

Universitas Sumatera Utara

Repositori Institusi USU

<http://repositori.usu.ac.id>

Departemen Teknologi Informasi

Tesis Magister

2020

Kinerja Deep Learning dalam Analisis Sentimen

Manalu, Boy Utomo

Universitas Sumatera Utara

<http://repositori.usu.ac.id/handle/123456789/28796>

Downloaded from Repositori Institusi USU, Universitas Sumatera Utara

KINERJA *DEEP LEARNING* DALAM ANALISIS SENTIMEN

TESIS

BOY UTOMO MANALU

157038086



**PROGRAM STUDI S2 TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA
MEDAN
2020**

KINERJA DEEP LEARNING DALAM ANALISIS SENTIMEN

TESIS

Diajukan untuk melengkapi tugas dan memenuhi syarat memperoleh ijazah Magister
Teknik Informatika

BOY UTOMO MANALU

157038086



**PROGRAM STUDI S2 TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA
MEDAN
2020**

PERSETUJUAN

Judul : KINERJA DEEP LEARNING DALAM ANALISIS
SENTIMEN
Kategori : TESIS
Nama : BOY UTOMO MANALU
Nomor Induk Mahasiswa : 157038086
Program Studi : MAGISTER (S2) TEKNIK INFORMATIKA
Fakultas : ILMU KOMPUTER DAN TEKNOLOGI
INFORMASI
UNIVERSITAS SUMATERA UTARA
Komisi Pembimbing :

Pembimbing II

Pembimbing I



Dr. Syahril Efendi, S.Si, M.IT

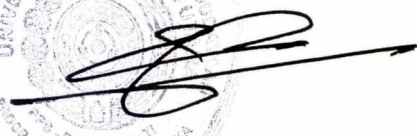


Prof. Dr. Tulus, Vor. Dipl. Math., M.Si.

Diketahui/disetujui oleh

Program Studi Magister (S2) Teknik Informatika

Ketua,


Prof. Dr. Muhammad Zarlis

NIP. 19570701 198601 1 003

PERYATAAN ORISINILITAS**KINERJA DEEP LEARNING DALAM ANALISIS SENTIMEN****TESIS**

Saya mengakui bahwa tesis ini adalah hasil karya saya sendiri, kecuali beberapa kutipan dan ringkasan yang masing-masing telah disebutkan sumbernya.

Medan, 7 September 2020



Boy Utomo Manalu

157038086

**PERNYATAAN PERSETUJUAN PUBLIKASI KARYA ILMIAH
UNTUK KEPENTINGAN AKADEMIS**

Sebagai sivitas akademika Universitas Sumatera Utara, saya yang bertandatangan di bawah ini:

Nama : Boy Utomo Manalu
NIM : 157038086
Program Studi : Magister (S2) Teknik Informatika
Jenis Karya Ilmiah : Tesis

Demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Sumatera Utara Hak Bebas Royalti Non-Eksklusif (Non-Exclusive Royalty Free Right) atas tesis saya yang berjudul:

KINERJA DEEP LEARNING DALAM ANALISIS SENTIMEN

Beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Non-Eksklusif ini, Universitas Sumatera Utara berhak menyimpan, mengalih media, memformat, mengelola dalam bentuk database, merawat dan mempublikasikan tesis saya tanpa meminta izin dari saya selama tetap mencantumkan nama saya sebagai penulis dan sebagai pemegang dan/atau sebagai pemilik hak cipta.

Demikian pernyataan ini dibuat dengan sebenarnya.

Medan, 7 September 2020



Boy Utomo Manalu

157038086

Telah diuji pada

Tanggal: 7 September 2020

PANITIA PENGUJI TESIS

Ketua : Prof. Dr. Tulus, Vor.Dipl.Math., M.Si.

Anggota : 1. Dr. Syahril Efendi, S.Si, M.IT
2. Prof. Dr. Muhammad Zarlis
3. Dr. Zakarias Situmorang

RIWAYAT HIDUP

DATA PRIBADI

Nama Lengkap : Boy Utomo Manalu
Tempat dan Tanggal Lahir : Medan, 26 Mei 1989
Alamat Rumah : Jl. Bendungan II Gg. Pancur, Kel. Bangun
Mulia, Kec. Medan Amplas, Kota Medan
Telp/HP : 081376200916
Email : xmanalu@gmail.com

DATA PENDIDIKAN

SD	: KARTIKA I-1 Medan	TAMAT : 2001
SMP	: SMP Negeri 3 Medan	TAMAT : 2004
SMA	: SMA Kartika I-1 Medan	TAMAT : 2007
S1	: Teknologi Informasi USU	TAMAT : 2014
S2	: Teknik Informatika USU	TAMAT : 2020

UCAPAN TERIMA KASIH

Puji dan syukur kehadiran Allah SWT karena atas rahmat dan karuniaNya penulis dapat menyelesaikan tesis yang berjudul **“KINERJA DEEP LEARNING DALAM ANALISIS SENTIMEN ”** untuk memenuhi salah satu syarat dalam mencapai gelar Magister pada Jurusan Teknik Informatika Universitas Sumatera Utara Medan. Dalam kesempatan ini penulis menyadari bahwa banyak pihak yang ikut berperan dalam menyelesaikan tesis ini baik moril maupun materil. Oleh karena itu penulis mengucapkan rasa terima kasih kepada:

1. Bapak Prof. Dr. Runtung Sitepu, S.H., M.Hum., selaku Rektor Universitas Sumatera Utara Medan.
2. Bapak Prof. Dr. Opim Salim Sitompul, M.Sc, selaku Dekan Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara Medan.
3. Bapak Prof. Dr. Muhammad Zarlis, M.Sc, selaku Ketua Program Studi S2 Teknik Informatika, Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara Medan.
4. Bapak Dr. Syahril Efendi, S.Si, M.IT, selaku Sekretaris Program Studi S2 Teknik Informatika, Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara Medan
5. Bapak Prof. Dr. Tulus, Vor. Dipl. Math., M.Si., Ph.D, selaku Dosen Pembimbing I yang telah memberikan bimbingan dan arahan dalam penyelesaian tesis ini.
6. Bapak Dr. Syahril Efendi, S.Si, M.IT, selaku Dosen Pembimbing II yang juga telah memberikan saran dan masukan untuk perbaikan dan penyelesaian tesis ini.
7. Bapak Dr. Muhammad Zarlis, M.Sc, Dosen Pembimbing/Penguji I yang telah memberikan saran dan masukan untuk perbaikan dan penyelesaian tesis ini;
8. Bapak Dr. Zakarias Situmorang, sebagai Dosen Pembimbing/Penguji II yang telah memberikan saran dan masukan untuk perbaikan dan penyelesaian tesis ini
9. Kepada orang tua penulis, yaitu Mamak Dra. Rosnah Siregar, M.Si, Ibu mertua Nurhayati, SmHk. serta kepada Istri tercinta Lia Silviana, S.Ti. dan keluarga besar yang telah memberikan doa, dukungan moril maupun materil kepada penulis selama ini sehingga penulis mampu menyelesaikan kegiatan perkuliahan ini.
10. Seluruh tenaga pengajar dan pegawai di Fakultas Ilmu Komputer dan Teknologi Informasi USU dan Progam Studi Magister (S2) Teknik Informatika.

11. Seluruh pihak yang terlibat langsung dan tidak langsung dalam penulisan tesis ini dan tidak bisa disebutkan satu persatu.

Semoga Allah SWT memberikan rahmat, kasih sayang, dan balasan kepada semua pihak yang telah memberikan bantuan, masukan, dan semangat kepada penulis untuk menyelesaikan tesis ini. Penulis berharap tesis ini dapat bermanfaat kepada penulis dan pembaca.

Medan, 7 September 2020
Penulis



Boy Utomo Manalu
157038086

ABSTRAK

Masalah analisis sentimen adalah representasi teks, yang mengkodekan teks ke dalam vektor kontinu dengan mengatur proyeksi dari semantik ke titik-titik dalam ruang dimensi tinggi. Metode pembelajaran mendalam telah banyak digunakan untuk menyelesaikan berbagai masalah analisis sentimen. Untuk meningkatkan kinerja pembelajaran mendalam dalam analisis sentimen diperlukan metode representasi teks yang baik untuk digunakan sebagai lapisan penyematan. Dalam penelitian ini menganalisis pembelajaran mendalam yaitu *metode Recurrent Neural Network* (RNN) dengan varian *Long Short-Term Memory* (LSTM), yang dibandingkan dengan RNN-LSTM dan Word2Vec sebagai embedding kata dalam klasifikasi sentimen. Data sentimen yang digunakan berasal dari ulasan-ulasan terhadap sebuah aplikasi yang disediakan pengguna di Google Play. Dalam proses pelatihan melibatkan lapisan Dropout dan *Early Stopping points* untuk mencegah overfitting. Hasil penelitian menunjukkan bahwa jaringan LSTM menggunakan Word embedded Word2Vec dengan dimensi 300 kata menerima nilai kesalahan rendah 0,3287 dengan akurasi 86,76%. Sedangkan hasil tes LSTM tanpa Word2Vec mendapatkan kesalahan terendah 0,3751 dengan akurasi 84,14%.

Kata kunci: Analisis Sentimen, *RNN*, *LSTM*, *Word Embedding*, *Word2Vec*.

DEEP LEARNING PERFORMANCE IN SENTIMENT ANALYSIS

ABSTRACT

Problem of sentiment analysis is the text representation, which encodes text into a continuous vector by arranging projections from semantics to points in high dimensional space. Deep learning methods have been widely used to solve various sentiment analysis problems. To improve the performance of deep learning in sentiment analysis requires a good method of text representation to be used as an embedding layer. In this study, we analyzed deep learning with the Recurrent Neural Network (RNN) method with Long Short-Term Memory (LSTM) variants, which were compared with RNN-LSTM and Word2Vec as word embedding in sentiment classification. Sentiment data used is derived from user-provided reviews of the applications in Google Play. In the training process involves Dropout layer and Early Stopping points to prevent overfitting. The results showed that the LSTM network using word embedding Word2Vec with 300 words dimension received a low error value of 0.3287 with an accuracy of 86.76%. While the LSTM test results without Word2Vec get the lowest error of 0.3751 with an accuracy of 84.14%.

Keywords: Sentiment Analysis, RNN, LSTM, Word Embedding, Word2Vec.

DAFTAR ISI

	<i>Hal.</i>
PERSETUJUAN	ii
PERYATAAN ORISINILITAS	iii
PERNYATAAN PERSETUJUAN PUBLIKASI	iv
RIWAYAT HIDUP	vi
UCAPAN TERIMA KASIH	vii
ABSTRAK	ix
ABSTRACT	x
DAFTAR ISI	xi
DAFTAR TABEL	xiii
DAFTAR GAMBAR	xiv
 BAB 1 PENDAHULUAN	 1
1.1. Latar Belakang	1
1.2. Rumusan Masalah	3
1.3. Batasan Masalah	4
1.4. Tujuan Penelitian	4
1.5. Manfaat Penelitian	4
 BAB 2 LANDASAN TEORI	 5
2.1. <i>Sentiment Analysis</i>	5
2.2. <i>Deep Learning</i>	5
2.3. <i>Recurrent Neural Network (RNN)</i>	6
2.4. Long Short-Term Memory	7
2.5. <i>Word Embedding</i>	12
2.5.1. CBOW	13
2.5.2. Skip-gram	14
2.6. Confusion Matrix	14
2.7. Penelitian Terdahulu	15
 BAB 3 METODOLOGI PENELITIAN	 20
3.1. Data yang digunakan	20
3.2. Arsitektur Umum	22
3.3. Preprocessing	25
3.3.1. Case Folding	26
3.3.2. Filtering	26
3.3.3. Stopword Removal	26
3.3.4. Stemming	27
3.3.5. Tokenizing	28
3.4. Representasi Teks	28
3.4.1. Word sequence	28
3.4.2. Word Embedding	29
3.5. Arsitektur Jaringan	31
 BAB 4 HASIL DAN PEMBAHASAN	 33

4.1.	Spesifikasi perangkat yang digunakan	33
4.2.	Hasil Preprocessing Data	34
4.3.	Arsitektur Jaringan	35
4.4.	Hasil Pemrosesan Data	36
4.4.1.	Hasil training dan testing LSTM tanpa Word2Vec	36
4.4.2.	Hasil training dan testing LSTM menggunakan Word2Vec 100 dimensi kata	39
4.4.3.	Hasil training dan testing LSTM menggunakan Word2Vec 200 dimensi kata	42
4.4.4.	Hasil training dan testing LSTM menggunakan Word2Vec 300 dimensi kata	45
4.5.	Pembahasan	48
BAB 5 KESIMPULAN DAN SARAN		50
5.1.	Kesimpulan	50
5.2.	Saran	50
DAFTAR PUSTAKA		52
LAMPIRAN 1 SOURCE CODE PROGRAM DAN LINK DATASET		55

DAFTAR TABEL

	<i>Hal.</i>
Tabel 2. 1 Tabel penelitian terdahulu	18
Tabel 3. 1 Data Ulasan yang diambil dari Google Play	21
Tabel 3. 2 Rincian Jumlah Data Berdasarkan Jenis Sentimen	22
Tabel 3. 3 Contoh Data Ulasan dengan jenis Sentimen	22
Tabel 3. 4 Penjelasan Blok Diagram Arsitektur Sistem	24
Tabel 3. 5 Contoh Hasil Proses <i>Case Folding</i>	26
Tabel 3. 6 Contoh Filtering	26
Tabel 3. 7 Contoh Stopword	27
Tabel 3. 8 Contoh Penerapan Stopword Removal	27
Tabel 3. 9 Contoh Penerapan Stemming	28
Tabel 3. 10 Contoh Penerapan Tokenizing	28
Tabel 3. 11 Contoh Penerapan Representasi Teks	29
Tabel 4. 1 Spesifikasi Perangkat yang digunakan	33
Tabel 4. 2 Hasil preprocessing data	34
Tabel 4. 3 Hasil Training LSTM tanpa Word2Vec	36
Tabel 4. 4 Tabel Confusion matrix hasil testing model LSTM tanpa Word2Vec	38
Tabel 4. 5 Kinerja klasifikasi model LSTM tanpa Word2Vec	39
Tabel 4. 6 Hasil Training Data dengan Word2Vec 100 dimensi kata	39
Tabel 4. 7 Tabel Confusion matrix hasil testing model LSTM dengan Word2Vec 100 Dimensi kata	41
Tabel 4. 8 Kinerja klasifikasi model LSTM dengan Word2Vec 100 Dimensi kata	41
Tabel 4. 9 Training Data dengan Word2Vec 200 dimensi kata	42
Tabel 4. 10 Tabel Confusion matrix hasil testing model LSTM dengan Word2Vec 200 Dimensi kata	44
Tabel 4. 11 Kinerja klasifikasi model LSTM+Word2Vec 200 Dimensi kata	44
Tabel 4. 12 Hasil Training Data dengan Word2Vec 300 dimensi	45
Tabel 4. 13 Tabel Confusion matrix hasil testing model LSTM dengan Word2Vec 300 Dimensi kata	46
Tabel 4. 14 Kinerja klasifikasi model LSTM dengan Word2Vec 300 Dimensi Kata	47
Tabel 4. 15 Perbandingan Kinerja <i>Training</i> 4 Model Percobaan	48
Tabel 4. 16 Perbandingan Kinerja <i>Testing</i> 4 Model Percobaan	49

DAFTAR GAMBAR

	<i>Hal.</i>
Gambar 2. 1 Perulangan pada arsitektur <i>RNN</i> (Olah, 2015)	6
Gambar 2. 2 Pengiriman informasi masa lalu pada <i>RNN</i> (Olah, 2015)	7
Gambar 2. 3 Sebuah lapisan <i>tanh</i> pada jaringan <i>RNN</i> (Olah, 2015)	7
Gambar 2. 4 Perulangan dengan empat layer pada <i>LSTM</i> (Olah, 2015)	8
Gambar 2. 5 Cell state pada jaringan <i>LSTM</i> (Olah, 2015)	8
Gambar 2. 6 Lapisan <i>Sigmoid</i> pada jaringan <i>LSTM</i> (Olah, 2015)	9
Gambar 2. 7 Jaringan <i>LSTM</i> (Miedema, 2018)	10
Gambar 2. 8 Hubungan Kata pada Word2Vec (Mikolov et. al, 2013)	13
Gambar 2. 9 Arsitektur CBOW dan Skip-Gram (Mikolov et. al, 2013)	13
Gambar 3. 1 Contoh ulasan di situs <i>Google Play</i>	20
Gambar 3. 2 Arsitektur Sistem	23
Gambar 4. 1 Hasil Training LSTM Tanpa Word2Vec	37
Gambar 4. 2 Confusion Matriks dari Hasil Testing LSTM Tanpa Word2Vec	38
Gambar 4. 3 Hasil Training LSTM dengan Word2Vec 100 Dimensi Kata	40
Gambar 4. 4 Confusion Matriks dari Hasil Testing LSTM dengan Word2Vec 100 Dimensi Kata	41
Gambar 4. 5 Hasil Training LSTM dengan Word2Vec 200 Dimensi Kata	43
Gambar 4. 6 Confusion Matriks dari Hasil Testing LSTM dengan Word2Vec 200 Dimensi kata	44
Gambar 4. 7 Hasil Training LSTM dengan Word2Vec 300 Dimensi Kata	46
Gambar 4. 8 Confusion Matriks dari Hasil Testing LSTM dengan Word2Vec 300 Dimensi Kata	47

BAB 1

PENDAHULUAN

1.1. Latar Belakang

Pertumbuhan penggunaan Internet memberikan pengaruh besar terhadap perkembangan penggunaan aplikasi media sosial. Pada tahun 2018, penggunaan Internet khususnya untuk media sosial di Indonesia mencapai lebih 130 juta pengguna (Haryanto, 2018). Selain itu berkembangannya berbagai situs yang menyajikan fitur ulasan untuk menilai layanan sebuah situs, film, aplikasi dan lainnya. Perkembangan tersebut menyebabkan tersedianya jumlah data teks yang sangat besar, baik melalui pesan, tulisan, maupun komentar mengenai sebuah topik.

