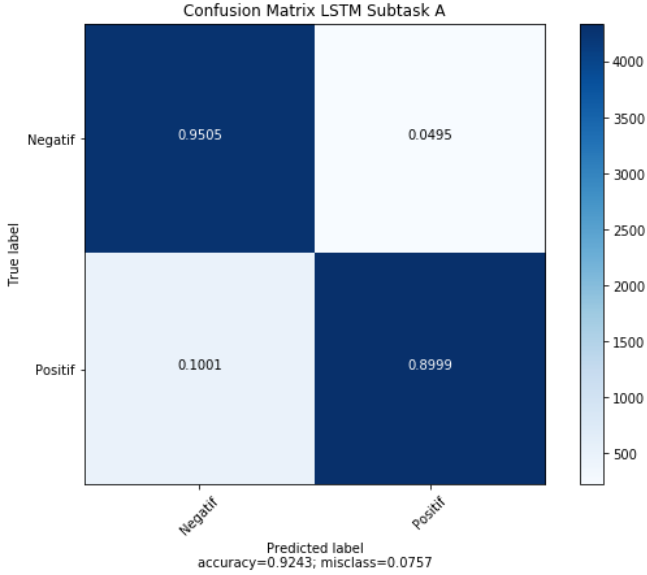
Jika dilihat dari performa yang dihasilkan berdasarkan perubahan setiap epoch didapatkan bahwa grafik Bi-LSTM terlihat lebih stabil dibandingkan dengan grafik LSTM. Untuk melihat kestabilan pada masing-masing grafik maka dilakukan perhitungan *standar deviation* pada model terbaik yang dihasilkan. Sehingga didapatkan hasil standar deviation untuk model terbaik Bi-LSTM adalah 0,513. Sedangkan untuk model terbaik LSTM menghasilkan nilai *standar deviation* sebesar 16,96. Dari hasil yang didapat nilai paling stabil dihasilkan oleh model Bi-LSTM.

- **Model LSTM**

Pada confusion matrix yang dihasilkan, kesalahan sering terjadi dalam melakukan klasifikasi untuk label positif dilihat dari perbandingan kebenaran model dalam melakukan klasifikasi terhadap labelnya.



Gambar 6.2 Confusion Matrix LSTM Subtask A

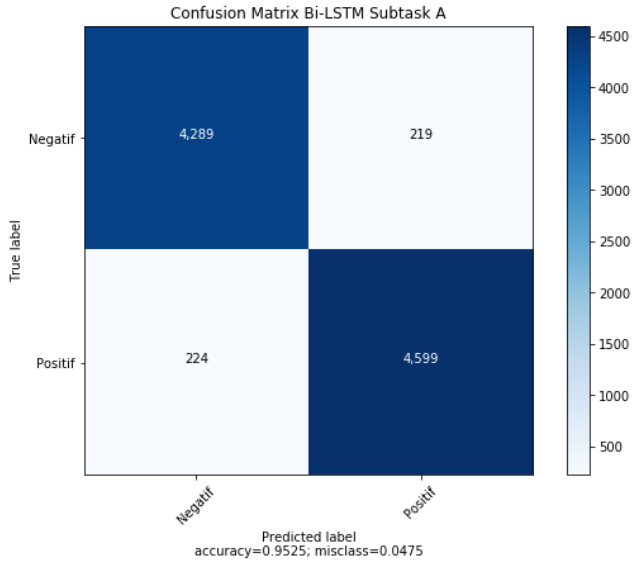


Gambar 6.3 Normalized Confusion Matrix LSTM Subtask A

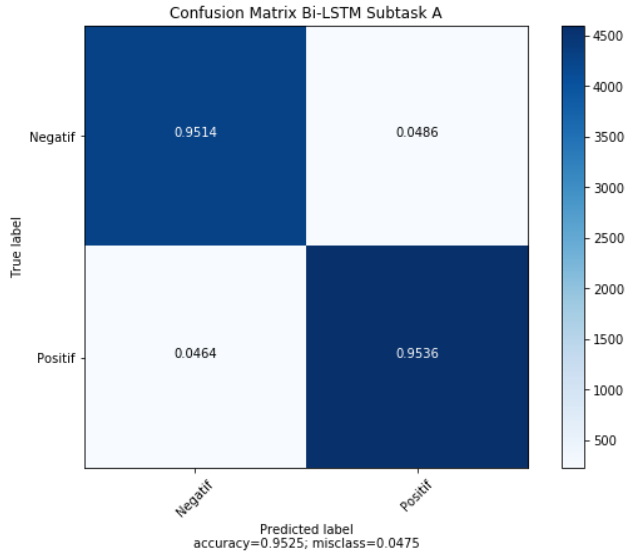
Gambar diatas menunjukkan hasil confusion matrix yang dinormalisasi untuk melihat perbandingan nilai antar tiap label dalam confusion matrix. Normalisasi dilakukan untuk menyamakan tingkat pengukuran dalam skala 0 – 1. Hasil confusion yang dinormalisasi secara jelas dapat menunjukkan bahwa proses klasifikasi label positif tepat dilakukan memiliki nilai yang lebih rendah dari label negatif secara tepat.

- **Model Bi-LSTM**

Pada confusion matrix yang dihasilkan, kesalahan dalam melakukan klasifikasi untuk label negatif dan positif sangat kecil dilihat dari perbandingan kebenaran model dalam melakukan klasifikasi terhadap labelnya.



Gambar 6.4 Confusion Matrix Bi-LSTM Subtask A



Gambar 6.5 Normalized Confusion Matrix LSTM Subtask A

Gambar diatas menunjukkan hasil confusion matrix yang dinormalisasi untuk melihat perbandingan nilai antar tiap label dalam confusion matrix. Normalisasi dilakukan untuk menyamakan tingkat pengukuran dalam skala 0 – 1. Nilai normalisasi tersebut menunjukkan proses klasifikasi label positif dan negatif tetap konsisten baik.

