ITS
Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR – IS184853

# PENERAPAN WEIGHTED WORD EMBEDDING PADA PENGKLASIFIKASIAN TEKS BERBASIS RECURRENT NEURAL NETWORK UNTUK LAYANAN PENGADUAN PERUSAHAAN TRANSPORTASI

# *IMPLEMENTING WEIGHTED WORD EMBEDDING ON RECURRENT NEURAL NETWORK BASED TEXT CLASSIFICATION FOR TRANSPORTATION COMPANY CUSTOMER SERVICE*

MUHAMMAD DAVID RAHMAN
NRP. 05211640000066

Dosen Pembimbing
Prof. Ir. Arif Djunaidy, M.Sc., Ph.D.
Faisal Mahananto, S.Kom, M.Eng, Ph.D

DEPARTEMEN SISTEM INFORMASI
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

TUGAS AKHIR – IS184853

# PENERAPAN WEIGHTED WORD EMBEDDING PADA PENGKLASIFIKASIAN TEKS BERBASIS RECURRENT NEURAL NETWORK UNTUK LAYANAN PENGADUAN PERUSAHAAN TRANSPORTASI

**MUHAMMAD DAVID RAHMAN**
NRP. 05211640000066

**Dosen Pembimbing**
Prof. Ir. Arif Djunaidy, M.Sc., Ph.D.
Faisal Mahananto, S.Kom, M.Eng, Ph.D

**DEPARTEMEN SISTEM INFORMASI**
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

UNDERGRADUATE THESIS – IS184853

# *IMPLEMENTING WEIGHTED WORD EMBEDDING ON RECURRENT NEURAL NETWORK BASED TEXT CLASSIFICATION FOR TRANSPORTATION COMPANY CUSTOMER SERVICE*

**MUHAMMAD DAVID RAHMAN**
NRP. 05211640000066

**Supervisors**
Prof. Ir. Arif Djunaidy, M.Sc., Ph.D.
Faisal Mahananto, S.Kom, M.Eng, Ph.D

**INFORMATION SYSTEMS DEPARTMENT**
Faculty of Intelligent Electrical and Informatics Technology
Sepuluh Nopember Institute of Technology
Surabaya 2020

# LEMBAR PENGESAHAN

# PENERAPAN WEIGHTED WORD EMBEDDING PADA PENGKLASIFIKASIAN TEKS BERBASIS RECURRENT NEURAL NETWORK UNTUK LAYANAN PENGADUAN PERUSAHAAN TRANSPORTASI

**TUGAS AKHIR**

Disusun Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer (S.Kom)
pada
Departemen Sistem Informasi
Fakultas Teknologi Elektro dan Informatika Cerdas (ELECTICS)
Institut Teknologi Sepuluh Nopember

Oleh

**Muhammad David Rahman**
**05211640000066**

Surabaya, 14 Agustus 2020

**Kepala Departemen Sistem Informasi**

**Dr. Mudjahidin, ST., MT.**
**NIP. 19701102003121001**

**LEMBAR PERSETUJUAN**

# PENERAPAN WEIGHTED WORD EMBEDDING PADA PENGKLASIFIKASIAN TEKS BERBASIS RECURRENT NEURAL NETWORK UNTUK LAYANAN PENGADUAN PERUSAHAAN TRANSPORTASI

**TUGAS AKHIR**

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

Oleh:

**MUHAMMAD DAVID RAHMAN**
05211640000066

Disetujui Tim Penguji : Tanggal Ujian : 13 Juli 2020
Periode Wisuda : September 2020

**Prof. Ir. Arif Djunaidy, M.Sc., Ph.D** **(Pembimbing I)**

**Faisal Mahananto, S.Kom, M.Eng, Ph.D** **(Pembimbing II)**

**Ahmad Mukhlason, S.Kom, M.Sc, Ph.D** **(Penguji I)**

**Retno Aulia Vinarti, S.Kom, M.Kom, Ph.D** **(Penguji II)**

# PENERAPAN WEIGHTED WORD EMBEDDING PADA PENGKLASIFIKASIAN TEKS BERBASIS RECURRENT NEURAL NETWORK UNTUK LAYANAN PENGADUAN PERUSAHAAN TRANSPORTASI


| | |
|---|---|
| **Nama Mahasiswa** | : Muhamad David Rahman |
| **NRP** | : 05211640000066 |
| **Departemen** | : Sistem Informasi FTEIC-ITS |
| **Pembimbing 1** | : Prof. Ir. Arif Djunaidy, M.Sc., Ph.D |
| **Pembimbing 2** | : Faisal Mahananto, S.Kom, M.Eng, Ph.D |


## ABSTRAK


*Twitter menjadi salah satu media sosial yang paling sering dan paling populer digunakan oleh perusahaan sebagai penyedia layanan pelanggan perusahaan. Adanya ribuan cuitan yang dapat masuk dalam setiap hari, tentu akan merepotkan operator layanan untuk mengkategorikan jenis berbagai cuitan tersebut, lebih-lebih jika proses pemilahan kategori cuitan harus dilakukan secara manual.*

*Dalam Tugas Akhir ini, kategoriasi cuitan secara otomatis dibangun dan diimplementasikan menggunakan model klasifikasi berbasis recurrent neural network (RNN) yang dikombinasikan dengan model weighted word embedding (WWE). RNN merupakan salah satu jenis jaringan syaraf tiruan yang populer dan banyak digunakan dalam persoalan klasifikasi, sedangkan WWE merupakan metode yang memungkinkan untuk menghubungkan kata-kata yang serupa dengan mengukur jarak semantik antara vektor yang disematkan pada kata tersebut dan memberikan bobot yang berbeda pada setiap kata pada suatu kelas tertentu.*