Kumpulan teks tersebut menjadi sumber daya besar yang kaya opini, pendapat maupun sentimen. Analisis sentimen telah menjadi salah satu topik penting dalam pemrosesan bahasa alami (*Natural Language Processing*). Analisis sentimen dapat didefinisikan sebagai proses komputasi dari pernyataan, opini-opini, sentimen dan emosi yang diekspresikan dalam bentuk teks. Analisis sentimen bertujuan untuk mengklasifikasikan polaritas sentimen dari teks yang diberikan yaitu sebagai sentimen negatif, positif atau kelas yang lain (Li, et al. 2019). Analisis sentimen ini dapat membantu perorangan maupun perusahaan untuk memproses dan mengekstrak informasi berharga dari jumlah data yang besar, yang mengandung nilai bisnis akan sebuah merek, layanan pelanggan, situasi pasar, politik, dan layanan sosial (Li & Qiu, 2018).

Masalah mendasar analisis sentimen adalah representasi teks, yang menyandikan teks menjadi vektor kontinu dengan menyusun proyeksi dari semantik ke titik dalam ruang dimensi tinggi (Liu, 2012). Model representasi pembelajaran mesin tradisional bergantung pada pengetahuan linguistik seperti *bag-of-words* dan *sentiment lexicon*, di mana polaritas sentimen teks sebagian besar ditentukan untuk menjadi positif jika jumlah kata-kata positif lebih besar daripada kata-kata negatif. Sebaliknya, saat ini sedang populer model representasi berbasis *deep learning* memanfaatkan jaringan saraf

yang mendalam untuk mempelajari informasi semantik yang terkandung dalam teks. Kinerja model representasi berbasis *deep learning* sering lebih unggul daripada model representasi berbasis *machine learning* ketika struktur sintaksis teks tersebut lebih kompleks (Li, et al. 2019).

Deep learning merupakan salah satu teknik dalam *machine learning* yang memiliki arsitektur yang lebih mendalam dibanding dengan teknik *machine learning* lainnya dalam menyelesaikan masalah prediksi maupun klasifikasi (Patterson & Gibson, 2017). Arsitektur umum *deep learning* adalah *Deep Neural Network* (DNN), *Deep Believe Network* (DBN), *Deep Convolutional Neural Network* (DCNN), dan *Deep Recurrent Neural Network* (DRNN).

Dalam beberapa penelitian DNN sesuai digunakan untuk data berjenis multivariasi dengan banyak *input* neuron, yang kemudian dilakukan *feed-forward* satu arah terhadap DBN tersebut. Sedangkan DCNN menggunakan *max* and *pool layer*, serta *dense layer* yang lebih sesuai dengan klasifikasi citra gambar. Untuk pengenalan teks, bahasa, ataupun data dengan tipe *time series* maka DRNN akan lebih sesuai diterapkan (Nikoskinen, 2015).

Metode *deep learning* telah banyak digunakan untuk memecahkan berbagai permasalahan analisis sentimen, diantaranya oleh Wijanarko & Zulfa (2017) yang melakukan mengklasifikasi data Tweet berbahasa Indonesia yang berasal dari Twitter. Dalam penelitiannya mereka melihat bagaimana sentimen yang terdapat pada data uji apakah bernilai positif, negatif, atau netral. Peneliti ingin melihat bagaimana kinerja model klasifikasi yang dibangun dengan menerapkan metode Deep Belief Network pada data Tweet yang dikumpulkan. Hasil pengujian dengan model tersebut memperlihatkan bahwa metode terbaik adalah dengan menggunakan metode DBN yang memperoleh keakuratan sebesar 93,31%. Hasil yang diperoleh tersebut lebih baik dibandingkan dengan metode Naive Bayes yang dengan keakuratan sebesar 79,10% dan metode Support Vector Machine dengan keakuratan sebesar 92,18%.

Long Short-Term Memory (LSTM), mulai banyak digunakan untuk memecahkan berbagai masalah dengan data berjenis *time-series*. Beberapa penelitian tentang LSTM diantaranya dilakukan oleh Hassan & Mahmood (2017) melakukan klasifikasi sentimen kalimat menggunakan *Recurrent Neural Network* (RNN) dengan Teknik LSTM, penelitiannya memaparkan teknik LSTM satu *single layer* dan model *word2vec* yang diuji pada dua *benchmark* dataset, *rate error* dari SSTb dataset 14,3%

unggul dari tujuh metode lain, dan *rate error* pada IMDB dataset 11,32% unggul dari sepuluh metode lain.

Ciftci & Apaydin (2018) dalam penelitiannya mengusulkan *deep learning* untuk analisis sentimen berbahasa Turki. Peneliti menganalisis bahwa masih terdapat kekurangan dalam metode pembelajaran mesin seperti Logistic Regression maupun Naive Bayes. Penulis menerapkan teknik *Recurrent Neural Network* menggunakan LSTM kemudian dibandingkan dengan *Logistic Regression* dan *Naive Bayes*. Eksperimen menggunakan data yang diambil dari situs belanja dan film. Hasil yang didapat adalah LSTM memiliki akurasi yang lebih baik dari metode lainnya dengan nilai validasi akurasi, pengujian akurasi, presisi dan *recall* masing-masing adalah 83,3%, 82,9%, 0,86, dan 0,83.

Marwa Naili, et al. (2017) dalam penelitiannya menguji segmentasi topik dengan menggunakan *word embedding* sebagai dasar representasi data. Peneliti menggunakan beberapa metode yaitu LSA, Word2Vec dan GloVe untuk mengidentifikasi metode mana lebih efektif untuk mempelajari representasi vektor kata yang memberikan makna semantik kata-kata untuk bahasa Inggris dan bahasa Arab. Hasil penelitian menunjukkan bahwa Word2Vec dan GloVe lebih efektif daripada LSA untuk kedua bahasa. Apalagi dibandingkan dengan GloVe, Word2Vec menghadirkan representasi vektor kata terbaik dengan ruang semantik dimensi kecil. Kemudian dalam penelitian tersebut menunjukkan bahwa kualitas segmentasi topik tergantung pada bahasa yang digunakan.

Dari beberapa penelitian terdahulu penulis akan menganalisis lebih dalam sehingga mendapatkan hasil kinerja yg lebih optimal. Dalam penelitian ini, penulis akan menggunakan metode *deep learning* Recurrent Neural Network dengan varian Long Short Term Memory (RNN – LSTM) serta penambahan *word embedding* pada analisis sentimen tersebut. Hasil yang diharapkan nantinya berupa analisis kinerja seperti kecepatan maupun akurasi analisis sentimen.

1.2. Rumusan Masalah

Kinerja dalam analisis sentimen menggunakan *deep learning* RNN – LSTM masih memerlukan akurasi yang tinggi, oleh karena itu diperlukan suatu metode yang dapat meningkatkan kinerja dalam analisis sentimen tersebut.

1.3. Batasan Masalah

Adapun batasan masalah dari penelitian ini adalah:

1. Metode *deep learning* yang digunakan adalah *RNN - LSTM* serta penambahan *optimizer*.
2. Menggunakan Algoritma *word embedding Word2Vec* untuk pemetaan dari kata menjadi vektor.
3. Data yang digunakan adalah data ulasan (*review*) beberapa aplikasi di Google Play Store.

1.4. Tujuan Penelitian

Tujuan dari penelitian ini adalah mendapatkan kinerja yang lebih baik menggunakan metode *deep learning* dengan teknik *RNN-LSTM* serta penambahan *Word Embedding* dalam analisis sentimen.

1.5. Manfaat Penelitian

Adapun manfaat dari penelitian ini adalah:

1. Menghasilkan *Deep Learning* yang mampu melakukan klasifikasi secara optimal.
2. Memudahkan didalam pengklasifikasian berdasarkan jenis sentimen pada sebuah teks

BAB 2

LANDASAN TEORI

2.1. *Sentiment Analysis*

Sentiment Analysis (analisis sentimen) yang merupakan bagian dari *opinion mining* adalah salah satu bidang studi untuk menganalisis pendapat, sentimen, penilaian dan evaluasi, sikap seseorang, keadaan emosi terhadap sebuah entitas seperti produk, layanan, individu, organisasi, peristiwa, topik, dan atributnya (Liu, 2012). Tujuan dasar dalam analisis sentimen adalah mengetahui dan mengelompokkan kecenderungan muatan dari teks yang ada dalam sebuah kalimat.

Dengan analisis sentimen ini kita dapat mengetahui bagaimana pendapat yang terkandung dalam dokumen atau kalimat mengacu pada topik tertentu. Pernyataan pendapat yang terdapat pada sebuah topik bisa saja berbeda makna dengan pernyataan yang sama pada subjek yang berbeda. Sebagai contoh, dalam menganalisa bagaimana sentimen pada data produk, kita tidak bisa melihat dari objek yang melekat pada subjek tersebut, misalnya kita mau melihat sentimen terhadap sebuah aplikasi, kita akan melihat sentimen terhadap hal-hal yang melekat pada aplikasi tersebut, seperti fitur, kemampuan, kehandalan, penggunaan memori dan sebagainya. Semua yang kita jadikan nilai pada aplikasi bisa saja tidak cocok kita terapkan ketika ingin memprediksi sentimen terhadap kendaraan. Oleh karena itu pada beberapa penelitian sebelumnya, untuk melihat bagaimana sentimen terhadap sebuah produk, hal yang akan dikerjakan terlebih dahulu adalah menentukan elemen atau objek dari sebuah produk yang sedang dibicarakan sebelum memulai proses *opinion mining* (Ian, 2010).

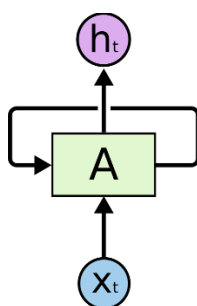
2.2. *Deep Learning*

Butuh waktu yang lama menerapkan arsitektur jaringan saraf dibandingkan dengan model pembelajaran mesin lainnya, untuk memecahkan beberapa masalah yang tidak menguntungkan karena kurangnya sumber daya komputasi dan data. Namun, dapat

dilihat sekarang bahwa teknologi tersebut yang dimiliki telah maju secara eksponensial dari tahun ke tahun yang juga dipengaruhi oleh pertumbuhan infrastruktur komputasi dan oleh ketersediaan sejumlah besar data pelatihan dengan kualitas yang baik. Akibatnya, model jaringan saraf kompleks telah dikembangkan. *Deep learning* merupakan salah satu teknik dalam *machine learning* yang memiliki arsitektur yang lebih mendalam dibanding dengan teknik *machine learning* lainnya dalam menyelesaikan masalah prediksi maupun klasifikasi (Patterson & Gibson, 2017). Arsitektur umum *deep learning* adalah *Deep Neural Network* (DNN), *Deep Believe Network* (DBN), *Deep Convolutional*

2.3. *Recurrent Neural Network (RNN)*

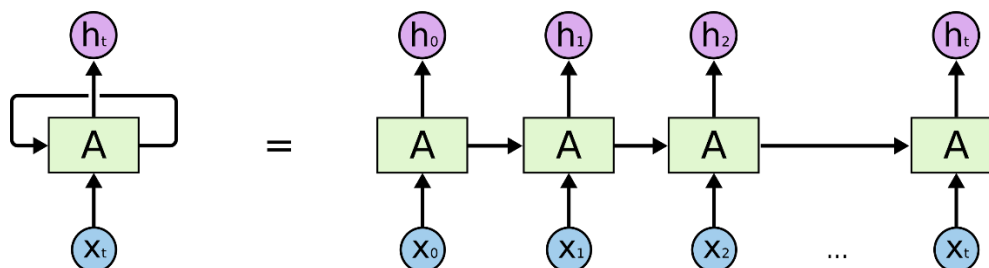
Recurrent Neural Network (RNN) adalah salah satu arsitektur neural network yang pemrosesannya dipanggil berulang-ulang untuk mengolah data masukan yang biasanya sejumlah data yang bersambung (*sequential data*). Dasar dari pengembangan arsitektur *RNN* berdasarkan dari pola pikir manusia yang tidak mengambil keputusan secara tunggal setiap saat, karena manusia selalu memperhitungkan informasi yang diterima atau yang disimpan pada masa lalu dalam membuat sebuah keputusan. *RNN* memperhitungkan informasi yang diterima pada masa sebelumnya dalam membuat sebuah keputusan. *RNN* menyimpan informasi dari masa lalu dengan melakukan pengulangan dalam arsitekturnya sehingga informasi dari masa lalu tetap tersimpan seperti yang dapat dilihat pada gambar 2.1.



Gambar 2. 1 Perulangan pada arsitektur *RNN* (Olah, 2015)

Dari Gambar 2.1 diatas, sebuah jaringan saraf dimana x_t sebagai masukan, h_t sebagai keluaran dan terdapat sebuah perulangan sehingga memungkinkan informasi dilewatkan dari satu langkah jaringan ke langkah selanjutnya. *RNN* tidak jauh berbeda

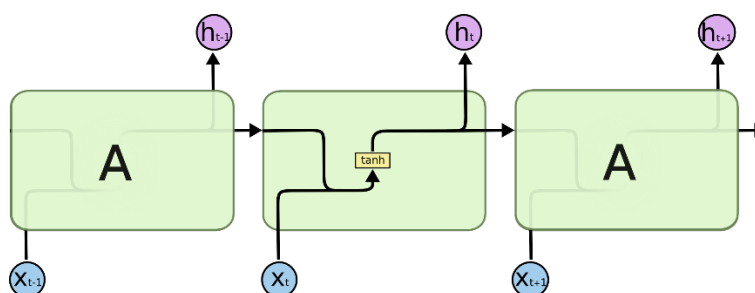
dari jaringan saraf normal. RNN dapat digambarkan sebagai beberapa salinan dari jaringan yang sama dimana tiap-tiap jaringan diteruskan ke jaringan berikutnya seperti yang dapat dilihat pada Gambar 2.2 dibawah ini.



Gambar 2. 2 Pengiriman informasi masa lalu pada RNN (Olah, 2015)

2.4. Long Short-Term Memory

Long Short-Term Memory (LSTM) merupakan salah satu jenis arsitektur jaringan RNN yang dikembangkan untuk menghindari kendala RNN yang kurang baik terhadap masalah memori jangka panjang. LSTM memiliki kemampuan dalam mengingat informasi jangka panjang. Dalam Arsitektur RNN, jaringan hanya menggunakan satu layer sederhana pada perulangannya, yaitu sebuah layer *tanh* seperti pada gambar 2.3



Gambar 2. 3 Sebuah lapisan *tanh* pada jaringan RNN (Olah, 2015)

Persamaan tanh diuraikan pada persamaan 2.1.

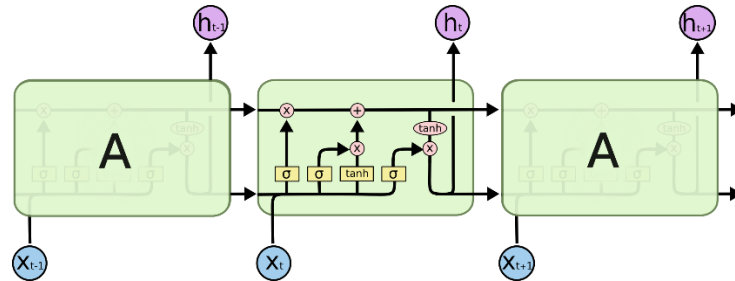
$$\tanh(x) = 2\sigma(2x) - 1 \quad (2.1)$$

Dimana:

σ = fungsi aktivasi sigmoid

x = data input

Sementara itu, pada *LSTM* terdapat empat *layer* pada perulangannya seperti yang dapat dilihat pada gambar 2.4.



Gambar 2.4 Perulangan dengan empat layer pada *LSTM* (Olah, 2015)

Menurut Hochreiter & Schmidhber (1997), persamaan yang ada pada metode *LSTM* dapat diuraikan pada persamaan dibawah ini:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (2.2)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (2.3)$$

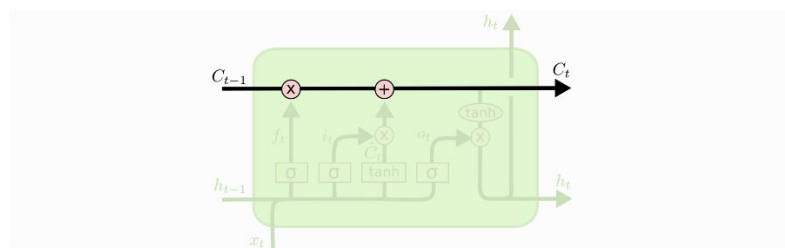
$$\tilde{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \quad (2.4)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad (2.5)$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (2.6)$$

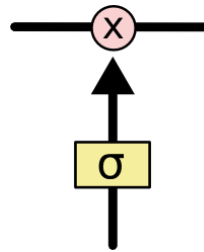
$$h_t = o_t * \tanh C_t \quad (2.7)$$

Kunci untuk *LSTM* adalah *cell state*, garis horizontal berjalan melalui bagian atas diagram. Sel Ini berjalan lurus ke bawah seluruh rantai, dengan hanya beberapa interaksi linear kecil. Sangat mudah bagi informasi untuk mengalir tanpa berubah. *Cell state* menghubungkan semua lapisan keluaran pada *LSTM* seperti terlihat pada gambar 2.5.