6.2.2 Subtask B

Tabel Percobaan pada Model LSTM

Tabel 6.11 Percobaan pada Model LSTM Subtask B

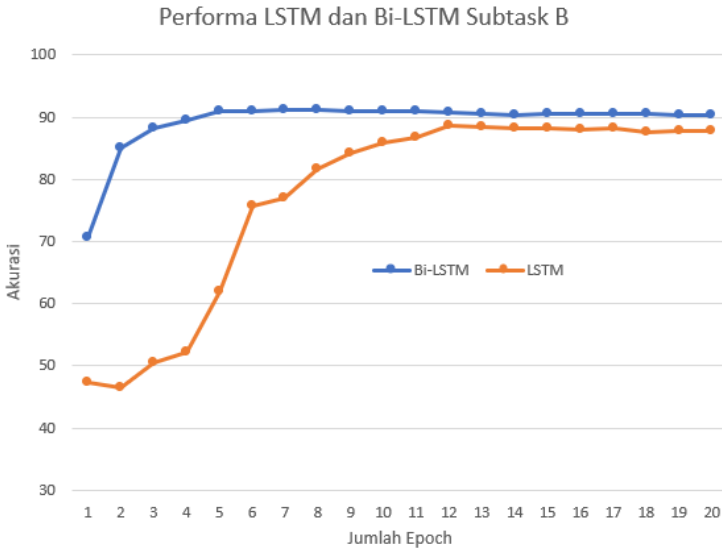
Optimizer	L. Rate	Akurasi	F1	Recall	Presisi
Adam	**0.001**	88,67	72,46	92,23	90,00
Adam	**0.01**	88,49	79,20	76,37	85,64
Adadelta	**0.01**	39,49	39,70	69,98	27,73
Adadelta	**1**	73,19	81,02	86,53	77,39

Tabel 6.12 Percobaan pada Model Bi-LSTM Subtask B

Optimizer	L. Rate	Akurasi	F1	Recall	Presisi
Adam	**0.001**	90,17	89,79	91,23	92,15
Adam	**0.01**	90,96	84,50	83,9	85,5
Adadelta	**0.01**	72,30	72,18	73,05	71,44
Adadelta	**1**	91,25	92,22	95,78	90,6

Dari percobaan pada subtask B diatas yang divariasikan *optimizer* dan *learning rate*, untuk model LSTM dihasilkan akurasi terbaik pada oprimizer Adam dan *learning rate* 0.001 yaitu sebesar 88,67%. Sedangkan untuk model Bi-LSTM dihasilkan akurasi terbaik pada *optimizer* Adadelta dengan *learning rate* 1 yaitu sebesar 91,25%. Jika dilihat dari keseluruhan percobaan pada subtask B ini, Bi-LSTM konsisten memiliki performa akurasi yang lebih baik dibandingkan dengan LSTM.

Kemudian dari model terbaik LSTM dan Bi-LSTM, kita bandingkan performanya seiring bertambahnya epoch pada subtask B menggunakan grafik dibawah ini.



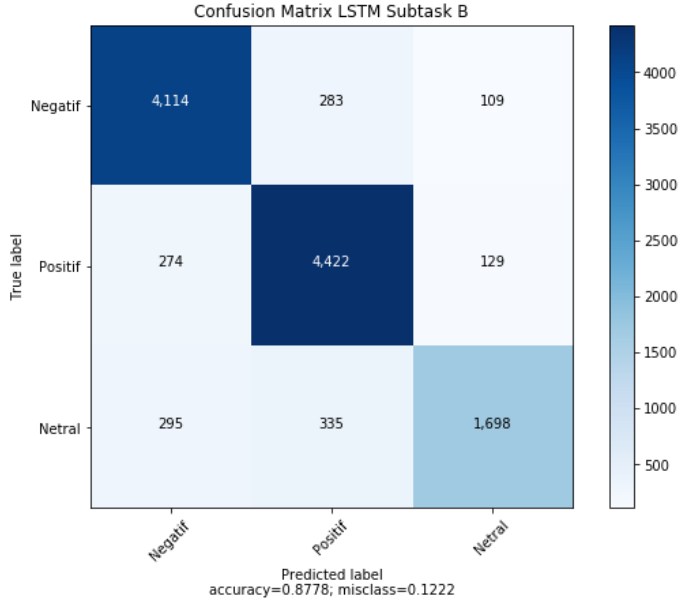
Gambar 6.6 Performa LSTM dan Bi-LSTM Subtask B

Jika dilihat dari performa yang dihasilkan berdasarkan perubahan setiap epoch didapatkan bahwa grafik Bi-LSTM terlihat lebih stabil dibandingkan dengan grafik LSTM. Bi-LSTM tidak membutuhkan waktu yang lama untuk melakukan konvergensi. Untuk melihat kestabilan pada masing-masing grafik maka dilakukan perhitungan *standar deviation* pada model terbaik yang dihasilkan. Sehingga didapatkan hasil standar deviation untuk model terbaik Bi-LSTM adalah 4,58. Sedangkan untuk model terbaik LSTM menghasilkan nilai *standar deviation* sebesar 15,74. Dari hasil yang didapat nilai paling stabil dihasilkan oleh model Bi-LSTM.