Gambar 2.5 Cell state pada jaringan *LSTM* (Olah, 2015)

LSTM memang memiliki kemampuan untuk menghapus atau menambahkan informasi ke keadaan sel, diatur dengan cermat oleh struktur yang disebut gerbang (*gates*). Gates adalah cara untuk membiarkan informasi masuk secara opsional. Mereka terdiri dari lapisan jaringan saraf sigmoid dan operasi multiplikasi yang searah seperti yang ditunjukkan pada Gambar 2.6.



Gambar 2. 6 Lapisan Sigmoid pada jaringan LSTM (Olah, 2015)

Pada lapisan sigmoid ini diterapkan fungsi sigmoid seperti yang diuraikan pada persamaan 2.8. Keluaran dari lapisan sigmoid adalah angka 1 atau 0. Angka 0 menyatakan bahwa informasi tidak diteruskan, sementara angka 1 menyatakan akan meneruskan informasi yang diterima.

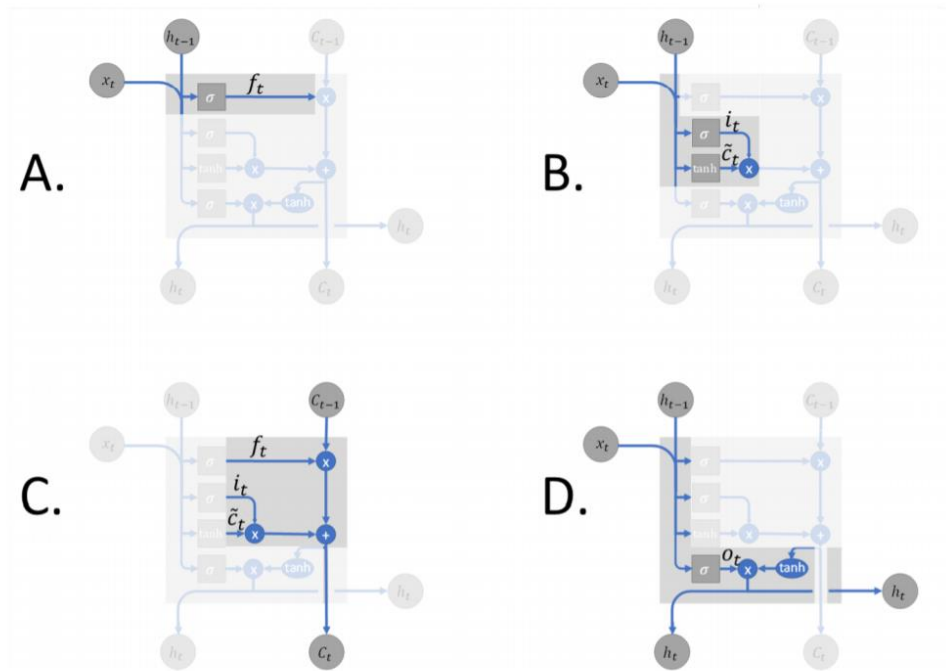
$$\sigma(x) = 1/(1 + e^{-x}) \quad (2.8)$$

Dimana:

x = data *input*

e = konstanta matematika (2,71828 18284 59045 23536 02874 71352)

Terdapat 3 jenis gerbang pada jaringan yaitu: gerbang *Forget* yang merupakan gerbang menentukan informasi mana yang akan dihapus dari *cell*. Kemudian gerbang *Input* yang merupakan gerbang yang memutuskan nilai dari *input* untuk di diperbarui pada *state* memori. Yang terakhir gerbang *Output* yang memutuskan keluaran apa yang akan dihasilkan berdasarkan *input* dan memori pada *cell*. Proses yang terjadi pada jaringan LSTM dapat dilihat pada gambar 2.7 dibawah ini.



Gambar 2.7 Jaringan LSTM (Miedema, 2018)

Pada tahapan pertama, model jaringan LSTM akan menentukan informasi apa yang akan dibuang dari *cell state* seperti yang ditunjukkan pada Gambar 2.7 bagian A. Lapisan ini disebut *forget gate*. Input pada lapisan adalah output dari langkah sebelumnya yaitu h_{t-1} dan x_t sebuah fungsi aktivasi sigmoid akan digunakan untuk menghasilkan keluaran berupa 0 atau 1 pada C_{t-1} . Persamaan *forget gate* diuraikan pada persamaan 2.2

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

Dimana σ adalah fungsi sigmoid, W_f dan b_f masing-masing merupakan matriks bobot dan bias pada *forget gate*.

Untuk menentukan nilai bobot pada W_f diuraikan pada persamaan 2.9.

$$W = \left(-\frac{1}{\sqrt{d}}, \frac{1}{d} \right) \quad (2.9)$$

Tahapan selanjutnya adalah menentukan informasi yang akan disimpan di *cell state*. Lapisan *input gate* pertama kali menggunakan lapisan sigmoid di atas input untuk

menentukan bagian mana dari *cell state* akan diperbarui. Kemudian lapisan *tanh* membuat satu kandidat yang baru, \tilde{C}_t , yang dapat ditambahkan ke *cell state*. Pada langkah berikutnya, keduanya akan digabungkan untuk memperbarui *cell state*. Tahapan ini digambarkan pada gambar 2.7 poin B. Persamaan *input gate* diuraikan pada persamaan 2.3:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

Dimana σ adalah fungsi sigmoid, W_i dan b_i masing-masing adalah matriks bobot dan bias pada *input gate*. Persamaan kandidat baru C_t diuraikan pada persamaan 2.4:

$$\tilde{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$$

Dimana: *tanh* adalah fungsi tanh, W_c dan b_c masing-masing adalah nilai bias untuk *cell state*.

Selanjutnya *cell state* lama C_{t-1} dikalikan dengan f_t untuk menghapus informasi yang sudah ditentukan sebelumnya pada lapisan *forget gate*. Kemudian informasi baru $i_t * \tilde{C}_t$ ditambahkan untuk memperbarui *cell state*. Persamaan *cell state* diuraikan pada persamaan 2.5.

$$C_t = f_{t-1} * C_t + i_t * \tilde{C}_t$$

Dimana:

$C_t = \text{Cell state}$

$f_t = \text{forget gate}$

$C_{t-1} = \text{Cell state}$ sebelum orde ke t

$i_t = \text{input gate}$

$\tilde{C}_t =$ nilai baru yang dapat ditambahkan ke *cell state*

Pada langkah terakhir, ditentukan apa output h_t , ditunjukkan pada Gambar 2.7 bagian D. Output ini didasarkan pada *cell state* tetapi dalam versi yang difilter. Pertama, lapisan sigmoid diterapkan ke sebelumnya output h_{t-1} dan input x_t , untuk menentukan nilai gerbang output o_t . Ini adalah nilai antara 0 dan 1 yang menunjukkan bagian mana dari *cell state* menjadi output. Kemudian *cell state* C_t diubah oleh fungsi tanh menjadi

nilai antara -1 dan 1. Nilai *cell state* yang ditransformasikan ini kemudian dikalikan dengan nilai *output gate* atau berakhir dengan output h_t . Keluaran ini akan dicetak dan didorong ke langkah berikutnya dari jaringan. Persamaan *output gate* o_t diuraikan pada persamaan 2.6.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

Dimana σ adalah fungsi sigmoid, W_o dan b_o masing-masing adalah matriks bobot dan nilai bias pada *output gate*. Persamaan nilai *output* orde t diuraikan pada persamaan 2.7.

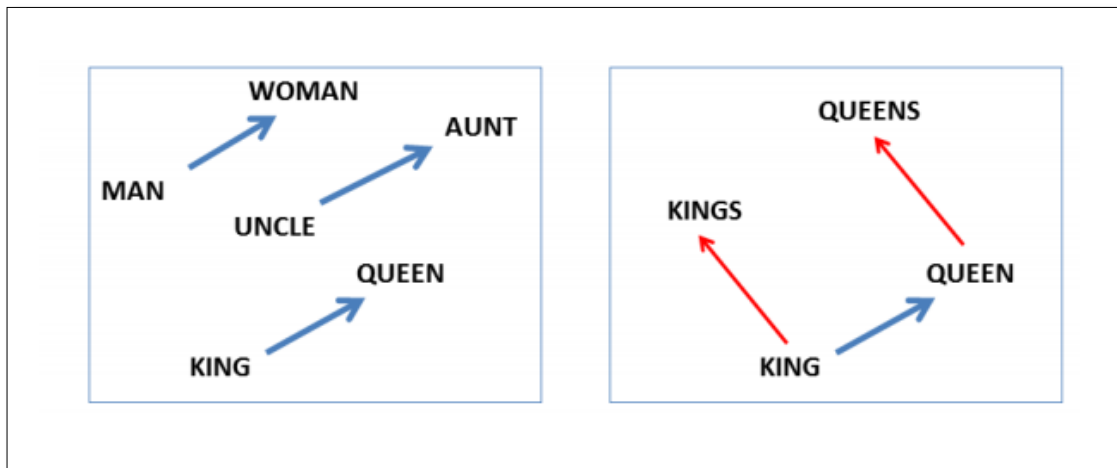
$$h_t = o_t * \tanh C_t$$

Dimana h_t adalah nilai *output* orde t , o_t adalah *output gate*, \tanh adalah fungsi $\tanh C_t$ adalah *Cell state*.

2.5. Word Embedding

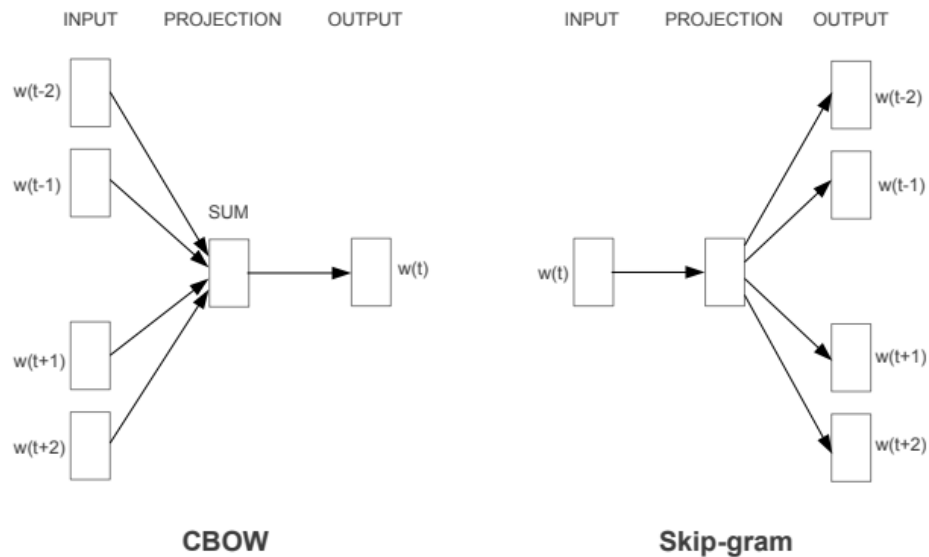
Word embedding merupakan teknik pembelajaran fitur dalam *NLP* untuk membangun representasi vektor kata dimensi rendah dari kumpulan teks. Keuntungan utama dari *word embedding* adalah memungkinkan untuk menawarkan representasi yang lebih ekspresif dan efisien dengan mempertahankan kesamaan kontekstual kata-kata dan dengan membangun vektor dimensi rendah (Naili, et al., 2017). Sebuah metode yang digunakan untuk melakukan *word embedding* menggunakan Word2Vec yang berbasis jaringan saraf tiruan (Mikolov et. al, 2013). Word embedding dapat meningkatkan kinerja fitur dan memungkinkan juga digunakan pada pengklasifikasi RNN dengan basis LSTM (Yepes, 2017).

Word2vec adalah salah satu metode yang merepresentasikan kata-kata di dalam sebuah konteks sebagai sebuah vektor dengan N dimensi. Word2Vec mengimplementasi neural network untuk kesamaan kontekstual dan semantik dari setiap kata (inputan) yang berbentuk *one-hot encoded vectors* dalam mempresentasikan suatu kata (Mikolov et. al 2013). Dengan hasil representasi ini dapat dilihat bagaimana relasi suatu kata dengan kata lainnya, misalnya relasi antara kata ‘Raja-Ratu’, ‘Pria-Wanita’, dan bahkan relasi pada ‘Negara-Ibu Kota’, seperti yang diilustrasikan pada gambar dibawah ini (mikolov et. al, 2013).



Gambar 2. 8 Hubungan Kata pada Word2Vec (Mikolov et. al, 2013)

Word2Vec memprediksi kata berdasarkan konteks kata dengan menggunakan salah satu dari dua model saraf yang berbeda: model *Continuous Bag-of-Words* (CBOW) dan model *Skip-Gram*. Arsitektur CBOW dan Skip-Gram dapat dilihat pada gambar dibawah ini.



Gambar 2. 9 Arsitektur CBOW dan Skip-Gram (Mikolov et. al, 2013)

2.5.1. CBOW

Continuous Bag of Words (CBOW) memprediksi kata saat ini berdasarkan konteksnya. Dalam proses CBOW, tiga lapisan digunakan. Lapisan input sesuai dengan konteksnya.

Lapisan tersembunyi sesuai dengan proyeksi setiap kata dari lapisan input ke matriks bobot yang diproyeksikan ke lapisan ketiga yang merupakan lapisan keluaran. Langkah terakhir dari model ini adalah perbandingan antara output dan kata itu sendiri untuk memperbaiki perwakilannya berdasarkan propagasi balik dari gradien kesalahannya.

2.5.2. *Skip-gram*

Skip-Gram merupakan kebalikan dari model CBOW. Bahkan, lapisan input sesuai dengan kata target dan lapisan output sesuai dengan konteksnya. Jadi, Skip-Gram mencari prediksi dari konteks yang diberikan kata bukan prediksi kata yang diberikan konteksnya seperti CBOW. Langkah terakhir dari Skip-Gram adalah perbandingan antara output dan setiap kata dari konteks untuk memperbaiki perwakilannya berdasarkan propagasi balik dari gradien kesalahan.

Masing-masing model ini memiliki keunggulannya sendiri. Sebagai contoh, Skip-Gram lebih efisien dengan data pelatihan yang kecil. Apalagi kata-kata yang jarang disajikan dengan baik. Di sisi lain, CBOW lebih cepat dan bekerja dengan baik dengan kata-kata yang berulang (Mikolov et al., 2013).

2.6. Confusion Matrix

Confusion Matrix merupakan sebuah teknik pembelajaran mesin yang memiliki informasi tentang keadaan data sebenarnya dan hasil prediksi sebuah klasifikasi yang telah dilakukan menggunakan model atau metode klasifikasi yang digunakan. Kinerja klasifikasi dievaluasi menggunakan data dalam matriks yang berisi data aktual sebuah hasil prediksi dan metode klasifikasi yang digunakan. Berikut ini beberapa nilai pada Confusion Matrix yang digunakan pada penelitian ini yaitu:

1. True Negative (TN) adalah jumlah prediksi yang dinyatakan benar bahwa sebuah data diprediksi negatif,
2. False Positive (FP) adalah jumlah prediksi yang dinyatakan salah bahwa sebuah data diprediksi positif,
3. False Negative (FN) adalah jumlah prediksi yang dinyatakan salah bahwa sebuah data diprediksi negatif, dan
4. True Positive (TP) adalah jumlah prediksi yang dinyatakan benar bahwa sebuah data diprediksi bernilai positif.

Beberapa perhitungan kinerja klasifikasi dapat dijelaskan dari *confusion matrix* antara lain :

2.6.1. Accuracy

Accuracy adalah presentase dari jumlah total prediksi yang benar pada proses klasifikasi (Deng et al, 2016).

$$Accuracy = \frac{TP + TN}{n} \quad (2.10)$$

2.6.2. Recall

Recall adalah presentase data positif yang diprediksi sebagai nilai positif (Saraswathi & Sheela, 2016).

$$Recall = \frac{TP}{TP + FN} \quad (2.11)$$

2.6.3. Precision

Precision merupakan perbandingan prediksi benar positif dibandingkan dengan keseluruhan hasil yang diprediksi positif.

$$Precision = \frac{TP}{TP + FP} \quad (2.13)$$

2.6.4. F1 Score

F1 Score merupakan perbandingan rata-rata presisi dan recall yang dibobotkan.

$$F1\ Score = \frac{2 * Precision * Recall}{Precision + Recall} \quad (2.14)$$

2.7. Penelitian Terdahulu

Metode deep learning telah banyak digunakan untuk memecahkan berbagai permasalahan analisis sentimen, diantaranya oleh Wijanarko & Zulfa (2017) yang melakukan pengklasifikasian data Tweet berbahasa Indonesia yang berasal dari Twitter. Dalam penelitiannya mereka melihat bagaimana sentimen yang terdapat pada data uji

apakah bernilai positif, negatif, atau netral. Peneliti ingin melihat bagaimana kinerja model klasifikasi yang dibangun dengan menerapkan metode Deep Belief Network pada data Tweet yang dikumpulkan. Hasil pengujian dengan model tersebut memperlihatkan bahwa metode terbaik adalah dengan menggunakan metode DBN yang memperoleh keakuratan sebesar 93,31%. Hasil yang diperoleh tersebut lebih baik dibandingkan dengan metode Naive Bayes yang dengan keakuratan sebesar 79,10%, dan metode Support Vector Machine dengan keakuratan sebesar 92,18%.