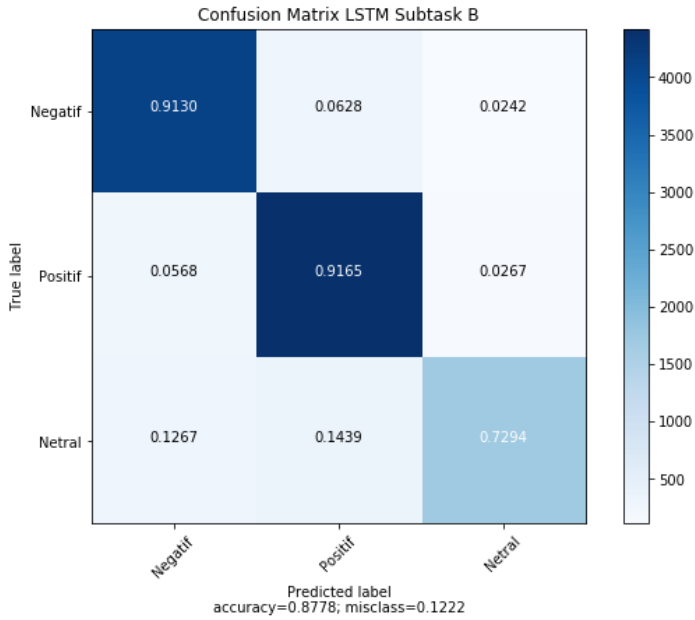
- **Model LSTM**

Pada confusion matrix yang dihasilkan, kesalahan sering terjadi dalam melakukan klasifikasi untuk label netral dilihat dari perbandingan kebenaran model dalam melakukan klasifikasi terhadap labelnya, dimana data netral lebih banyak

diklasifikasikan menjadi label positif. Hal ini dapat dipahami karena perbedaan data antara label tersebut tidak terlalu jauh.



Gambar 6.7 Confusion Matrix LSTM Subtask B

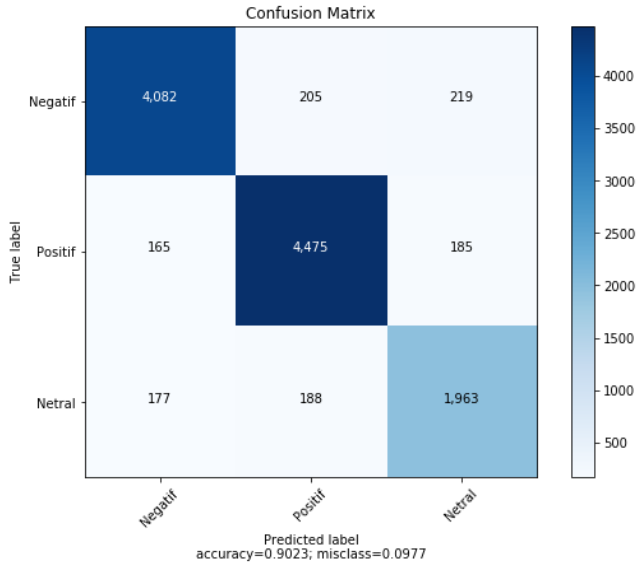


Gambar 6.8 Normalized Confusion Matrix LSTM Subtask B

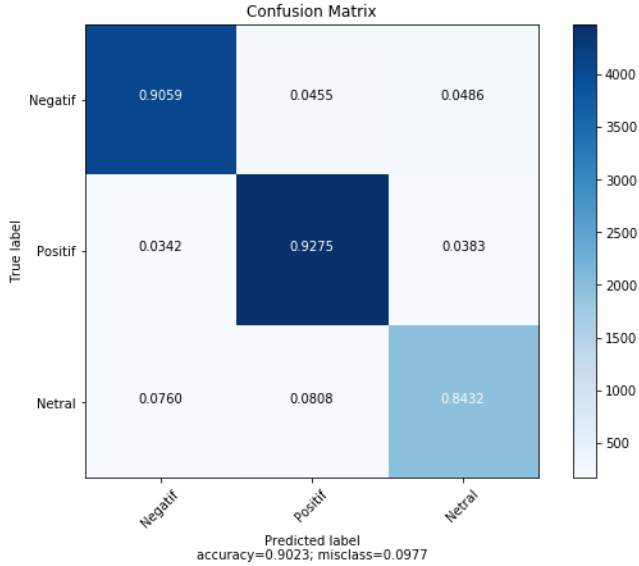
Hasil yang didapatkan perbandingan dalam skala yang sama menunjukkan proses klasifikasi pada label netral masih menghasilkan proses klasifikasi yang kurang baik dimana sebesar 0.1439 data yang memiliki label netral diprediksi menjadi label positif, sedangkan yang berhasil melakukan proses klasifikasi hanya sebesar 0.73.

- **Model Bi-LSTM**

Pada confusion matrix yang dihasilkan oleh model Bi-LSTM pada subtask B, kesalahan sering terjadi dalam melakukan klasifikasi untuk label negatif dilihat dari perbandingan kebenaran model dalam melakukan klasifikasi label.



Gambar 6.9 Confusion Matrix Bi-LSTM Subtask B



Gambar 6.10 Normalized Confusion MatrixBi- LSTM Subtask B

Hasil yang didapatkan perbandingan dalam skala yang sama menunjukkan proses klasifikasi pada label netral masih menghasilkan proses klasifikasi yang kurang baik dimana sebesar data 0,08 yang memiliki label netral diprediksi menjadi label positif, sedangkan yang berhasil melakukan proses klasifikasi sebesar 0,843, lebih baik jika dibandingkan dengan model LSTM.

6.2.3 Subtask C

Tabel 6.13 Percobaan pada Model LSTM

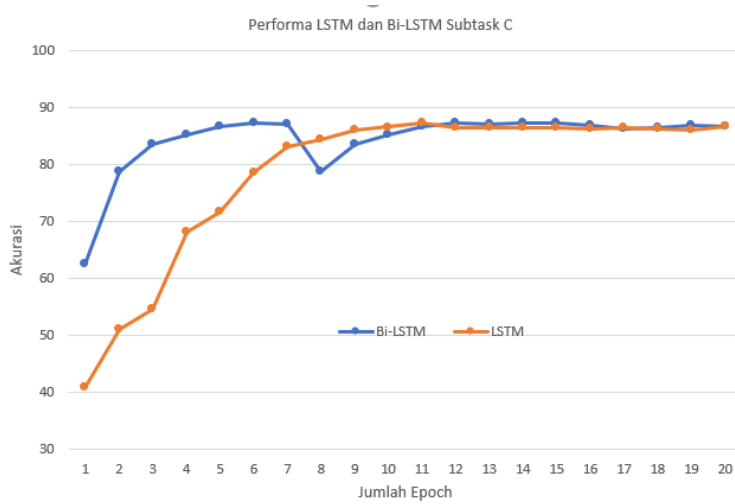
Optimizer	L. Rate	Akurasi	F1	Recall	Presisi
Adam	**0.001**	82,40	95,77	98,23	96,05
Adam	**0.01**	81,12	72,77	79,22	71,13
Adadelta	**0.01**	19,91	3,39	0	0
Adadelta	**1**	87,24	78	86,63	80,35