LSTM mulai banyak digunakan untuk memecahkan berbagai masalah dengan data berjenis *time-series*. Beberapa penelitian tentang LSTM diantaranya dilakukan oleh Hassan & Mahmood (2017) melakukan klasifikasi sentimen kalimat menggunakan *Recurrent Neural Network (RNN) - Long Short Term Memory (LSTM)*, penelitiannya memaparkan teknik LSTM satu *single layer* dan model *word2vec* yang diuji pada dua *benchmark* dataset, *rate error* dari SSTb dataset 14,3% unggul dari tujuh metode lain, dan *rate error* pada IMDB dataset 11,32% unggul dari sepuluh metode lain.

Wang & Cao (2017) melakukan penelitian terhadap ulasan produk belanja online, semua ulasan dan data ini telah ditandai oleh peneliti sebagai positif dan negatif. Korpus ini terdiri dari 13.000 ulasan, yang memiliki 7000 komentar positif dan mencampur data positif dan negatif. Dalam penelitian ini dibandingkan model LSTM baru dengan model LSTM tradisional untuk analisis sentimen berbahasa Cina. Dalam percobaan, kalimat-kalimatnya disegmentasi dan dilakukan penyisipan kata pada semua ulasan produk. Dalam percobaan pengolah Keras digunakan untuk membangun model LSTM. Tiga Model LSTM (LSTM, LSTM dengan L2 dan Nadam, BLSTM) di kedalaman yang berbeda. Peneliti membandingkan akurasi dari tiga sentimen teks model analisis, termasuk berbasis LSTM, BLSTM dan LSTM berbasis Nadam dan L2. Hasilnya menunjukkan akurasi pengenalan model yang berbeda, akurasi BLSTM LSTM tinggi dan hanya membutuhkan hanya beberapa *epoch*. LSTM berdasarkan model Nadam dan L2 telah meningkat dalam hal kecepatan dan ketepatan konvergensi, yang jauh lebih tinggi dari LSTM. Karena LSTM berdasarkan L2 dan Nadam memiliki kemampuan yang lebih kuat untuk mencegah *over-fitting* dan meningkatkan model generalisasi dan kecepatan konvergensi.

Bin, et al (2018) yang melakukan penelitian analisis sentimen berbahasa China dimana dalam penelitian ini diusulkan strategi pemberian tag otomatis untuk pelatihan korpus dan sebuah algoritma klasifikasi untuk sentimen berbahasa Cina berdasarkan

pada kamus dan LSTM. Algoritma ini dapat memberi label korpus pelatihan secara otomatis dan akurat dan efisien, dan juga mengekstrak kata-kata sentimen. Dalam penelitian ini dilakukan perbandingan dengan SVM dan IRNLP tools, dimana LSTM menunjukkan hasil lebih baik. Eksperimen menunjukkan metode ini efektif dan akurasi algoritma LSTM telah mencapai 93,51% pada kumpulan data klasifikasi sentimen campuran.

Mirza & Cosan (2018) memperkenalkan sebuah kerangka kerja autoencoder berurutan yang menggunakan jaringan saraf *Long Short-Term Memory* (LSTM) untuk deteksi intrusi jaringan komputer. Penelitian ini mengeksplorasi pengurangan dimensi dan properti ekstraksi fitur kerangka kerja autoencoder untuk melakukan proses rekonstruksi secara efisien. Selanjutnya, peneliti menggunakan jaringan LSTM untuk menangani sifat berurutan dari data jaringan komputer. Peneliti menetapkan sebuah nilai ambang batas berdasarkan validasi silang untuk mengklasifikasikan apakah urutan data jaringan yang masuk adalah anomali atau tidak. Dalam penelitian ini digunakan juga versi LSTM, GRU, Bi-LSTM dan Jaringan Saraf Tiruan. Melalui serangkaian eksperimen yang komprehensif, dalam penelitian ini ditunjukkan bahwa kerangka kerja deteksi intrusi berurutan yang kami usulkan berkinerja baik dan dinamis, kuat dan terukur. Eksperimen ini menunjukkan LSTM dengan max pooling dan jaringan Deep Auto LSTM menunjukkan skor f1 terbaik.

Ciftci & Apaydin (2018) dalam penelitiannya mengusulkan *deep learning* untuk analisis sentimen berbahasa Turki. Peneliti menganalisis bahwa masih terdapat kekurangan dalam metode pembelajaran mesin seperti Logistic Regression maupun Naive Bayes. Penulis menerapkan teknik *Recurrent Neural Network* menggunakan LSTM kemudian dibandingkan dengan *Logistic Regression* dan *Naive Bayes*. Eksperimen menggunakan data yang diambil dari situs belanja dan film. Hasil yang didapat adalah LSTM memiliki akurasi yang lebih baik dari metode lainnya dengan nilai validasi akurasi, pengujian akurasi, presisi dan *recall* masing-masing adalah 83,3%, 82,9%, 0,86, dan 0,83.

Marwa Naili (2017) dalam penelitiannya menguji segmentasi topik dengan menggunakan *word embedding* sebagai dasar representasi data. Peneliti menggunakan beberapa metode yaitu LSA, Word2Vec dan GloVe untuk mengidentifikasi metode mana lebih efektif untuk mempelajari representasi vektor kata yang memberikan makna semantik kata-kata untuk bahasa Inggris dan bahasa Arab. Hasil penelitian

menunjukkan bahwa Word2Vec dan GloVe lebih efektif daripada LSA untuk kedua bahasa. Apalagi dibandingkan dengan GloVe, Word2Vec menghadirkan representasi vektor kata terbaik dengan ruang semantik dimensi kecil. Kemudian dalam penelitian tersebut menunjukkan bahwa kualitas segmentasi topik tergantung pada bahasa yang digunakan.

Tabel 2. 1 Tabel penelitian terdahulu

No	Peneliti (Tahun)	Metode	Keterangan
1	Wijanarko & Zulfa (2017)	<i>Deep Believe Network</i>	Analisis sentimen twitter dengan DBN memperoleh diperoleh keakuratan sebesar 93,31%, lebih baik daripada metode Naive Bayes dengan akurasi sebesar 79,10% atau metode Support Vector Machine dengan akurasi sebesar 92,18%
2	Hassan & Mahmood (2017)	<i>Recurrent Neural Network (RNN) - Long Short Term Memory (LSTM)</i>	Penelitiannya memaparkan teknik LSTM satu <i>single layer</i> dan model <i>word2vec</i> yang diuji pada dua <i>benchmark</i> dataset, <i>rate</i> error dari SSTb dataset 14,3% unggul dari tujuh metode lain, dan <i>rate</i> error pada IMDB dataset 11,32% unggul dari sepuluh metode lain
3	Wang & Cao (2017)	<i>Long Short Term Memory (LSTM)</i>	Dalam percobaan Tiga Model LSTM (LSTM, LSTM dengan L2 dan Nadam, BLSTM). Peneliti membandingkan akurasi dari tiga sentimen teks model analisis, termasuk berbasis LSTM
4	Bin, et al (2018)	<i>Long Short Term Memory (LSTM)</i>	Dalam penelitian ini dilakukan perbandingan dengan SVM dann IRNLP tools, dimana LSTM menunjukkan hasil lebih baik. Eksperimen menunjukkan metode ini efektif dan akurasi algoritma LSTM telah mencapai 93,51% pada kumpulan data klasifikasi sentimen campuran
5	Mirza & Cosan (2018)	<i>Long Short Term Memory (LSTM)</i>	Dalam penelitian ini digunakan juga menggunakan versi LSTM, GRU, Bi-LSTM dan Jaringan Saraf Tiruan. Melalui serangkaian eksperimen yang komprehensif, dalam penelitian ini ditunjukkan bahwa kerangka kerja

No	Peneliti (Tahun)	Metode	Keterangan
6	Ciftci & Apaydin (2018)	<i>Long Short Term Memory (LSTM)</i>	deteksi intrusi berurutan yang kami usulkan berkinerja baik dan dinamis, kuat dan terukur. Eksperimen ini menunjukkan LSTM dengan max pooling dan jaringan Deep Auto LSTM menunjukkan skor f1 terbaik. Menerapkan teknik RNN menggunakan LSTM kemudian dibandingkan dengan <i>Logistic Regression</i> dan <i>Naive Bayes</i> untuk analisis sentimen berbahasa Turki menggunakan data yang diambil dari situs belanja dan film. Hasil yang didapat, LSTM memiliki akurasi yang lebih baik dari metode lainnya dengan nilai validasi akurasi, pengujian akurasi, presisi dan <i>recall</i> masing-masing adalah 83,3%, 82,9%, 0,86, 0,83
7	Naili, M. et. al. (2017)	<i>LSA, Word2Vec, dan GloVe</i>	Hasil penelitian menunjukkan bahwa Word2Vec dan GloVe lebih efektif daripada LSA untuk kedua bahasa. Apalagi dibandingkan dengan GloVe, Word2Vec menghadirkan representasi vektor kata terbaik dengan ruang semantik dimensi kecil.

BAB 3

METODOLOGI PENELITIAN

Penelitian ini dilakukan untuk mendapatkan hasil analisis sentimen dengan hasil kinerja yang baik atau memiliki akurasi yang tinggi. Pengklasifikasian disini menggunakan metode *Long Short- Term Memory*. Metodologi penelitian diawali dengan data yang digunakan, arsitektur umum penelitian, alur pengklasifikasian, dan analisis kinerja pengklasifikasian.

3.1. Data yang digunakan

Pengambilan data dilakukan dengan melakukan *web scraping* pada situs ulasan (*review*) mengenai sebuah aplikasi yang terdapat dalam Google Play. Ulasan-ulasan tersebut merupakan komentar para pengguna mengenai sebuah aplikasi dimana komentar tersebut bermuatan positif dan negatif. Selain memberikan komentar, pengguna juga dapat memberikan rating berupa bintang terhadap aplikasi itu yang berkaitan dengan komentar yang telah diberikan. Contoh ulasan pada Google Play dapat dilihat pada gambar 3.1.



Gambar 3. 1 Contoh ulasan di situs *Google Play*

Berdasarkan hasil *web scraping* yang dilakukan, telah dikumpulkan ulasan yang berasal dari beberapa aplikasi yang berkategori sosial dan komunikasi. Ulasan-ulasan yang diambil adalah ulasan yang berbahasa Indonesia. Total hasil scraping dari situs Google Play sebanyak 347.510 ulasan. Rincian data yang diambil ditampilkan pada table 3.1 dibawah ini.

Tabel 3. 1 Data Ulasan yang diambil dari Google Play

No	Nama Aplikasi	Id Aplikasi	Kategori	Jumlah Data
1	Facebook	com.facebook.katana	Sosial	20.878
2	Facebook Lite	com.facebook.lite	Sosial	24.078
3	Google Hangout	com.google.android.talk	Komunikasi	20.877
4	Google+	com.google.android.apps.plus	Sosial	20.880
5	Instagram	com.instagram.android	Sosial	24.079
6	KakaoTalk	com.kakao.talk	Komunikasi	20.880
7	Line	jp.naver.line.android	Komunikasi	20.720
8	Line Lite	com.linecorp.linelite	Komunikasi	18.785
9	Messenger	com.facebook.orca	Komunikasi	20.877
10	Messenger Lite	com.facebook.mlite	Komunikasi	19.277
11	Skype	com.skype.raider	Komunikasi	16.080
12	Snapchat	com.snapchat.android	Komunikasi	20.876
13	Telegram	org.telegram.messenger	Komunikasi	8.080
14	Tumblr	com.tumblr	Sosial	6.959
15	Twitter Lite	com.twitter.android.lite	Sosial	2.280
16	Twittter	com.twitter.android	Sosial	20.875
17	WeChat	com.tencent.mm	Komunikasi	20.878
18	WhatsApp	com.whatsapp	Komunikasi	24.075
19	WhatsApp Bussiness	com.whatsapp.w4b	Komunikasi	16.076
Total Hasil Web Scrapping				347.510

Dari total 347.510 data ulasan tersebut, dilakukan penggabungan menjadi 1 dataset yang berisi nama pengguna, ulasan, rating, dan tanggal ulasan tersebut diberikan. Data-data tersebut diklasifikasikan dengan cara menentukan berdasarkan rating yang terdapat pada ulasan tersebut. Untuk ulasan yang memiliki rating 1 dan 2 di klasifikasikan sebagai ulasan dengan sentimen negatif. Kemudian ulasan yang memiliki rating 4 dan 5 diklasifikasi sebagai ulasan dengan sentimen positif. Data dengan rating

3 tidak digunakan, karena memiliki bias apakah positif atau negatif. Hasil klasifikasi dengan rating tersebut kemudian di cek secara manual untuk menghindari rating yang tidak cocok terhadap sentimen pada ulasan tersebut. Ulasan yang tidak sesuai rating akan diubah menjadi sentimen sesungguhnya yang terkandung pada ulasan tersebut.

Setelah itu semua ulasan akan digabung menjadi sebuah dataset. Setelah proses penggabungan tersebut maka dilakukan penghapusan terhadap data yang tidak memiliki ulasan (*empty string*) dan penghapusan data yang memiliki rating 3. Total data yang data yang didapatkan sebanyak 262.555. Rincian jumlah data tersebut dapat dilihat pada Tabel 3.2 dibawah ini.

Tabel 3. 2 Rincian Jumlah Dara Berdasarkan Jenis Sentimen

Sentimen	Jumlah Data
Positif	155.528
Negatif	107.027

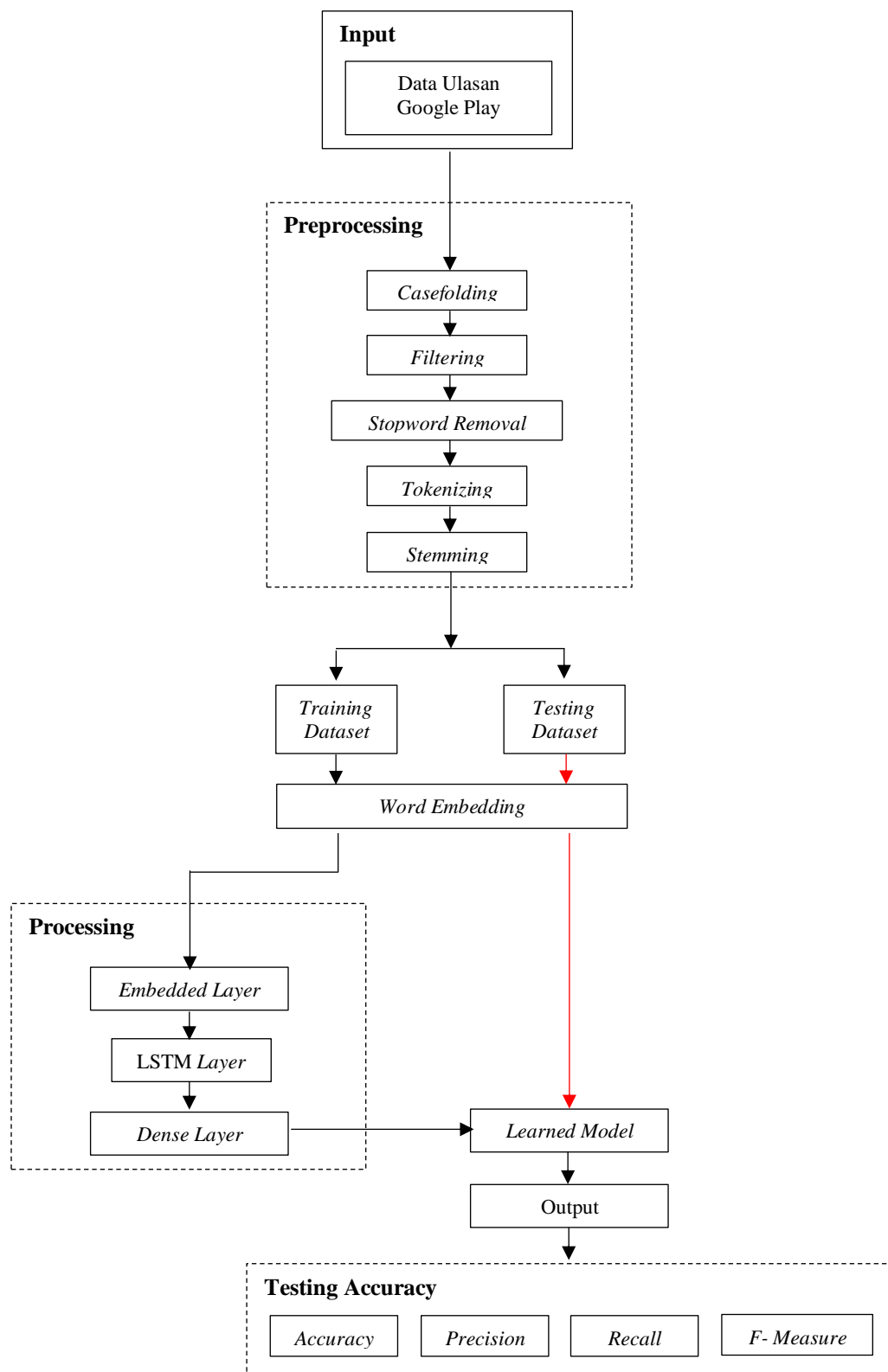
Contoh data yang diperoleh dan telah ditentukan sentimen yang terkandung dalam data tersebut ditunjukkan padan Tabel 3.3 dibawah ini.

Tabel 3. 3 Contoh Data Ulasan dengan jenis Sentimen

No	Nama Pengguna	Rating	Ulasan	Sentimen
1	Moch Nur Amien	1	Wa update terbaru lemot, ketika diklik wa ada layar baru muncul layar whattapp from facebook baru menu kunci...biasanya langsung menu kunci..ini ada muncul menu yg tak perlu spt diatas.	Negatif
2	Iswanto Radjak	2	Bug chatting yg masuk ketika di buka tidak ada, nama pengirim ada tapi isi chatting tidak bisa muncul. Kecewa	Negatif
3	Bocil338	5	Lancar dan smakin cepat dlm berkirimunikasi dgn kluarga dan relasi	Positif

3.2. Arsitektur Umum

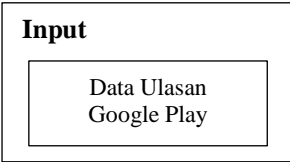
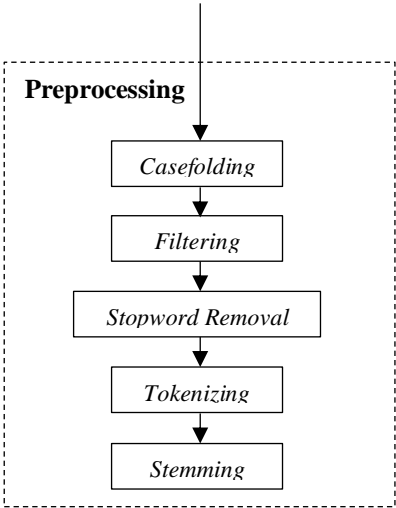

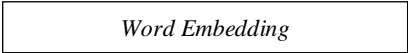
Arsitektur umum penelitian untuk peningkatan kinerja deep learning dalam analisis sentimen dapat dilihat pada Gambar 3.1.

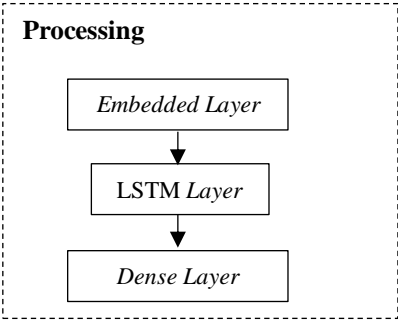

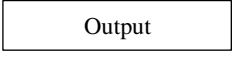
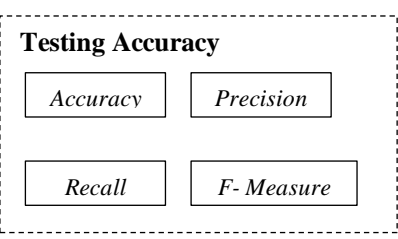


Gambar 3. 2 Arsitektur Sistem

Gambar 3.2 merupakan gambaran arsitektur umum penelitian kinerja deep learning dalam pengklasifikasian sentimen menggunakan arsitektur LSTM dengan pemilihan *word embedding* dan *optimizer*. Pada tahap awal dilakukan pengumpulan data yang berkaitan dengan data yang akan diteliti, dimana dalam hal ini adalah data ulasan yang diambil dari Google Play. Kemudian dilakukan pembersihan dari data kosong serta transformasi data yaitu mengubah *rating* menjadi label sentimen apakah positif atau negative. Data yang sudah diberi label ini akan menjadi Input ke dalam sistem. Penjelasan mengenai arsitektur diatas akan dijelaskan pada tabel dibawah ini.

Tabel 3. 4 Penjelasan Blok Diagram Arsitektur Sistem

Blok Diagram	Penjelasan
	<p>Input pada sistem ini adalah data ulasan yang diambil dari Google Play yang telah dibagi menjadi dua sentimen, positif atau negatif.</p>
	<p>Input yang telah masuk, akan diproses pada tahapan preprocessing. Tahapan ini dilakukan agar data yang dimiliki bersih dari <i>noise</i>. Pada tahapan ini ada beberapa proses yang dilewati yaitu:</p> <ol style="list-style-type: none"> 1. <i>Casefolding</i>: proses pengubahan bentuk tulisan menjadi lowercase 2. <i>Filtering</i>: proses penghapusan tanda baca, dan link. 3. <i>Stopword Removal</i>: penghapusan kata-kata yang tidak ada pengaruhnya terhadap sentiment 4. <i>Tokenizing</i>: pemenggalan kata menjadi satu suku kata 5. <i>Stemming</i>: pengubahan kata menjadi kata dasarnya <p>Penjelasan lengkap mengenai proses ini ada pada bagian 3.3.</p>
	<p>Hasil Preprocessing akan menghasilkan data yang bersih (<i>data clean</i>). Data bersih ini akan dibagi mejadi dua bagian yang itu data pelatihan (Training Dataset) dan data pengujian (Testing Dataset)</p>
	<p>Word Embedding: Data pelatihan dan data pengujian akan direpresentasikan menjadi vector. Pada tahapan ini akan menubah data training menjadi vector berupa <i>sequence</i></p>

Blok Diagram	Penjelasan
	<p>of Integer yang digunakan sebagai input pada jaringan saraf. Selain itu, pada tahap ini juga masing-masing data akan diproses dengan Word2Vec yang kan digunakan sebagai bobot pada <i>Embedded Layer</i>.</p> <p>Penejelasan lengkap mengenai proses ini dapat dilihat pada bagian 3.4.</p>
<p>Processing</p>  <pre> graph TD A[Embedded Layer] --> B[LSTM Layer] B --> C[Dense Layer] </pre>	<p>Pada tahapan ini, Data training akan digunakan sebagai Input pada bagian processing (pelatihan menggunakan jaringan RNN – LSTM). Pada jaringan ini terdapat beberapa lapisan jaringan, yaitu:</p> <ol style="list-style-type: none"> 1. Embedded Layer: yaitu lapisan dengan input yang berasal dari vektor data training 2. LSTM Layer: lapisan lanjutan yang melakukan perhitungan LSTM. 3. Dense Layer: lapisan yang merupakan hasil dari proses yang menentukan apakah sebuah input terprediksi positif atau negatif. <p>Penejelasan lengkap mengenai proses ini dapat dilihat pada bagian 3.5.</p>
 <pre> graph TD A[Learned Model] </pre>	<p><i>Learned Model</i>: model yang dihasilkan dari proses pelatihan, yang akan digunakan untuk melakukan pengujian data tes.</p>
 <pre> graph TD A[Output] </pre>	<p>Hasil keluaran berupa hasil pengujian</p>
<p>Testing Accuracy</p>  <pre> graph TD subgraph AccuracyMetrics [Testing Accuracy] A[Accuracy] B[Precision] C[Recall] D[F-Measure] end </pre>	<p>Pada tahapan ini akan dilihat nilai Akurasi, Precision, Recall dan F-Measure pada masing-masing precobaasn yang dilakukan. Hasil akhir pengklasifikasian nantinya akan dianalisis masing-masing kinerjanya.</p>

3.3. Preprocessing

Pada tahapan ini dilakukan beberapa proses pemrosesan awal, dimana data ulasan yang telah dimasukkan akan melewati beberapa proses yang akan menghasilkan sebuah data yang bersih untuk diolah ke proses selanjutnya.

3.3.1. Case Folding

Pada tahapan ini seluruh teks pada dataset akan diubah menjadi huruf kecil (*lowercase*) agar memudahkan proses pengolahan selanjutnya. Contoh hasil dari proses *case folding* dapat dilihat pada Tabel 3.4.

Tabel 3. 5 Contoh Hasil Proses Case Folding

Sebelum Case Folding	Setelah Case Folding
Kecewa, masak font nya jdi besar-besar wlpun ukuran font udh kecil ,, tolong dong di perbaiki lagi	kecewa, masak font nya jdi besar-besar wlpun ukuran font udh kecil ,, tolong dong di perbaiki lagi

3.3.2. Filtering

Pada tahap *filtering* ini dilakukan pembersihan data dari tanda baca seperti (!"#\$%&'()*+,-./:;<=>?@[\\]^_`{|}~) yang akan diganti dengan karakter spasi. Penghapusan tanda baca ini dilakukan karena pada proses pelatihan tanda baca akan dihiraukan sehingga proses pelatihan akan menjadi lebih sederhana. Contoh dari *filtering* dapat dilihat pada Tabel 3.5.

Tabel 3. 6 Contoh Filtering

Sebelum Filtering	Setelah Filtering
kecewa, masak font nya jdi besar-besar wlpun ukuran font udh kecil ,, tolong dong di perbaiki lagi	kecewa masak font nya jdi besar-besar wlpun ukuran font udh kecil tolong dong di perbaiki lagi

3.3.3. Stopword Removal

Tahap pra-poses selanjutnya adalah menghapus kata-kata yang tidak ada kaitannya terhadap nilai sentimen. Stopword adalah kumpulan kata yang bukan merupakan ciri atau kata unik yang terdapat dalam sebuah dokumen (Dragut et al. 2009). Kata-kata tersebut antara lain misalnya kata sambung "dan", " atau" atau kata-kata seperti “dalam”, “selalu”, “oleh”, “bagian”, “karena”, “sekalian”, “sekaligus”, “jika” dan lain sebagainya yang tidak ada pengaruhnya terhadap sentimen. Sebelum tahap ini diterapkan, terlebih dahulu dibuat daftar kata yang termasuk *stopword* yang dinamakan *stoplist*. Jika kata-kata yang kita peroleh pada tahap sebelumnya terdapat di dalam daftar *stoplist* maka kata-kata pada teks tersebut akan dihapus sehingga kata-kata yang tidak dihapus dapat dianggap yang mencirikan deskripsi dari dokumen tersebut.

Tabel 3. 7 Contoh Stopword

dan	atau	di	ini
itu	bagian	bukan	lalu
oleh	kepada	yaitu	karena
sekaligus	sekalian	tidak	jika

Tabel 3. 8 Contoh Penerapan Stopword Removal

Sebelum <i>Stopword Removal</i>	Setelah <i>Stopword Removal</i>
kecewa masak font nya jdi besar-besar wlpun ukuran font udh kecil tolong dong di perbaiki lagi	kecewa masak font jdi besar-besar wlpun ukuran font udh tolong perbaiki

3.3.4. Stemming

Stemming merupakan salah satu tahapan dalam pra proses pengolah teks. *Stemming* bertujuan untuk mengubah sebuah kata menjadi asal kata (*root word*). Semua imbuhan kata apakah itu awalan kata (*prefixes*), sisipan kata (*infixes*), akhiran kata (*suffixes*) akan dihilangkan. Selain imbuhan juga akan dihilangkan kata-kata turunan yang memiliki awalan dan akhiran (*confixes*). Untuk itu dalam penelitian ini akan digunakan Algoritma stemming untuk mencari akar sebuah kata. Salah satu algoritma *stemming* untuk Bahasa Indonesia yaitu Algoritma Nazief dan Adriani (Adriani, et al.,2007). Tahapan algoritma tersebut meliputi:

1. Awal proses dilakukan pemeriksaan terhadap token sebuah kata apakah kata tersebut terdapat dalam daftar kata dasar yang telah disimpan sebelumnya. Jika kata tersebut ada pada daftar tersebut, maka pada tahap ini proses berhenti sehingga kata tersebut memang benar merupakan sebuah kata dasar,
2. Inflection Suffixes yang ditambahkan pada akhir kata sebagai variasi kata dihapus. Inflection Suffixes antara lain seperti akhiran (“-kah”, “-lah”, “-tah”, “-nya”, “-mu”, atau “-ku”). Kalau kata tersebut mengandung partikel (“-kah”, “-tah”, “-lah” atau “-pun”) maka langkah ini diulangi lagi untuk menghapus kata ganti kepunyaan yaitu (“-mu”, “-ku”, atau “-nya),
3. Menghapus imbuhan turunan -i, -kan, -an.

4. Menghapus awalan turunan be-, di-, ke-, me-, pe-, se- dan te-.
5. Bila pada langkah sebelumnya di atas kata tersebut belum dapat ditemukan, maka selanjutnya kata tersebut dianalisis dan di cek apakah masuk dalam tabel diambiguitas pada kolom terakhir atau tidak.
6. Bila semua langkah telah dilewati namun tidak dapat ditemukan maka kata tersebut diasumsikan sebagai kata dasar. Proses selesai.

Tabel 3. 9 Contoh Penerapan Stemming

Sebelum <i>Stopword Removal</i>	Setelah <i>Stopword Removal</i>
kecewa masak font jdi besar-besar wlpun ukuran font udh tolong perbaiki	kecewa masak font jdi besar wlpun font udh tolong perbaiki

3.3.5. Tokenizing

Tokenizing (tokenisasi) adalah sebuah proses mengurai konten teks menjadi kata-kata, istilah, simbol, atau elemen-elemen yang menyusun sebuah teks. Uraian tersebut disebut token. Pada proses ini akan dihilangkan karakter seperti spasi, titik (.), koma (,), dan karakter lain yang digunakan sebagai pemisah dari kata-kata tersebut. Daftar token ini akan digunakan sebagai input untuk pemrosesan selanjutnya untuk representasi teks (Weiss et al, 2005).

Tabel 3. 10 Contoh Penerapan Tokenizing

Sebelum <i>Tokenizing</i>	Setelah <i>Tokenizing</i>
kecewa masak font jdi besar wlpun ukuran font udh tolong perbaiki	“kecewa” “masak” “font” “jdi” “besar” “wlpun” “ukuran” “font” “udh” “tolong” “perbaiki”

3.4. Representasi Teks

3.4.1. Word sequence

Data ulasan yang telah dibersihkan, kemudian diubah formatnya menjadi tensor. Pada arsitektur jaringan saraf, lapisan *embedding* adalah lapisan yang menerima masukan berupa tensor integer dua dimensi. Dalam penelitian ini digunakan 59761 kata paling tinggi frekuensinya dalam dataset. Maksimal jumlah kata dari masing-masing ulasan ditentukan sebanyak 300 kata. Kemudian 59761 kata tersebut diubah menjadi urutan

kata yang paling sering muncul dalam bentuk integer. Urutan tersebut diubah menjadi tensor 2 dimensi dengan ketentuan (59761, 300) yang artinya 59761 urutan tersebut masing-masing memiliki panjang 300. Semua urutan yang akan menjadi input di lapisan *embedding* harus mempunyai panjang yang sama. Untuk itu dilakukan pemampatan sequences apabila kurang dari 300 maka sequence akan dihapuskan, dan apabila sequences lebih dari 300 maka akan dipotong dengan batas maksimal 300.

Tabel 3. 11 Contoh Penerapan Representasi Teks

[illegible]

47), ('jdi', 48), ('besar', 49), ('wlpun', 50), ('ukur', 51), ('udh', 52), ('tolong', 53), ('perbaiki', 54)]

Setelah dilatih, data training akan dibuat menjadi sebuah matriks, dimana tiap kata akan dicari nilai hubungannya sesuai dengan ukuran dimensi yang digunakan. Berikut ini contoh matriks word embedding dengan dimensi 300 kata

```
[ 1.76025495e-01  1.45162758e-03 -6.52425513e-02 -1.19027995e-01 -9.12725329e-
02      2.48387545e-01  -1.66925371e-01  -3.44547182e-02      9.87227112e-02
6.61438927e-02  6.75340444e-02  1.34137467e-01  9.02777910e-02  4.06420976e-02
-5.14076697e-03  2.04617277e-01  -9.20221657e-02  4.09860769e-03  5.53923063e-
02      3.08091611e-01      2.03872249e-01  5.03645651e-03  2.12446414e-02  -
8.21791887e-02 -3.64309619e-03  6.02244772e-02  6.97885007e-02  6.12235256e-02
1.39239550e-01 -1.38985692e-02 -6.50061481e-03 -4.69606109e-02  6.34659082e-02
2.09601875e-02 -1.28830150e-01 -8.75408873e-02  7.03612566e-02 -6.63237125e-02
-8.41983929e-02 -8.55797455e-02 -3.47764753e-02  2.05557365e-02  3.96264084e-
02 -8.47397372e-02 ----- 1.79924592e-01 -5.67817278e-02  1.61408588e-01
2.27058232e-01 -9.47487876e-02  1.35362178e-01 -5.90828136e-02 -2.54610199e-02
-8.42407867e-02 -3.39236446e-02  1.04971528e-02  1.83498308e-01 -1.28874620e-
02      -3.13725621e-02      -9.59494859e-02      -5.69281727e-02      5.03955083e-03
2.36436222e-02 -2.91628437e-03  5.15652413e-04  8.24684836e-03  8.18296745e-02
2.81751174e-02 -1.79979697e-01 -6.52367920e-02 -7.46349022e-02 -1.91826560e-02
4.88320999e-02  1.77191943e-01 -6.48554191e-02 -8.50464106e-02  2.76168227e-01
-2.11387686e-02 -9.67477486e-02 -1.34055791e-02  1.82283327e-01  1.99085981e-
01      2.28512734e-02      5.04344888e-02      -7.16583654e-02      1.85523495e-01  -
2.68569380e-01 -2.23791242e-01  9.68608856e-02 -1.63236391e-02 -3.56477022e-01
-1.33099288e-01 -5.33914752e-03 -2.22320303e-01 -8.25001672e-02  3.77346903e-
01      -2.42939085e-01      -2.29068324e-02      -3.13435942e-01      -1.76251382e-01
2.53490746e-01 -2.36540273e-01 -4.60298359e-01  1.03812125e-02 -1.98509187e-01
-2.58283794e-01 -2.42506251e-01  3.34274024e-01 -7.66629651e-02 -4.50382605e-
02      9.94471535e-02      -1.42872304e-01      7.31554180e-02      5.33933938e-02  -
3.30376774e-02 -1.12997279e-01  1.63604766e-01 -9.73658636e-02 -1.08879112e-01
1.27531350e-01  4.42157835e-02  1.00563448e-02 -3.40548642e-02 -1.87589098e-02
```

1.63229719e-01 -5.88270687e-02 -3.09202392e-02 1.80328965e-01 -3.93301360e-02]

Dengan matriks tersebut akan dijadikan bobot dalam lapisan *Embedding* yang digunakan oleh model LSTM untuk melakukan training terhadap dataset ulasan.