Tabel 6.14 Percobaan pada Model Bi-LSTM

Optimizer	L. Rate	Akurasi	F1	Recall	Presisi
Adam	**0.001**	86,12	82,28	78,66	87,28
Adam	**0.01**	86,74	97,70	98,52	96,93
Adadelta	**0.01**	48,27	37,34	34,70	41,39
Adadelta	**1**	87,19	84,5	87,09	83,34

Dari percobaan diatas yang berdasarkan *optimizer* dan *learning rate*, untuk subtask C, dihasilkan akurasi terbaik pada oprimizer Adadelta dan *learning rate* 1 yaitu sebesar 87,24 untuk LSTM, dimana lebih besar dari Bi-LSTM yaitu 87,19. Untuk model LSTM, pengukuran FMeasure, recall dan presisi menunjukkan nilai terbaik pada optimizer Adam dan *learning rate* 0.001. Kemudian dari model terbaik LSTM dan Bi-LSTM,

kita bandingkan performanya seiring bertambahnya epoch pada subtask C menggunakan grafik dibawah ini.

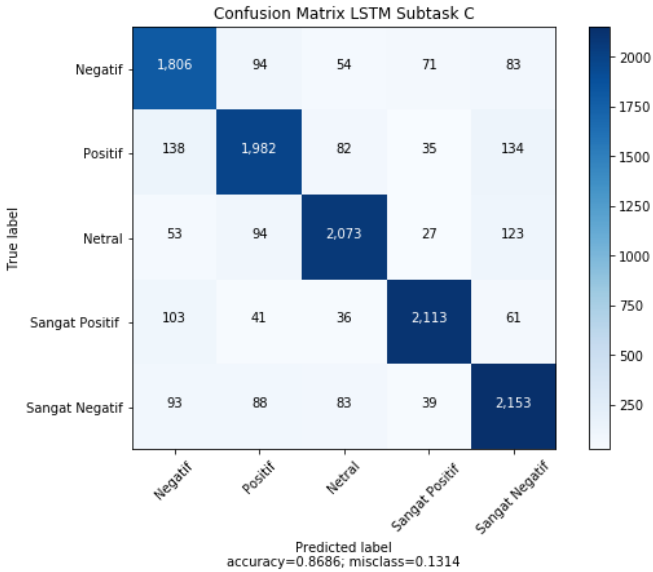


Gambar 6.11 Performa LSTM dan Bi-LSTM Subtask C

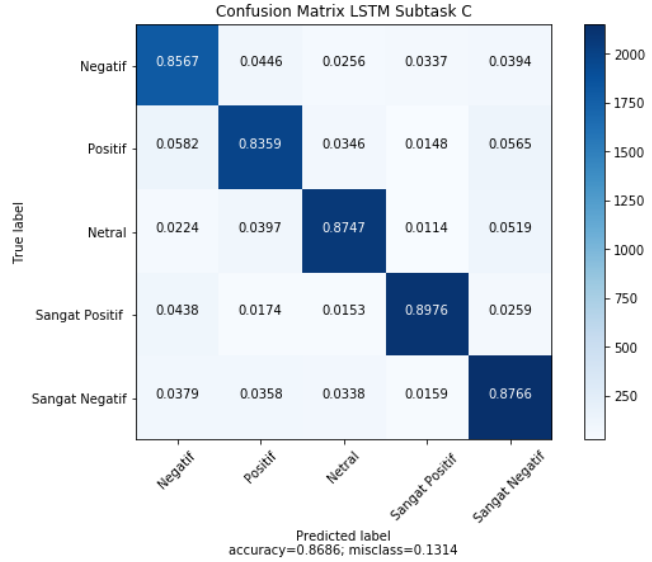
Jika dilihat dari performa yang dilakukan berdasarkan perubahan setiap epoch didapatkan bahwa grafik Bi-LSTM terlihat lebih stabil dibandingkan dengan grafik LSTM. Bi-LSTM tidak membutuhkan waktu yang lama untuk melakukan konvergensi. Untuk melihat kestabilan pada masing-masing grafik maka dilakukan perhitungan *standar deviation* pada model terbaik yang dihasilkan. Sehingga didapatkan hasil standar deviation untuk model terbaik Bi-LSTM adalah 5,72. Sedangkan untuk model terbaik LSTM menghasilkan nilai *standar deviation* sebesar 13,99. Dari hasil yang didapat nilai paling stabil dihasilkan oleh model Bi-LSTM.

- **Model LSTM**

Pada confusion matrix yang dihasilkan, kesalahan sering terjadi dalam melakukan klasifikasi untuk label netral dilihat dari perbandingan kebenaran model dalam melakukan klasifikasi terhadap labelnya, dimana data netral lebih banyak diklasifikasikan menjadi label positif. Hal ini dapat dipahami karena perbedaan data antara label tersebut tidak terlalu jauh.