3.5. Arsitektur Jaringan

Arsitektur jaringan dibentuk untuk menghasilkan akurasi yang optimal. Secara umum, model pelatihan dapat memiliki berbagai jumlah lapisan, tetapi dalam penelitian ini terdiri dari empat lapisan yaitu lapisan Embedding, lapisan RNN-LSTM, satu lapisan Dense dengan berbagai fitur input. Dataset keseluruhan dibagi tiga menjadi data pelatihan, data validasi, dan data tes. Selain itu, algoritma optimisasi Adam dan binary cross entropy loss masing-masing digunakan untuk fungsi optimizer dan loss. Untuk aktivasi a sigmoid dan fungsi ReLU digunakan.

Untuk melatih model tidak hanya fungsi *optimizer* atau *loss* tetapi juga *batch-size* dan jumlah *epoch* adalah signifikan, di mana satu *epoch* adalah ketika seluruh dataset dilewatkan jaringan saraf maju (*forward*) dan mundur (*backpropagation*) hanya sekali dan ukuran batch adalah jumlah total contoh pelatihan yang disajikan dalam satu kelompok. Alasan membagi satu *epoch* ke dalam kelompok-kelompok yang lebih kecil adalah bahwa ia memiliki ukuran terlalu besar untuk perhitungan secara keseluruhan. Namun, jumlah *epoch* dan *batch-size* yang tepat tidak diketahui dan dapat bervariasi dari domain ke domain, dari dataset yang lebih besar ke dataset yang lebih kecil.

Dalam penelitian ini, jumlah *epoch* dan *batch-size* ditetapkan, yaitu 100 epoch dan 1024 batchsize yang digunakan. Alasan menggunakan epoch yang kecil adalah untuk menghindari overfitting. Kendala yang sering terjadi pada model arsitektur LSTM adalah kondisi model yang overfitting dimana nilai *accuracy* dan *loss* pada saat pelatihan berbeda saat validasi.

Kinerja pada data training akan terlihat selalu meningkat tetapi pada beberapa titik tertentu walaupun dengan jumlah epoch yang sama terjadi penurunan kinerja pada saat validasi. Untuk mendapatkan kinerja model akan dilakukan usaha dengan *tuning hyperparameter* agar menghasilkan kinerja yang *good fit*. Dalam penelitian ini akan menggunakan lapisan *Dropout* dan fungsi *Callback*. Lapisan *Dropout* ini digunakan

untuk mengurangi *overfitting* pada model LSTM. Sementara itu, fungsi *Callback* akan menghentikan proses pelatihan apabila terjadi *overfitting* meskipun belum mencapai nilai epoch maksimum yang sudah ditetapkan diawal.

BAB 4

HASIL DAN PEMBAHASAN

Bab ini membahas hasil pre-processing data, pelatihan data (*training*), dan pengujian data (*testing*) dalam analisis sentimen menggunakan *deep learning* dengan arsitektur RNN-LSTM dengan penambahan word embedding serta pemilihan *optimizer*. Simulasi analisis dilakukan dengan dua metode yang berbeda, yang pertama tanpa menggunakan word embedding, dan yang kedua dengan menggunakan word embedding.

4.1. Spesifikasi perangkat yang digunakan

Pada bagian ini dilakukan pemaparan spesifikasi perangkat keras (*hardware*) dan perangkat lunak (*software*) yang digunakan. Dalam menjalankan teknik Deep Learning, diperlukan spesifikasi perangkat tersendiri dalam memudahkan pemrosesannya, hal ini dikarenakan kedalaman pemrosesan tentunya memakan resource yang besar. Penulis menggunakan layanan Google Colaboratory, atau "Colab" yang merupakan produk dari Google Research. Colab memungkinkan siapa saja untuk menulis dan mengeksekusi kode python melalui browser, dan sangat cocok untuk pembelajaran mesin, analisis data, dan pendidikan. Spesifikasi perangkat yang digunakan dapat dilihat pada Tabel 4.1.

Tabel 4. 1 Spesifikasi Perangkat yang digunakan

Kriteria	Spesifikasi
<i>Cloud Workstation</i>	<i>Google Colaboratory Jupyter Notebook</i>
<i>Processor</i>	<i>Intel(R) Xeon(R) CPU @ 2.30GHz</i>
<i>Memory</i>	<i>12,6 GB VRAM</i>
<i>GPU</i>	<i>Tesla P100-PCIE-16GB, Cuda Cores: 2496</i>
<i>Harddisk</i>	<i>33 GB</i>
Bahasa Pemrograman	<i>Python-3.6</i>
Pustaka	<i>PySastrawi-1.2.0, Gensim-3.6.0, Tensorflow-2.2.0, dan Keras-2.3.1</i>

4.2. Hasil Preprocessing Data

Dataset ulasan yang telah dikumpulkan, telah dilakukan proses pre-processing yaitu mulai dari *case folding*, *filtering*, *stopword removal*, *stemming* dan *tokenizing*. Kemudian dataset yang telah melalui proses tersebut di cek kembali, dilakukan penghapusan baris dari datase yang terdapat data yang kosong dan hanya memiliki satu kata saja. Hasil dari tahapan preprocessing dan pelabelan seperti pada Tabel 4.2.

Tabel 4. 2 Hasil preprocessing data

No	Nama Pengguna	Sentimen	Ulasan
1	Jas Hujan	POSITIF	aplikasi bagus kurang lapor akun halaman grup bau keras ujar benci spam dll akun halaman grup biar tengkar debat dll muncul facebook ulasan lengkap
2	Zainal Latulanit	NEGATIF	tolong facebook indonesia opsi pesan kirim facebook pilih nomor cantum akun facebook rata rata nomor akun facebook mati kadaluwarsa ganti nomor ulasan lengkap
3	Muhammad Faris	POSITIF	bagus vidio call sang bagus keren banget wajib coba pakai aplikasi bagus donlot aplikasi rugi karena gambar vidio postingn teman bintang kagum aplikasi ulasan lengkap
4	Bungsu Pampo	NEGATIF	aplikasi baru capek downloadnya systemnya alih akun pemberitahuan fitur perangkat aktif systemnya yg bikin halaman nonton transfer 50 000 promosi saldo ludes gk tw kmn no rekening fb bener saldo gk akun fbq blokir fb maaf 50 000 miskin harga bahagia ulang ulasan lengkap
5	GHIEANTZ GAGA	NEGATIF	aplikasi ik facebook nonaktifakn diberitahu detail salah facebook facebook beritahh salah langgar laku detail kirim link postingan langgar ulasan lengkap
6	Attieng	NEGATIF	selamat malam kecewa foto2 hilang nohon kembali karna situ kenang yg simpan sengaja simpan fb biar anak cucu hilang
7	Islachul Majid	POSITIF	

No	Nama Pengguna	Sentimen	Ulasan
8	DHANIEL GAMING	POSITIF	selamat siang facebook yg facebook ambil hack ambil sednagkan email sandi ganti si curi tolong jawab
9	Omi Pnd	POSITIF	fb bayar buka fb gratis harap beda beda gak liat fto video doang bayar publikasi luas jangkau tayang beranda teman update video publikasi sempit fb gratis inboxnya yg gtatis ulas lengkap
10	TLL PROJECT	NEGATIF	akun buka akun tulis waktu login habis login akun baik
...
241852	Indra Wati	NEGATIF	1 klu bagus ru aq tambain
241853	Angkur Dominique	POSITIF	uap kren mantap
241854	Pengguna Google	NEGATIF	lamo nian donload nyo paket udah full ni
241855	Pengguna Google	NEGATIF	telegram ferivikasi
241856	Firqindziazmi	POSITIF	canda bareng aja skuy
241857	Andre Setyawan	NEGATIF	bosok tenan asu nho telfon mati mati ae cok
241858	Dedeh Nurani	POSITIF	saran kirim foto grup ppsu
241859	Ruli Yawati	POSITIF	coba bagus
241860	Lisyanti Rumlan	POSITIF	ter download
241861	Pengguna Google	POSITIF	moga manfaat

4.3. Arsitektur Jaringan

Berikut ini rincian arsitektur jaringan RNN-LSTM yang digunakan, yaitu:

1. Layer pertama yang dibuat adalah Embedding Layer yang menggunakan vector dengan panjang dimensi 100, 200, atau 300 kata untuk merepresentasikan masing-masing kata.
2. Sebuah hidden layer dengan 300 neuron sebagai dropout layer
3. LSTM layer dengan 100 neuron.
4. Layer terakhir adalah Dense Output Layer dengan 1 neuron dan fungsi aktivasi menggunakan sigmoid.

Dikarenakan penelitian merupakan *binary classification* yaitu sentimen positif atau negatif, maka digunakan *loss function binary_crossentropy* (keras) dan dengan fungsi optimasi menggunakan 'Adam'. Batch size yang digunakan adalah 1024 dengan

epochs berjumlah 100 dan parameter evaluasi model adalah validasi akurasi ‘val_accuracy’. Untuk menghindari masalah utama dalam LSTM yaitu overfitting, pada penelitian menggunakan Early Stopping function yang menghentikan proses training apabila akan terjadi overfitting pada model yang dibuat.

4.4. Hasil Pemrosesan Data

Subbab ini memaparkan hasil dari pemrosesan data, yaitu hasil training data dan hasil testing data dari klasifikasi dengan epoch maximum sebesar 100 menggunakan beberapa skema pengujian beberapa arsitektur jaringan yaitu:

1. Tanpa menggunakan Word2Vec
2. Menggunakan Word2Vec dengan dimensi 200, 250, dan 300 Kata
3. Membandingkan penggunaan Fungsi Aktivasi dan Optimizer pada jaringan

4.4.1. Hasil training dan testing LSTM tanpa Word2Vec

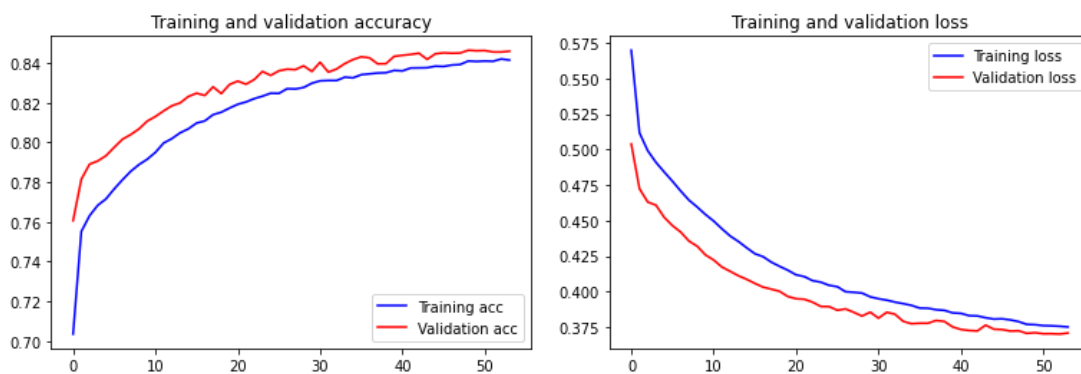
Proses training dengan teknik LSTM tanpa menggunakan Word2Vec dengan data ulasan Google Play sebanyak 210.044 data, dan menggunakan spesifikasi jaringan seperti pada subbab sebelumnya. Hasil error training (*loss*) dan akurasi data menggunakan teknik LSTM tanpa menggunakan Word2Vec dapat dilihat pada Tabel 4.3 dibawah ini.

Tabel 4. 3 Hasil Training LSTM tanpa Word2Vec

Epoch	Akurasi	Error	Val Akurasi	Val Loss
1	70,35%	0,5698	76,06%	0,5039
2	75,51%	0,5116	78,14%	0,4723
3	76,30%	0,4991	78,90%	0,4630
4	76,83%	0,4909	79,06%	0,4608
5	77,15%	0,4841	79,33%	0,4522
6	77,66%	0,4776	79,75%	0,4464
7	78,11%	0,4707	80,16%	0,4419
8	78,54%	0,4643	80,39%	0,4355
9	78,88%	0,4595	80,68%	0,4319
10	79,15%	0,4542	81,07%	0,4258
...
50	84,08%	0,3767	84,61%	0,3711
51	84,10%	0,3760	84,62%	0,3703

Epoch	Akurasi	Error	Val Akurasi	Val Loss
52	84,08%	0,3759	84,55%	0,3704
53	84,20%	0,3756	84,55%	0,3701
54	84,14%	0,3751	84,59%	0,3709

Dari Tabel 4.3 diatas, dapat diketahui bahwa semakin banyak perulangan atau epoch yang dilakukan, nilai error yang dihasilkan makin kecil, seperti pada epoch ke-5 error yang didapat sebesar 0,4841, selanjutnya pada epoch ke-10 error yang didapat sebesar 0,4542 dan seterusnya sampai pada epoch maximum dimana training akan berhenti oleh fungsi Callback untuk mencegah overfitting, sehingga training berhenti pada epoch ke-54 dengan error yang didapat sebesar 0,3751. Dari tabel tersebut juga dapat dilihat hasil training menghasilkan tingkat akurasi yang baik hingga 84,14% dengan error yang sangat kecil sekitar 0,3751. Proses training tersebut berlangsung selama 1 jam 25 menit. Perubahan error serta akurasi dari epoch ke-1 sampai epoch ke-54 dapat dilihat dalam grafik, grafik hasil error dan akurasi training jaringan LSTM tanpa Word2Vec dapat dilihat pada Gambar. 4.1 dibawah ini.



(a) Grafik Akurasi LSTM

(b) Grafik Loss (Error) LSTM

Gambar 4. 1 Hasil Training LSTM Tanpa Word2Vec

Dari gambar 4.1. diatas kita dapat melihat validasi terhadap akurasi dan validasi terhadap error (loss) jg baik, sehingga model ini cukup baik kita gunakan untuk melakukan analisis terhadap sentimen data ulasan Google Play.

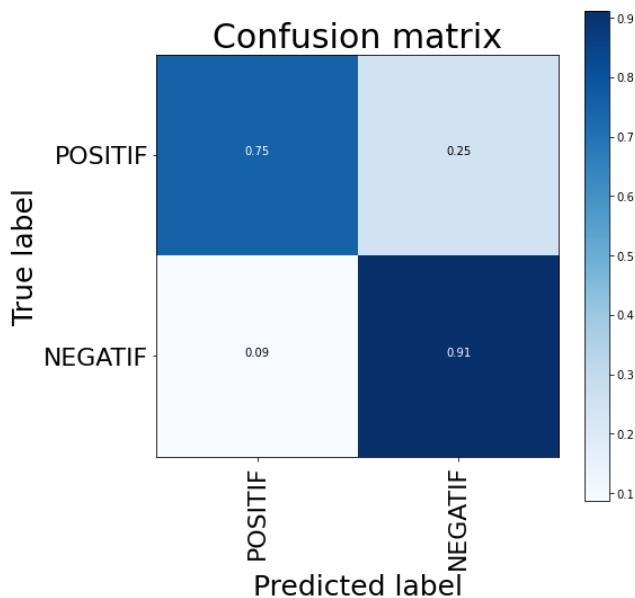
Dengan model yang telah dihasilkan pada proses training tersebut, dilakukan *testing* menggunakan data test yang telah dibagi dari dataset sebelumnya, sebesar 52.511 data. Dari proses testing yang dilakukan, model menghasilkan nilai akurasi

sebesar 84,68 % dengan error sebesar 0,3667. Kinerja model ini dapat dilihat berdasarkan *confussion matrix* nya pada table 4.4 dibawah ini.

Tabel 4. 4 Tabel Confusion matrix hasil testing model LSTM tanpa Word2Vec

Kelas	Positif	Negatif
Positif	28.432	2.740
Negatif	5.300	16.039

Dari *confusion matrix* di Tabel 4.4 diatas dapat dijelaskan bahwa model mengklasifikasikan secara benar sebesar 28.432 data sebagai positif dan 16.039 data sebagai negatif. Selain itu model salah dalam memprediksi 2.740 data ke dalam data negatif yang seharusnya positif (*false negative*), serta salah dalam memprediksi 5.300 data ke dalam data positif yang seharusnya negatif (*false positive*). Untuk plot persentase dari confusion matrix hasil testing model LSTM tanpa Word2Vec dapat dilihat pada Gambar 4.2 dibawah ini.



Gambar 4. 2 Confusion Matriks dari Hasil Testing LSTM Tanpa Word2Vec

Berdasarkan *confusion matrix* hasil testing model diatas, kita dapat mengukur kinerja dari model tersebut berdasarkan laporan klasifikasi yang dapat dilihat pada tabel 4.5 dibawah ini.

Tabel 4. 5 Kinerja klasifikasi model LSTM tanpa Word2Vec

Sentimen	Precision	Recall	F1-Score
Negatif	0,85	0,75	0,80
Positif	0,84	0,91	0,88

Berdasarkan laporan klasifikasi pada tabel 4.5 diatas didapatkan nilai *precision* 85% untuk sentimen negatif dan 84% untuk sentimen positif serta nilai recall 75% untuk sentimen negatif dan 91% untuk sentimen positif. Kemudian nilai f1-score: rata-rata dari presisi dan recall sebesar 80% untuk sentimen negatif dan 63% untuk sentimen positif. Berdasarkan laporan klasifikasi tersebut, kinerja model tersebut cukup baik, namun recall pada sentimen negatif masih cukup rendah karena banyak data negatif yang tidak terprediksi secara benar.