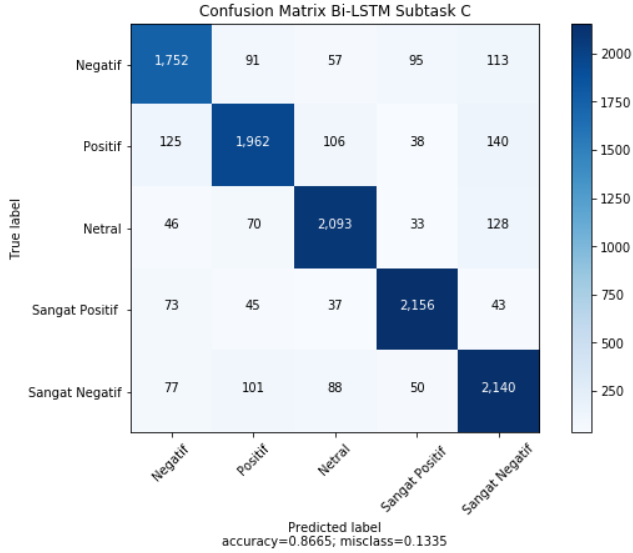
Gambar 6.12 Confusion Matrix LSTM Subtask C



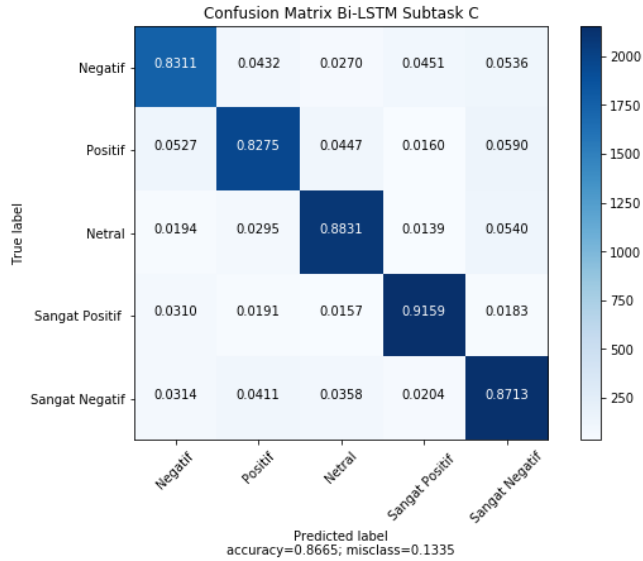
Gambar 6.13 Normalized Confusion Matrix LSTM Subtask C

- **Model Bi-LSTM**

Pada confusion matrix yang dihasilkan, kesalahan dalam melakukan klasifikasi tiap label relatif kecil dilihat dari perbandingan kebenaran model dalam melakukan klasifikasi terhadap labelnya.



Gambar 6.14 Confusion Matrix Bi-LSTM Subtask C



Gambar 6.15 Normalized Confusion Matrix Bi-LSTM Subtask C

6.3 Hasil Pengujian Model Ensembel

6.3.1 Subtask A

Tabel 6.15 Pengujian Ensembel Subtask A

Model	Akurasi
LSTM	95,77
CNN	96,85
Ensembel	96,22

Saat melakukan pengujian subtask ini, model LSTM dan CNN di training terlebih dahulu menggunakan *optimizer* Adadelta dengan *learning rate* 1. Untuk CNN mengunakan feature num 200 dan multi filter region size 1,1,1. Dapat dilihat bahwa performa ensembel belum mampu mengungguli model CNN pada subtask A.

6.3.2 Subtask B

Tabel 6.16 Pengujian Ensembel Subtask B

Model	Akurasi
LSTM	91,25
CNN	92,57
Ensembel	91,89

Saat melakukan pengujian subtask ini, model LSTM dan CNN di training terlebih dahulu menggunakan *optimizer* Adadelta dengan *learning rate* 1. Untuk CNN mengunakan feature num 100 dan multi filter region size 1,2,3. Dapat dilihat bahwa performa ensembel belum mampu mengungguli model CNN pada subtask B.

6.3.3 Subtask C

Tabel 6.17 Pengujian Ensembel Subtask C

Model	Akurasi
LSTM	*87,24*
CNN	89,7
Ensembel	89,22

Saat melakukan pengujian subtask ini, model LSTM dan CNN di training terlebih dahulu menggunakan *optimizer* Adadelta dengan *learning rate* 1. Untuk CNN mengunakan feature num 200 dan multi filter region size 1,2,3. Dapat dilihat bahwa performa ensembel belum mampu mengungguli model CNN pada subtask C.

Dari tabel 6.15, 6.16 dan 6.17, dapat dilihat bahwa penggabungan model ensembel performanya belum bisa lebih baik daripada model individu (CNN atau LSTM). Hal ini dikarenakan masing-masing model CNN maupun LSTM yang berpartisipasi pada model ensembel hanya 1 tiap model.

Sehingga pemanfaatan kemampuan CNN dalam menangkap fitur lokal yang ada dalam sebuah kalimat dan LSTM dalam menangkap hubungan dalam rangkaian kata didalam dataset yang digunakan menjadi kurang optimal. Kemudian juga dikarenakan dataset yang digunakan memiliki banyak fitur lokal berupa kata unik yang hadir pada label tertentu, membuat CNN tidak bisa terkecoh membuat prediksi secara benar pada label tertentu.