4.4.2. Hasil training dan testing LSTM menggunakan Word2Vec 100 dimensi kata

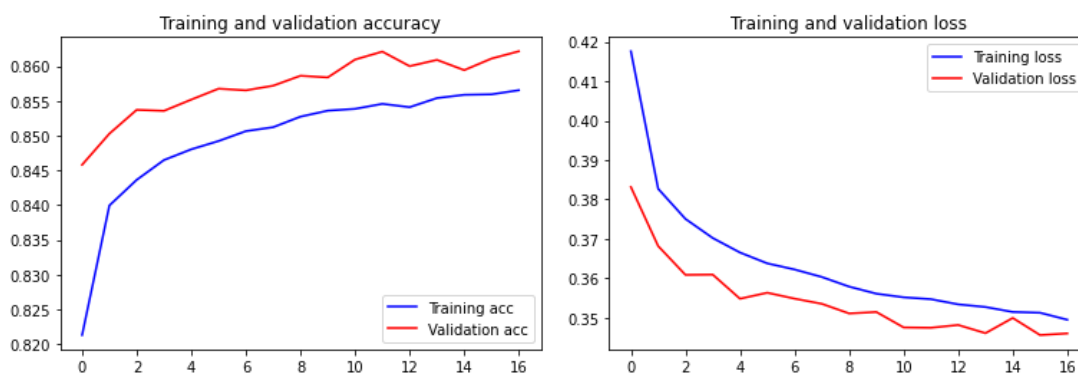
Proses training dengan teknik LSTM menggunakan Word2Vec dengan data ulasan Google Play sebanyak 210.044 data, dan menggunakan Word2Vec sebagai Embedding Layer dengan dimensi sebesar 100 kata. Hasil training data dapat dilihat pada Tabel 4.6 dibawah ini.

Tabel 4. 6 Hasil Training Data dengan Word2Vec 100 dimensi kata

Epoch	Akurasi	Error	Val Akurasi	Val Loss
1	82,13%	0,4176	84,58%	0,3832
2	83,99%	0,3827	85,03%	0,3682
3	84,36%	0,3751	85,37%	0,3609
4	84,65%	0,3702	85,36%	0,3609
5	84,80%	0,3665	85,52%	0,3548
6	84,92%	0,3638	85,67%	0,3564
7	85,06%	0,3622	85,65%	0,3549
8	85,12%	0,3603	85,72%	0,3535
9	85,27%	0,3579	85,86%	0,3511
10	85,36%	0,3561	85,84%	0,3515
11	85,38%	0,3552	86,09%	0,3475
12	85,46%	0,3547	86,21%	0,3475
13	85,41%	0,3534	86,00%	0,3482
14	85,54%	0,3527	86,09%	0,3461
15	85,59%	0,3515	85,94%	0,3500

Epoch	Akurasi	Error	Val Akurasi	Val Loss
16	85,59%	0,3513	86,11%	0,3456
17	85,65%	0,3495	86,21%	0,3460

Dari Tabel 4.6, dapat diketahui pada epoch ke-5 hasil training dengan error sebesar 0,3665 serta akurasi yang didapat sebesar 84,80%. Selanjutnya pada epoch ke-10 dengan error 0,3412 dan akurasi yang didapat sebesar 85,36%. Fungsi callback menghentikan proses pada epoch ke-17 dengan error yang didapat sebesar 0,3495 dengan akurasi sebesar 85,65 %. Proses training tersebut berlangsung selama 13 menit 31 detik. Hasil pengujian menggunakan Word2Vec ini lebih baik dari pengujian sebelumnya tanpa menggunakan Word2Vec. Perubahan error serta akurasi dari epoch ke-1 sampai epoch ke-17 dapat dilihat dalam grafik, grafik hasil error dan akurasi training jaringan LSTM dengan Word2Vec 100 dimensi dapat dilihat pada Gambar. 4.3



(a) Akurasi LSTM+WordVec 100 D (b) Error LSTM+WordVec 100 D

Gambar 4. 3 Hasil Training LSTM dengan Word2Vec 100 Dimensi Kata

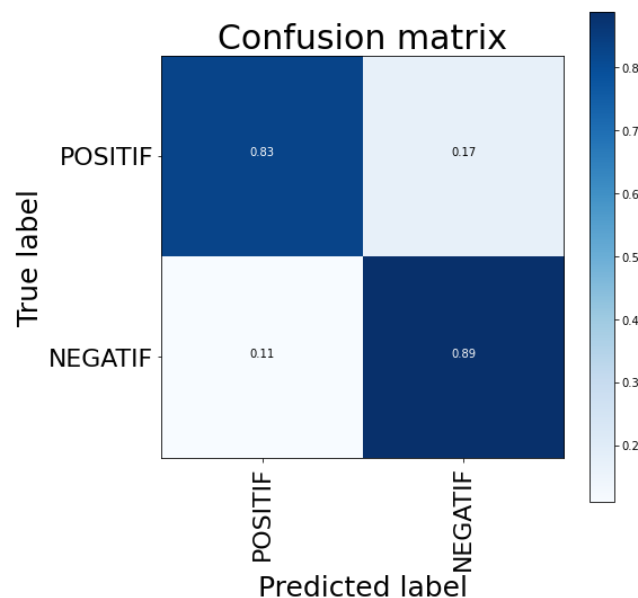
Dari gambar 4.3. diatas kita dapat melihat validasi terhadap akurasi dan validasi terhadap error (loss) cukup baik.

Dengan model tersebut dilakukan testing menggunakan data test yang telah dibagi dari dataset sebelumnya, sebesar 52.511 data. Dari proses testing yang dilakukan, model menghasilkan menghasilkan nilai akurasi sebesar 86,34 % dengan error sebesar 0,3362. Kinerja model ini dapat dilihat berdasarkan *confussion matrix* nya pada table 4.7 dibawah ini.

Tabel 4. 7 Tabel Confusion matrix hasil testing model LSTM dengan Word2Vec 100 Dimensi kata

Kelas	Positif	Negatif
Positif	27.711	3.461
Negatif	3.708	17.631

Dari *confusion matrix* di Tabel 4.7 diatas dapat dijelaskan bahwa model mengklasifikasikan secara benar sebesar 27.711 data sebagai positif dan 17.631 data sebagai negatif. Selain itu model salah dalam memprediksi 3.461 data ke dalam data negatif yang seharusnya positif (*false negative*), serta salah dalam memprediksi 3.708 data ke dalam data positif yang seharusnya negatif (*false positive*). Untuk plot persentase dari confusion matrix hasil testing model LSTM dengan Word2Vec 100 dimensi kata dapat dilihat pada Gambar 4.4 dibawah ini.



Gambar 4. 4 Confusion Matriks dari Hasil Testing LSTM dengan Word2Vec 100 Dimensi Kata

Berdasarkan *confusion matrix* hasil testing model diatas, kinerja dari model tersebut berdasarkan laporan klasifikasi yang dapat dilihat pada tabel 4.8 dibawah ini.

Tabel 4. 8 Kinerja klasifikasi model LSTM dengan Word2Vec 100 Dimensi kata

Sentimen	Precision	Recall	F1-Score
Negatif	0,84	0,83	0,83
Positif	0,88	0,89	0,89

Berdasarkan laporan klasifikasi pada tabel 4.8 diatas didapatkan nilai *precision* 84% untuk sentimen negatif dan 88% untuk sentimen positif serta nilai recall 89% untuk sentimen negatif dan 83% untuk sentimen positif. Kemudian nilai f1-score: rata-rata dari presisi dan recall sebesar 83% untuk sentimen negatif dan 89% untuk sentimen positif. Berdasrkan laporan klasifikasi tersebut, kinerja model tersebut cukup baik.

4.4.3. Hasil training dan testing LSTM menggunakan Word2Vec 200 dimensi kata

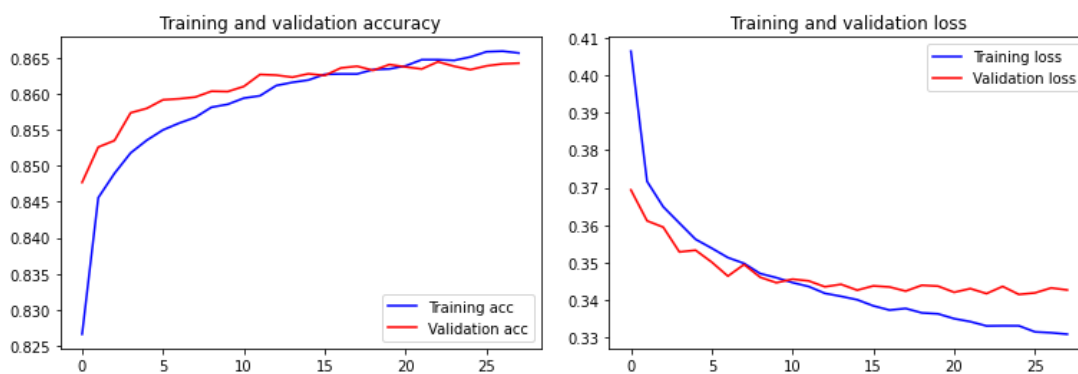
Proses training dengan teknik LSTM menggunakan Word2Vec sebagai Embedding Layer dengan dimensi sebesar 200 kata. Hasil training data dapat dilihat pada Tabel 4.9.

Tabel 4. 9 Training Data dengan Word2Vec 200 dimensi kata

Epoch	Akurasi	Error	Val Akurasi	Val Loss
1	82,66%	0,4064	84,77%	0,3694
2	84,56%	0,3717	85,26%	0,3612
3	84,89%	0,3649	85,35%	0,3595
4	85,18%	0,3606	85,74%	0,3529
5	85,36%	0,3562	85,80%	0,3533
6	85,50%	0,3539	85,92%	0,3501
7	85,59%	0,3514	85,93%	0,3464
8	85,67%	0,3498	85,96%	0,3495
9	85,81%	0,3471	86,04%	0,3461
10	85,85%	0,3460	86,03%	0,3446
11	85,94%	0,3447	86,10%	0,3456
12	85,97%	0,3436	86,27%	0,3451
13	86,12%	0,3418	86,26%	0,3436
14	86,16%	0,3410	86,23%	0,3442
15	86,19%	0,3401	86,28%	0,3427
16	86,27%	0,3385	86,26%	0,3438
17	86,28%	0,3374	86,36%	0,3435
18	86,28%	0,3378	86,38%	0,3424
19	86,34%	0,3366	86,33%	0,3440
20	86,35%	0,3364	86,41%	0,3437
21	86,39%	0,3351	86,37%	0,3421
22	86,48%	0,3343	86,35%	0,3431
23	86,48%	0,3331	86,45%	0,3417
24	86,47%	0,3332	86,38%	0,3437
25	86,51%	0,3332	86,34%	0,3415
26	86,59%	0,3315	86,39%	0,3419

Epoch	Akurasi	Error	Val Akurasi	Val Loss
27	86,59%	0,3313	86,42%	0,3433
28	86,57%	0,3309	86,43%	0,3427

Dari Tabel 4.9, dapat pada epoch ke-5 akurasi yang didapat sebesar 85,36% dengan error sebesar 0,3562, selanjutnya pada epoch ke-10 akurasi yang didapat sebesar 85,85% dengan error 0,3460. Pada epoch ke-11, validasi loss (error) beranjak lebih tinggi daripada nilai loss (error) nya dimana training akan menuju overfitting. Proses masih berlangsung sampai pada epoch maximum dimana training akan berhenti oleh fungsi Callback, sehingga training berhenti pada epoch ke-28 dengan error yang didapat sebesar 0,3309 dengan akurasi sebesar 86,57%. Proses training tersebut berlangsung selama 31 menit 7 detik. Perubahan error serta akurasi dari epoch ke-1 sampai epoch ke-28 dapat dilihat dalam grafik, grafik hasil error dan akurasi training jaringan LSTM dengan Word2Vec 200 dimensi dapat dilihat pada Gambar. 4.5.



(a) Akurasi LSTM+WordVec 200 D (b) Error LSTM+WordVec 200 D

Gambar 4. 5 Hasil Training LSTM dengan Word2Vec 200 Dimensi Kata

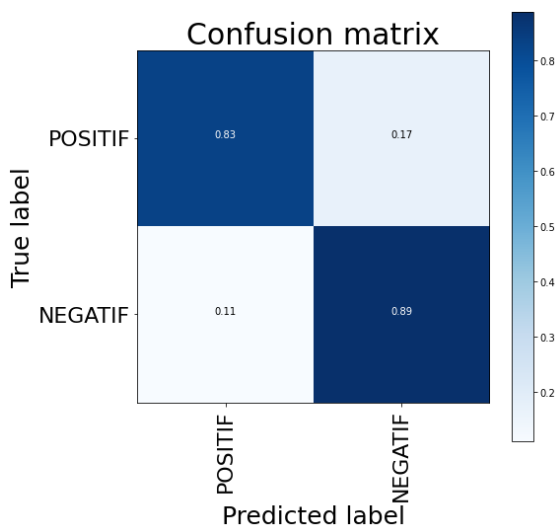
Pada gambar 4.5. diatas kita dapat melihat validasi terhadap akurasi dan validasi terhadap error (loss) sangat baik.

Dengan model tersebut dilakukan testing menggunakan data test yang telah dibagi dari dataset sebelumnya, sebesar 52.511 data. Dari proses testing yang dilakukan, model menghasilkan menghasilkan nilai akurasi sebesar 86,49 % dengan error sebesar 0,3376. Kinerja model ini dapat dilihat berdasarkan *confussion matrix* nya pada table 4.10 dibawah ini.

Tabel 4. 10 Tabel Confusion matrix hasil testing model LSTM dengan Word2Vec 200 Dimensi kata

Kelas	Positif	Negatif
Positif	27.680	3.492
Negatif	3.600	17.739

Dari *confusion matrix* di Tabel 4.10 diatas dapat dijelaskan bahwa model mengklasifikasikan secara benar sebesar 27.680 data sebagai positif dan 17.739 data sebagai negatif. Selain itu model salah dalam memprediksi 3.492 data ke dalam data negatif yang seharusnya positif (*false negative*), serta salah dalam memprediksi 3.600 data ke dalam data positif yang seharusnya negatif (*false positive*). Untuk plot persentase dari confusion matrix hasil testing model LSTM dengan Word2Vec 100 dimensi kata dapat dilihat pada Gambar 4.6 dibawah ini.



Gambar 4. 6 Confusion Matriks dari Hasil Testing LSTM dengan Word2Vec 200 Dimensi kata

Berdasarkan *confusion matrix* hasil testing model diatas, kinerja dari model tersebut berdasarkan laporan klasifikasi yang dapat dilihat pada tabel 4.11 dibawah ini.

Tabel 4. 11 Kinerja klasifikasi model LSTM+Word2Vec 200 Dimensi kata

Sentimen	Precision	Recall	F1-Score
Negatif	0,84	0,83	0,83
Positif	0,88	0,89	0,89

Berdasarkan laporan klasifikasi pada tabel 4.11 diatas didapatkan nilai *precision* 84% untuk sentimen negatif dan 88% untuk sentimen positif serta nilai recall 89% untuk sentimen negatif dan 83% untuk sentimen positif. Kemudian nilai f1-score: rata-rata dari presisi dan recall sebesar 83% untuk sentimen negatif dan 89% untuk sentimen positif. Berdasrkan laporan klasifikasi tersebut, kinerja model tersebut cukup baik.

4.4.4. Hasil training dan testing LSTM menggunakan Word2Vec 300 dimensi kata

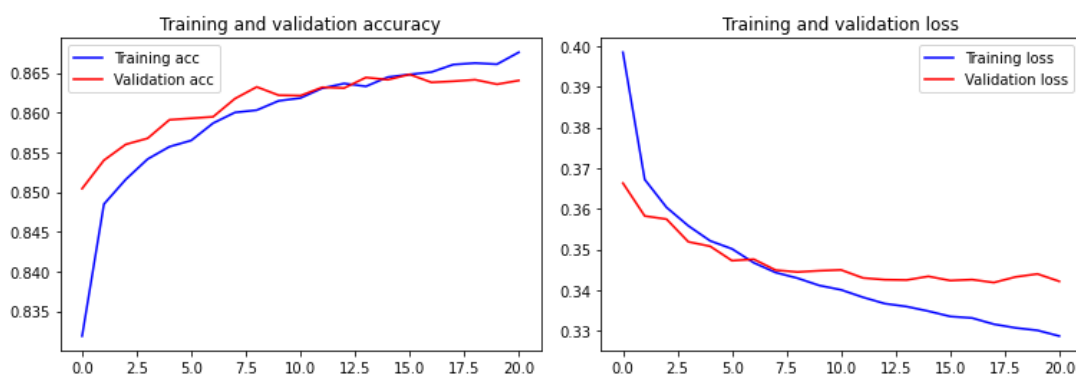
Proses training dengan teknik LSTM menggunakan Word2Vec sebagai Embedding Layer dengan dimensi sebesar 300 kata. Hasil training data dapat dilihat pada Tabel 4.12 dibawah ini.

Tabel 4. 12 Hasil Training Data dengan Word2Vec 300 dimensi

Epoch	Akurasi	Error	Val Akurasi	Val Loss
1	83,19%	0,3985	85,04%	0,3663
2	84,85%	0,3672	85,40%	0,3582
3	85,16%	0,3603	85,60%	0,3575
4	85,41%	0,3558	85,67%	0,3519
5	85,57%	0,3521	85,91%	0,3508
6	85,65%	0,3501	85,93%	0,3473
7	85,87%	0,3467	85,95%	0,3476
8	86,00%	0,3443	86,17%	0,3449
9	86,03%	0,3430	86,32%	0,3445
10	86,15%	0,3412	86,22%	0,3448
11	86,18%	0,3401	86,21%	0,3450
12	86,31%	0,3383	86,32%	0,3430
13	86,37%	0,3367	86,31%	0,3426
14	86,33%	0,3360	86,44%	0,3425
15	86,45%	0,3348	86,41%	0,3434
16	86,48%	0,3335	86,48%	0,3424
17	86,51%	0,3332	86,38%	0,3426
18	86,60%	0,3316	86,39%	0,3419
19	86,62%	0,3307	86,41%	0,3433
20	86,61%	0,3301	86,36%	0,3440
21	86,76%	0,3287	86,40%	0,3422

Dari Tabel 4.11 diatas, dapat diketahui bahwa semakin banyak perulangan atau epoch yang dilakukan, nilai error yang dihasilkan makin kecil, seperti pada epoch ke-5

akurasi yang didapat sebesar 85,57% dengan error sebesar 0,3521, selanjutnya pada epoch ke-10 akurasi yang didapat sebesar 86,15% dengan error 0,3412. Pada epoch ke-8, validasi loss (error) beranjak lebih tinggi daripada nilai loss (error) nya. Jaringan akan menuju overfitting, proses masih berlangsung sampai pada epoch maximum dimana training akan berhenti oleh fungsi Callback, sehingga training berhenti pada epoch ke-21 dengan error yang didapat sebesar 0,3287 dengan akurasi sebesar 86,76 %. Proses training tersebut berlangsung selama 33 menit 38 detik. Perubahan error serta akurasi dari epoch ke-1 sampai epoch ke-26 dapat dilihat dalam grafik, grafik hasil error dan akurasi training jaringan LSTM dengan Word2Vec 300 dimensi dapat dilihat pada Gambar. 4.7.