6.4 Perbandingan Kecepatan Pemrosesan CNN dan LSTM

Tabel 6.18 Perbandingan Rata-rata Kecepatan Pemrosesan CNN, LSTM dan Bi-LSTM

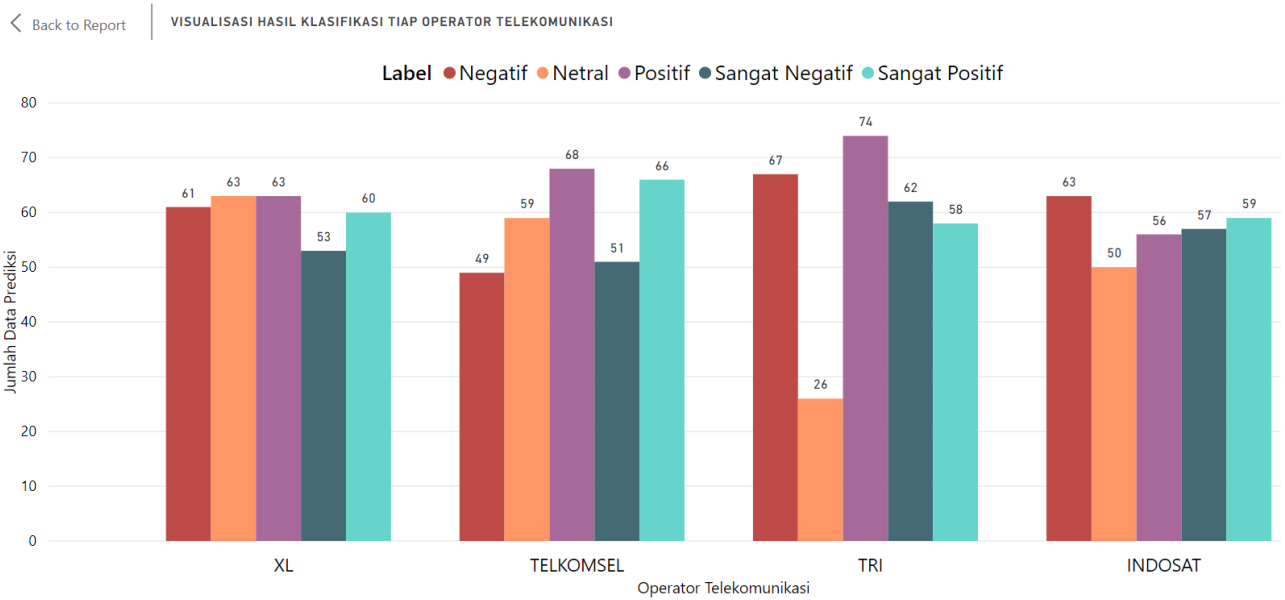
Subtask	CNN	LSTM	Bi-LSTM
A	259,5	303	399,5
B	374	389,75	556,8
C	400,3	415,5	559

Dapat dilihat dari tabel diatas bahwa disemua subtask, model CNN melakukan proses klasifikasi dengan kecepatan waktu yang paling singkat dibandingkan dua model lainnya. Pada subtask A hanya membutuhkan waktu 259,5 detik dibandingkan dengan LSTM yang membutuhkan waktu 303 detik dan Bi-LSTM membutuhkan waktu 399,5 detik. Kemudian disusul oleh LSTM kemudian Bi-LSTM yang membutuhkan waktu yang terlama. Hal ini dapat dipahami karena dataset yang digunakan memiliki rangkaian kata yang cukup panjang dalam satu kalimat hingga puluhan kata, LSTM hanya memproses satu arah sedangkan Bi-LSTM memproses klasifikasi dua arah.

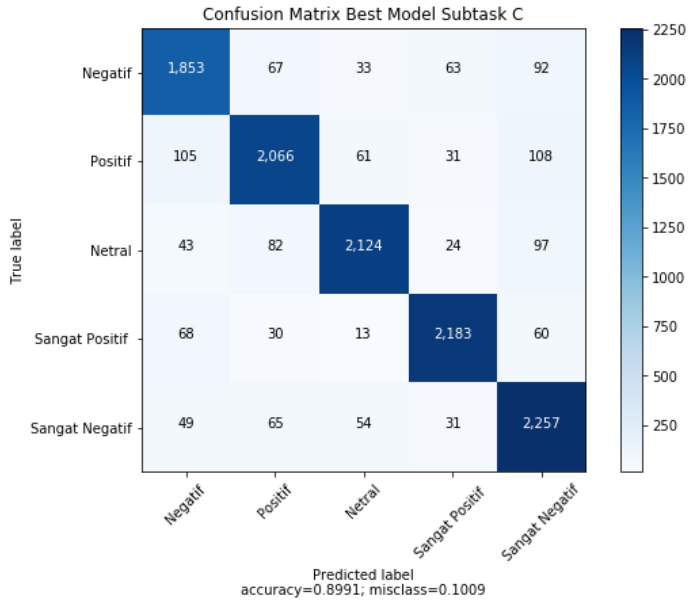
6.5 Hasil Klasifikasi Sentimen tiap Operator Telekomunikasi

Pada subbab ini, model yang memiliki performa paling baik pada subtask C digunakan untuk menghasilkan prediksi terhadap 1165 tweet. Hasil klasifikasi sentimen terhadap dataset

topik operator telekomunikasi direpresentasikan pada gambar 6.16 memperlihatkan sebaran sentimen terhadap operator telekomunikasi cukup seimbang. Namun hal tersebut berbeda untuk operator Tri, terlihat pada diagram memiliki sentimen netral yang paling sedikit serta sentimen negatif, sangat negatif dan positif yang paling banyak dibandingkan operator lain. Kemudian untuk sentimen sangat positif paling banyak diraih oleh operator Telkomsel.



Gambar 6.16 Hasil Klasifikasi Sentimen Tiap Operator Telekomunikasi



Gambar 6.17 Confusion Matrix Best Model Subtask C

Dari confusion matrix diatas, dapat dilihat bahwa label yang paling sering mengalami kesalahan adalah label positif yang seringkali diklasifikasikan sebagai label negatif dan sangat negatif. Berikut merupakan beberapa sampel *tweet* yang salah diprediksi berdasarkan label aslinya.

Tabel 6.19 Sampel Tweet yang Tidak Akurat

No	Tweet	Label Asli	Label Prediksi
1.	SEKALINYA ADA MASA EDGE! SAMPAHHHHHHH SAMPAH SAMPAAAHHHH!!! KECEWA GUE UDH 10TAHUN PAKE NI KARTU!!! @triindonesia	Positif	Sangat Negatif

2.	Membalas @KarinRiswanda Kui mending telkomsel iki marai nangis Terjemahkan Tweet	Positif	Sangat Negatif
3.	sama bro, kemarin2 pake enak lancar mangkanya gw ganti ke tsel lah kok skrng lemot gini buat main ml dan sosmed #telkomsel Bagus Sebastiawan menambahkan, Sigit Adiputra Hp @sigitadiputrahp Parah sinyal @Telkomsel buat main mobile legend, lag parah gue kena report mulu credit score gue berkurang	Positif	Negatif
4.	Membalas @navillairawan @myXL @myXLCare ga stabil vil. Meskipun sinyal 4G penuh tapi bisa kadang kenceng kadang lambat tergantung lokasi, ampun dah. Tapi 2 hari terakhir ini stlh gw mention gt, skrg agak mendingan sih haha	Positif	Negatif
5.	Membalas @triindonesia Woy goblok ini keluhan udah banyak lo masih ga ada tanggapan juga? Goblok lu. Jaringan ampas. Dasar goblok.	Positif	Negatif