(b) Akurasi LSTM+WordVec 300 D

(b) Error LSTM+WordVec 300 D

Gambar 4. 7 Hasil Training LSTM dengan Word2Vec 300 Dimensi Kata

Dari gambar 4.7 diatas kita dapat melihat validasi terhadap akurasi dan validasi terhadap error (loss) sangat baik.

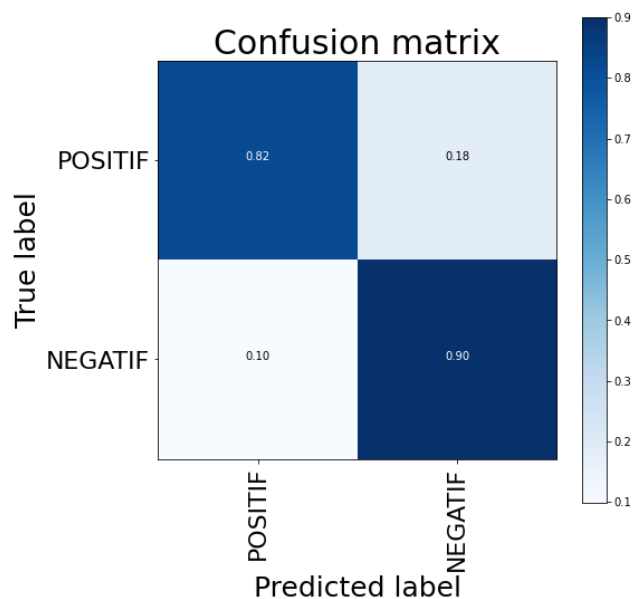
Dengan model tersebut dilakukan testing menggunakan data test yang telah dibagi dari dataset sebelumnya, sebesar 52.511 data. Dari proses testing yang dilakukan, model menghasilkan nilai akurasi sebesar 86,70 % dengan error sebesar 0,3362. Kinerja model ini dapat dilihat berdasarkan *confussion matrix* nya pada table 4.13 dibawah ini.

Tabel 4. 13 Tabel Confusion matrix hasil testing model LSTM dengan Word2Vec 300 Dimensi kata

Kelas	Positif	Negatif
Positif	27.706	3.466

Kelas	Positif	Negatif
Negatif	3.540	17.799

Dari *confusion matrix* di Tabel 4.13 diatas dapat dijelaskan bahwa model mengklasifikasikan secara benar sebesar 27.706 data sebagai positif dan 17.799 data sebagai negatif. Selain itu model salah dalam memprediksi 3.466 data ke dalam data negatif yang seharusnya positif (*false negative*), serta salah dalam memprediksi 3.540 data ke dalam data positif yang seharusnya negatif (*false positive*). Untuk plot persentase dari confusion matrix hasil testing model LSTM dengan Word2Vec 100 dimensi kata dapat dilihat pada Gambar 4.8 dibawah ini.



Gambar 4. 8 Confusion Matriks dari Hasil Testing LSTM dengan Word2Vec 300 Dimensi Kata

Berdasarkan *confusion matrix* hasil testing model diatas, kinerja dari model tersebut berdasarkan laporan klasifikasi yang dapat dilihat pada tabel 4.14 dibawah ini.

Tabel 4. 14 Kinerja klasifikasi model LSTM dengan Word2Vec 300 Dimensi Kata

Sentimen	Precision	Recall	F1-Score
Negatif	0,85	0,82	0,83
Positif	0,88	0,90	0,89

Berdasarkan laporan klasifikasi pada tabel 4.8 diatas didapatkan nilai *precision* 85% untuk sentimen negatif dan 88% untuk sentimen positif serta nilai recall 90% untuk

sentimen negatif dan 82% untuk sentimen positif. Kemudian nilai *f1-score*: rata-rata dari presisi dan recall sebesar 83% untuk sentimen negatif dan 89% untuk sentimen positif. Berdasarkan laporan klasifikasi tersebut, kinerja model tersebut sangat baik dibandingkan model lain yang telah diuji sebelumnya.

4.5. Pembahasan

Dari sub-bab sebelumnya yang membahas error dan akurasi pada saat training, dan akurasi dari hasil testing. Dari 4 (empat) model yang telah ditraining, selanjutnya akan dibandingkan kinerja masing-masing model berdasarkan tingkat akurasi, *error* dan waktu trainingnya nya sebagaimana yang terlihat pada tabel 4.15 berikut ini.

Tabel 4. 15 Perbandingan Kinerja *Training* 4 Model Percobaan

Model	Akurasi	Error	Waktu Training
LSTM tanpa Word2Vec	84,14%	0,3751	1 jam 25 Menit
LSTM + Word2Vec 100 Dimensi Kata	85,65%	0,3495	13 menit 13 detik
LSTM + Word2Vec 200 Dimensi Kata	86,57%	0,3309	31 menit 7 detik
LSTM + Word2Vec 300 Dimensi Kata	86,76%	0,3287	33 menit 38 detik

Berdasarkan tabel diatas, didapatkan akurasi model LSTM tanpa Word2Vec hanya menghasilkan akurasi sebesar 84,14%, *error* sebesar 0,3751 dengan waktu yang dibutuhkan selama proses training sebesar 1 jam 25 menit. Sementara itu dengan penambahan Word Embedding Word2Vec dapat memberikan peningkatan terhadap kinerja RNN-LSTM dalam analisis sentimen, dimana model-model yang menggunakan Word2Vec mendapatkan peningkatan akurasi, dan pengurangan *error*. Model LSTM dengan Word2Vec dimensi 300 kata mendapatkan akurasi tertinggi sebesar 86,76% dan *error* sebesar 0,3287 serta waktu training yang lebih cepat selama 33 menit 38 detik. Berdasarkan perbandingan tersebut, model LSTM dengan Word2Vec 300 Dimensi kata memiliki kinerja yang lebih baik dibandingkan model lainnya.

Sementara pada proses testing, dari 4 (empat) model yang telah diuji, selanjutnya akan dibandingkan kinerja masing-masing model berdasarkan tingkat akurasi, *precision*, *recall*, dan *f1-score* nya sebagaimana yang terlihat pada tabel 4.16 berikut ini.

Tabel 4. 16 Perbandingan Kinerja *Testing* 4 Model Percobaan

Model	Akurasi	<i>Precision</i>	<i>Recall</i>	<i>F1-Score</i>
LSTM tanpa Word2Vec	84,69%	84,29%	91,21%	87,61%
LSTM + Word2Vec 100 Dimensi Kata	86,35%	88,20%	88,90%	88,55%
LSTM + Word2Vec 200 Dimensi Kata	86,49%	88,49%	88,80%	88,64%
LSTM + Word2Vec 300 Dimensi Kata	86,66%	88,67%	88,88%	88,78%

Berdasarkan tabel diatas, didapatkan akurasi model LSTM tanpa Word2Vec hanya menghasilkan akurasi sebesar 84,69%, *Precision* sebesar 84,29% dan *F1-Score* 87,61%. Sementara itu dengan penambahan Word Embedding Word2Vec dapat memberikan peningkatan terhadap kinerja RNN-LSTM dalam analisis sentimen, dimana model-model yang menggunakan Word2Vec mendapatkan peningkatan akurasi, dimana model yang menggunakan Word2Vec dimensi 300 kata mendapatkan akurasi tertinggi sebesar 86,66%, *Precision* 88,67%, serta *F1-Score* sebesar 88,78%. Berdasarkan perbandingan nilai *precision* (presisi), dimana presisi adalah proporsi sentimen yang diprediksi dengan benar, didapatkan bahwa nilai presisi dari model yang menggunakan Word2Vec lebih besar daripada yang tidak menggunakan Word2Vec.

Kemudian hasil akurasi dan *F1-Score* yang didapatkan menyatakan bahwa makin tinggi nilai akurasi dan *F1-Score* nya maka kinerja model tersebut semakin baik. Berdasarkan hasil perbandingan ini didapatkan bahwa penambahan Word Embedding Word2Vec dapat memberikan peningkatan terhadap kinerja RNN-LSTM dalam analisis sentimen.

BAB 5

KESIMPULAN DAN SARAN

Penelitian ini dilakukan untuk mendapatkan hasil analisis sentimen dengan hasil kinerja yang baik atau memiliki akurasi yang tinggi. Pengklasifikasian disini menggunakan metode *Long Short- Term Memory*. Metodologi penelitian diawali dengan data yang digunakan, arsitektur umum penelitian, alur pengklasifikasian, dan analisis kinerja pengklasifikasian.

5.1. Kesimpulan

Dari hasil penelitian dan pembahasan maka kesimpulan yang didapat adalah sebagai berikut.

1. Hasil pelatihan menunjukkan bahwa jaringan LSTM yang menggunakan word embedding Word2Vec 300 dimensi kata mendapatkan nilai error yang rendah sebesar 0,3287 dengan akurasi sebesar 86,76 % pada epoch ke-21. Sedangkan hasil pengujian LSTM tanpa Word2Vec mendapatkan error terendah error yang sebesar 0,3751 dengan akurasi sebesar 84,14%.
2. Penambahan Word Embedding Word2Vec dapat memberikan peningkatan terhadap kinerja Deep Learning RNN-LSTM dalam hal kecepatan dan akurasi menuju global optima dimana model dengan Word2Vec dimensi 300 kata mendapatkan akurasi tertinggi sebesar 86,66%, *Precision* 88,67%, serta *F1-Score* sebesar 88,78%.

5.2. Saran

Berdasarkan hasil penelitian, maka saran untuk peneliti berikutnya adalah sebagai berikut.

1. Melakukan analisis lebih dalam terhadap metode penentuan kelas dan pengolahan dataset ulasan sehingga mendapatkan dataset yang bersih dan

memiliki sentimen yang tepat sehingga dapat meningkatkan akurasi yang lebih baik.

2. Penggunaan metode word embedding lainnya seperti *FastText*, *GloVe*s, *Doc2Vec* serta metode word embedding lainnya.

DAFTAR PUSTAKA

- Adam, K., Smagulova, K., & James, A. P. (2018). Memristive LSTM network hardware architecture for time-series predictive modeling problems. *2018 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*.
- Adriani, M., Asian, J., Nazief, B., Tahaghoghi, S.M.M., Williams, H.E. 2007. Stemming Indonesian: A Confix-Stripping Approach. *Transaction on Asian Language Information Processing*. Vol. 6, No. 4, Article 13. Association for Computing Machinery: New York.
- Araque, O., Corcuera-Platas, I., Sánchez-Rada, J. F., & Iglesias, C. A. (2017). Enhancing deep learning sentiment analysis with ensemble techniques in social applications. *Expert Systems with Applications*, 77, pp. 236–246.
- Bai, Y., Li, C., Sun, Z., & Chen, H. 2017. Deep neural network for manufacturing quality prediction. *Proceedings of prognostics and System Health Management Conference (PHM-Harbin)*, pp. 1-5.
- Berry, M.W. & Kogan, J. 2010. *Text Mining Application and theory*. WILEY: United Kingdom.
- Bin, G., Chunhui, H., Chong, Z., & Yanli, H. (2018). Classification Algorithm of Chinese Sentiment Orientation Based on Dictionary and LSTM. *Proceedings of the 2nd International Conference on Big Data Research – ICBDR*
- Ciftci, B., & Apaydin, M. S. (2018). A Deep Learning Approach to Sentiment Analysis in Turkish. *2018 International Conference on Artificial Intelligence and Data Processing (IDAP)*.
- Deng, X., Liu, Q., Deng, Y. & Mahadevan, S. 2016. An Improved Method to construct basic probability assignment based on the confusion matrix for classification problem. *Information Sciences* 340-341.
- Dragut, E., Fang, F., Sistla, P., Yu, S. & Meng, W. 2009. Stop Word and Related Problems in Web Interface Integration. (Online). <http://disi.unitn.it/~p2p/RelatedWork/Matching/vldb09a.pdf> (20 April 2019)
- Feldman, R & Sanger, J. 2007. *The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*. Cambridge University Press: New York.

- Haryanto, A.T. 2018. *130 Juta Orang Indonesia Tercatat Aktif di Medsos*.
<https://inet.detik.com/cyberlife/d-3912429/130-juta-orang-indonesia-tercatat-aktif-di-medsos> (1 Juni 2019)
- Hassan, A. & Mahmood, A., 2017. *Deep learning for sentence classification*. Proceedings of Long Island Systems, Applications and Technology Conference (LISAT), pp. 1-5.
- Jung, Y. 2017. Multiple Predicting K-Fold Cross-Validation for Model Selection. *Journal of Nonparametric Statistics* **30**(1) : 197-215.
- Li, W., Li, D., Yin, H., Zhang, L., Zhu, Z., dan Liu, P. 2019. Lexicon-Enhanced Attention Network Based on Text Representation for Sentiment Classification. *Applied Science* 2019, 9, 3717.
- Liu, Bing. 2012. *Sentiment Analysis And Opinion Mining*. Chicago: Morgan & Claypool Publisher. (Online). <https://www.cs.uic.edu/~liub/FBS/SentimentAnalysis-and-OpinionMining.pdf>. Diakses tanggal 10 April 2019.
- Miedema, F. 2018. Sentiment Analysis with Long Short-Term Memory Networks Research Paper Business Analytics. Vrije Universiteit Amsterdam. (Online). https://beta.vu.nl/nl/Images/werkstuk-miedema_tcm235-895557.pdf (10 April 2019)
- Mikolov, T., Chen, K., Corrado, G., & Dean, J. 2013. Efficient Estimation of Word Representations in Vector Space. (Online). <https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf> (10 April 2019).
- Mirza, A. H., & Cosan, S. (2018). Computer network intrusion detection using sequential LSTM Neural Networks autoencoders. 2018 26th Signal Processing and Communications Applications Conference (SIU).
- Naili, M., Chaibi, A. H., & Ben Ghezala, H. H. (2017). Comparative study of word embedding methods in topic segmentation. *Procedia Computer Science*, 112, 340–349. doi:10.1016/j.procs.2017.08.009
- Nikoskinen, T. 2015. From Neural Networks to Deep Neural Networks. (Online). http://sal.aalto.fi/publications/pdf-files/enik15_public.pdf. (02 Januari 2018).
- Olah, C. 2015. Understanding LSTM Networks. <https://colah.github.io/posts/2015-08-Understanding-LSTMs/> (10 April 2019)
- Patterson, J. and Gibson, A. 2017. *Deep Learning: A Practitioner's Approach*. O'Reilly Media, Inc.: Sebastopol.

- Saraswathi, D., & Sheela, L.M.I. 2016. Lung Image Segmentation Using K-Means Clustering Algorithm with Novel Distance Metric. *International Journal of Recent Trends in Engineering & Research* 2(12) : 236-245.
- Tala, Fadillah Z. 2003. A Study of Stemming Effects on Information Retrieval in Bahasa Indonesia. Institute for Logic, Language and Computation Universiteit van Amsterdam The Netherlands. (Online). [http://www.ilic.uva.nl/Research/Reports/ MoL-2003-02.text.pdf](http://www.ilic.uva.nl/Research/Reports/MoL-2003-02.text.pdf) (29 April 2019).
- Wong, T.-T., & Yang, N.-Y., 2017. Depedency Analysis of Accuracy Estimates in KFold Cross Validation. *IEEE Transactions of Knowledge and Data Engineering* **29**(11): 2417-2427.
- Yepes, A. J. (2017). Word embeddings and recurrent neural networks based on Long-Short Term Memory nodes in supervised biomedical word sense disambiguation. *Journal of Biomedical Informatics*, 73, pp. 137–147.
- Zulfa, I. & Winarko, E. Sentimen Analisis Tweet Berbahasa Indonesia dengan Deep Belief Network. *IJCCS (Indonesian Journal of Computing and Cybernetics Systems)*. IJCCS, Vol.11, No.2, July 2017, pp. 187-198.

Lampiran 1 Source Code Program dan Link Dataset

Semua *source code program* pada penelitian ini, dapat diunduh pada *link* dibawah ini :

Keterangan	URL
Kode Program LSTM-Word2Vec	https://github.com/boymanalu/DeepSentimentAnalysis/blob/master/m300W2C.ipynb
Hasil Scrapping Data Ulasan Aplikasi Google Play	https://github.com/boymanalu/DeepSentimentAnalysis/blob/master/all_dataset.xlsx
Data Ulasan Aplikasi Google Play yang sudah bersih	https://github.com/boymanalu/DeepSentimentAnalysis/blob/master/all_dataset_clean.xlsx
Model Word2Vec	https://github.com/boymanalu/DeepSentimentAnalysis/blob/master/Model_W2V_300.model