Dari tabel diatas dapat diketahui dari 5 sampel tweet hasil prediksi, 3 sampel memiliki label asli yang kurang tepat, yaitu pada *tweet* nomor 1, 3 dan 5. Hal ini bisa disebabkan karena kurangnya ketelitian pelabel untuk mengklasifikasikan tweet. Kemudian untuk tweet nomor 2 diprediksikan salah karena tweet bukan berbahasa Indonesia melainkan Bahasa daerah. Untuk *tweet* nomor 4 hasil prediksi menunjukkan label negatif bisa disebabkan karena banyaknya fitur kata bernuansa

negatif seperti “ga stabil” dan “lambat” sehingga model Convolutional Neural Network lebih cenderung mengkalkulasikan ke sentimen negatif.

BAB VII
KESIMPULAN DAN SARAN

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

7.1 Kesimpulan

Dari proses penelitian analisis sentimen teks Bahasa Indonesia pada media sosial *twitter*, didapat kesimpulan-kesimpulan sebagai berikut:

1. Penggunaan parameter *multi filter region size* pada model Convolutional Neural Network menghasilkan performa model yang lebih baik daripada *single filter region size*.
2. Pada dataset yang digunakan dalam penelitian ini, Bi-LSTM lebih baik dalam melakukan proses klasifikasi dibandingkan dengan LSTM pada subtask A dan B.
3. Adadelta dengan *learning rate* 1 merupakan optimizer paling baik untuk training menggunakan model Bi-LSTM semua subtask (A, B, C). Sedangkan untuk model LSTM optimizer terbaik adalah Adadelta dengan *learning rate* 1 untuk subtask A dan C dan Adam dengan *learning rate 0,001* untuk subtask B.
4. Convolutional Neural Network lebih baik dalam melakukan proses klasifikasi dibandingkan Long Short Term Memory pada semua subtask (A, B, C).
5. Penggabungan model Convolutional Neural Network dan Long Short Term Memory tidak dapat mengungguli performa model individu dalam melakukan proses klasifikasi.

7.2 Saran

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

1. Melakukan penambahan menggunakan kombinasi *multi filter region size* yang lebih variatif mendapatkan untuk dilakukan training pengujian pada model CNN
2. Melakukan penambahan variasi *optimizer* dan *learning rate* pada model LSTM, seperti Momentum dan SGD untuk dilakukan training pengujian
3. Menambah jumlah model yang digabungkan, jika pada penelitian ini menggunakan 1 buah model CNN dan 1 buah model LSTM, ditingkatkan jumlahnya kelipatan 2, 4, 5 dan seterusnya dengan parameter yang berbeda-beda tiap model.

DAFTAR PUSTAKA

[1] APJII, "Penetrasi & Perilaku Pengguna Internet Indonesia 2017," *Asos. Penyelenggara Jasa Internet Indones.*, pp. 1–39, 2017.

[2] Herman, "Indonesia Masuk Lima Besar Pengguna Twitter," 2017. [Online]. Available: https://www.beritasatu.com/digital-life/428591-indonesia-masuk-llima-besar-pengguna-twitter.html. [Accessed: 26-Feb-2019].

[3] S. Hashimoto, "7 Benefits of Sentiment Analysis You Can't Overlook," 2017. [Online]. Available: https://blog.insightsatlas.com/7-benefits-of-sentiment-analysis-you-cant-overlook. [Accessed: 27-Feb-2019].

[4] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A Neural Probabilistic Language Model," in *Journal of Machine Learning Research*, 2003.

[5] "What is Machine Learning? A definition," 2017. [Online]. Available: https://www.expertsystem.com/machine-learning-definition/.

[6] Y. Li and T. Yang, "Word Embedding for Understanding Natural Language: A Survey," vol. 26, 2017.

[7] M. Cliche, "BB_twtr at SemEval-2017 Task 4: Twitter Sentiment Analysis with CNNs and LSTMs," no. 2014, 2017.

[8] F. A. Pozzi, E. Fersini, E. Messina, and B. Liu, *Sentiment Analysis in Social Networks*, 1st ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2016.

[9] B. E. King and K. Reinold, "Natural language processing," *Find. Concept, Not Just Word*, vol. 1, no. 4, pp. 67–78, 2014.

[10] A. Copestake, "Natural Language Processing Lecture

Synopsis," pp. 2003–2004, 2004.

[11] J. Turian, L. Ratinov, and Y. Bengio, "Word Representations: A Simple and General Method for Semi-supervised Learning," in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, 2010, pp. 384–394.

[12] T. Mikolov, G. Corrado, K. Chen, and J. Dean, "Vector Space," pp. 1–12.

[13] V. Qazvinian, E. Rosengren, D. R. Radev, and Q. Mei, "Rumor has it : Identifying Misinformation in Microblogs," *Conf. Empir. Methods Nat. Lang. Process.*, pp. 1589–1599, 2011.

[14] G. Pant, P. Srinivasan, and F. Menczer, "Crawling the Web," in *Web Dynamics: Adapting to Change in Content, Size, Topology and Use*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 153–177.

[15] J. Zhang, Y. Li, J. Tian, and T. Li, "LSTM-CNN Hybrid Model for Text Classification," 2018, pp. 1675–1680.

[16] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[17] I. W. S. E. Putra, "Klasifikasi Citra Menggunakan Convolutional Neural Network (Cnn) Pada Caltech 101," *J. Tek. ITS*, vol. 5, no. 1, p. 76, 2016.

[18] Y. Kim, "Convolutional Neural Networks for Sentence Classification," *Proc. 2014 Conf. Empir. Methods Nat. Lang. Process.*, 2014.

[19] A. Dertat, "Applied Deep Learning - Part 4: Convolutional Neural Networks," 2017. [Online]. Available: https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2. [Accessed: 24-Feb-2019].

[20] J. Wu, "Introduction to convolutional neural networks,"

Natl. Key Lab Nov. Softw. Technol., pp. 1–31, 2017.

LAMPIRAN A

Hasil Pengujian Model LSTM dan Bi-LSTM Terbaik Tiap Epoch

epoch	A		B		C	
	Bi-LSTM	LSTM	Bi lstm	LSTM	Bi-LSTM	LSTM
1	93,152	49,520	70,684	47,267	62,570	40,780
2	95,648	49,520	85,076	46,547	78,798	51,033
3	95,767	49,520	88,129	50,501	83,583	54,566
4	95,585	49,550	89,467	52,165	85,273	68,016
5	95,350	70,990	90,882	61,934	86,697	71,747
6	95,338	75,050	90,934	75,667	87,177	78,472
7	95,199	83,900	91,191	76,963	86,971	83,198
8	95,060	83,380	91,251	81,586	78,798	84,398
9	95,146	84,440	91,037	84,236	83,583	85,968
10	95,189	90,390	90,891	85,814	85,273	86,542
11	95,370	89,930	90,943	86,637	86,697	87,237
12	95,250	91,790	90,771	88,669	87,177	86,482
13	95,167	93,230	90,462	88,438	86,975	86,379
14	95,145	90,390	90,402	88,181	87,194	86,414
15	95,135	93,630	90,436	88,121	87,186	86,508
16	95,253	89,590	90,505	88,077	86,869	86,302
17	95,070	94,140	90,513	88,112	86,182	86,465
18	95,124	94,410	90,488	87,494	86,508	86,233
19	94,985	91,440	90,368	87,734	86,877	86,053
20	95,090	92,440	90,230	87,778	86,705	86,619

Hasil Pengukuran Performa Model LSTM

Subtask	**Tipe**	**Optimizer**	**L. Rate**	**Nama File (SCENE_)**
A	**LSTM**	Adam	0.001	4ATV5
A	**LSTM**	Adam	0.01	DOAMQ

A	**LSTM**	Adadelta	0.01	PVBFD
A	**LSTM**	Adadelta	1	YZYYS
A	**Bi-LSTM**	Adam	0.001	XQV3W
A	**Bi-LSTM**	Adam	0.01	1IJCN
A	**Bi-LSTM**	Adadelta	0.01	3RLCL
A	**Bi-LSTM**	Adadelta	1	M3QUG
B	**LSTM**	Adam	0.001	6I53P
B	**LSTM**	Adam	0.01	MZMBQ
B	**LSTM**	Adadelta	0.01	HISSB
B	**LSTM**	Adadelta	1	X4BTS
B	**Bi-LSTM**	Adam	0.001	4AM9N
B	**Bi-LSTM**	Adam	0.01	8Y485
B	**Bi-LSTM**	Adadelta	0.01	SZ653
B	**Bi-LSTM**	Adadelta	1	AZDF5
C	**LSTM**	Adam	0.001	J3IDG
C	**LSTM**	Adam	0.01	31S31
C	**LSTM**	Adadelta	0.01	0Z0MT
C	**LSTM**	Adadelta	1	06W15
C	**Bi-LSTM**	Adam	0.001	KM57N
C	**Bi-LSTM**	Adam	0.01	S5VLA
C	**Bi-LSTM**	Adadelta	0.01	XFR1L

C	**Bi-LSTM**	Adadelta	1	WW5F1

Hasil Pengukuran Performa Model CNN

Subtask	**Region Size**	**Feature Maps**	**Nama File (SCENE_)**
A	1	100	XQTPU
A	2	100	PKR61
A	3	100	O9M7B
A	1,2,3	100	5PP0V
A	1,2,3	200	IX6RA
A	1,1,1	200	7G57X
A	1,1,1	100	LFE3D
B	1	100	VO8XJ
B	2	100	84VR2
B	3	100	C2H19
B	1,2,3	100	7WX9X
B	1,2,3	200	BDZE7
B	1,1,1	200	0SLPL
B	1,1,1	100	PA8FO
C	1	100	CHUD
C	2	100	OA039
C	3	100	PQC6X

C	1,2,3	200	CALEG
C	1,1,1	200	TJJBE
C	1,1,1	100	QDJBO

File tersebut bisa diakses pada link https://drive.google.com/open?id=1OS0tvobYwWAxqYuHJqKdQqvy4CM7-7Lv atau *http://bit.do/lampiranTAEvia*.

Halaman ini sengaja dikosongkan

BIODATA PENULIS

Penulis lahir di Kediri pada tanggal 26 Januari 1998. Merupakan anak pertama dari 3 bersaudara. Penulis telah menempuh beberapa pendidikan formal yaitu; SD Negeri Puhsarang I, SMP Negeri 8 Kediri dan SMA Negeri 1 Kediri.

Pada Tahun 2015 pasca kelulusan SMA, penulis melanjutkan pendidikan di Jurusan Sistem Informasi Fakultas Teknologi Informasi dan Komunikasi – Institut Teknologi Sepuluh Nopember (ITS) Surabaya dan terdaftar sebagai mahasiswa dengan NRP 05211540000056. Selama menjadi mahasiswa, penulis mengikuti berbagai kegiatan kemahasiswaan seperti beberapa kepanitian salah satunya Information System Expo sebagai Staff Publication and Roadshow serta pernah menjabat sebagai Ketua Panitia LKMW TD 2016 FTIF. Pada tahun kedua pernah menjabat sebagai Staff Kewirausahaan HMSI FTIK ITS. Di bidang akademik, penulis pernah menjadi asisten dosen praktikum pada mata kuliah Sistem Enterprise. Selain itu, pada tahun 2017 penulis menjadi Top 20 Finalis pada perlombaan skala nasional Business and Social IT Case FTIF Festifal.

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 eviananda98@gmail.com