*Implementasi model penggabungan RNN dan WWE diuji coba menggunakan data pengaduan di perusahaan transportasi untuk data cuitan pada tahun 2015-2016. Hasil uji coba menunjukkan bahwa implementasi WWE baik yang*

*menggunakan model FastText (Weighted FastText) maupun model Word2Vec (Weighted Word2Vec) memberikan hasil yang lebih baik dibandingkan dengan hasil kinerja yang menggabungkan RNN dan model word embedding biasa. Dengan menggunakan metode evaluasi berbasis 10-fold cross validation, model gabungan RNN-Weighted FastText dan RNN-Weighted Word2Vec berturut-turut memberikan hasil akurasi sebesar 88,2% dan 87,5%. Di lain pihak, dengan menggunakan metode evaluasi yang sama, model gabungan RNN-FastText dan RNN-Word2Vec memberikan hasil akurasi yang sama sebesar 83,4%.*

***Kata kunci: FastText, Klasifikasi teks, Layanan Pengaduan Transportasi, Recurrent Neural Network, Twitter, Weighted Word Embedding, Word2Vec***

***IMPLEMENTING WEIGHTED WORD EMBEDDING ON RECURRENT NEURAL NETWORK BASED TEXT CLASSIFICATION FOR TRANSPORTATION COMPANY CUSTOMER SERVICE***


| | |
|---|---|
| **Student Name** | : Muhammad David Rahman |
| **NRP** | : 05211640000066 |
| **Departement** | : Sistem Informasi FTEIC-ITS |
| **Supervisor 1** | : Prof. Ir. Arif Djunaidy, M.Sc., Ph.D |
| **Supervisor 2** | : Faisal Mahananto, S.Kom, M.Eng, Ph.D |


## ABSTRACT


*Twitter is one of the most frequently used platforms by companies as their customer service platform. There are thousands of incoming tweets posted in a day, it will certainly be inconvenient for service operators to categorize the tweets. If the operators still using manual process will certainly require a long time.*

*In this final project, tweet categorization automatically built and implemented using a recurrent neural network (RNN) model which is combined with the weighted word embedding (WWE) model. RNN is one of the most popular and widely used neural network in solving the classification problems, whereas WWE model enables similar words to be associated one each other by measuring the semantic distance between vectors embedded in the word and assign different weight on each word in its particular class.*

*The Implementation of the combined RNN and WWE model was tested using customer complaint data in public transportation company from its twitter account ranging from year 2015 to 2016. The results showed that the implementation using both FastText (weighted FastText) and Word2Vec (weighted Word2Vec) models are better than using the common word embedding model without the weight. By using the evaluation method based on 10-fold cross validation, the combined model*

*of RNN-Weighted FastText and RNN-Weighted Word2Vec gave an accuracy of 88.2% and 87.5%, respectively. On the other hand, using the same evaluation method, both the combined model RNN-FastText and RNN-Word2Vec gave the same accuracy of 83.4%.*

***Kata kunci: FastText, Recurrent Neural Network, Text Classification, Transportation Customer Service, Twitter, Weighted Word Embedding, Word2Vec***

# SURAT PERNYATAAN BEBAS PLAGIARISME

Saya yang bertandatangan di bawah ini:

| | |
|---|---|
| Nama | : Muhammad David Rahman |
| NRP | : 05211640000066 |
| Tempat/ Tanggal Lahir | : Surabaya, 25 Juli 1998 |
| Fakultas/ Departemen | : FTEIC/ Sistem Informasi |
| Nomor Telp/ HP/e-mail | : 085854633238/ muhdavidrahman@gmail.com |

Dengan ini menyatakan dengan sesungguhnya bahwa penelitian/makalah/tugas akhir saya yang berjudul:

PENERAPAN WEIGHTED WORD EMBEDDING PADA PENGKLASIFIKASIAN TEKS BERBASIS RECURRENT NEURAL NETWORK UNTUK LAYANAN PENGADUAN PERUSAHAAN TRANSPORTASI

**Bebas Dari Plagiarisme Dan Bukan Hasil Karya Orang Lain.**

Apabila dikemudian hari ditemukan seluruh atau sebagian penelitian/makalah/tugas akhir tersebut terdapat indikasi plagiarisme, maka saya bersedia menerima sanksi sesuai peraturan dan ketentuan yang berlaku.

Demikian surat pernyataan ini saya buat dengan sesungguhnya dan untuk dipergunakan sebagaimana mestinya.

Surabaya, 13 Juli 2020


**Muhammad David Rahman**

NRP. 05211640000066

*“Halaman ini sengaja dikosongkan”*

# KATA PENGANTAR

Puji syukur Alhamdulillah penulis panjatkan kehadirat Allah SWT yang telah melimpahkan rahmat dan hidayah-Nya, penulis dapat menyelesaikan Tugas Akhir dengan judul:

**"PENERAPAN WEIGHTED WORD EMBEDDING PADA PENGKLASIFIKASIAN TEKS BERBASIS RECURRENT NEURAL NETWORK UNTUK LAYANAN PENGADUAN PERUSAHAAN TRANSPORTASI"**

Penyusunan Tugas Akhir ini dilakukan sebagai salah satu syarat kelulusan di Departemen Sistem Informasi, Institut Teknologi Sepuluh Nopember (ITS), Surabaya. Dalam penyusunan Tugas Akhir ini, penulis mendapatkan banyak sekali doa, bimbingan, dan dukungan dari berbagai pihak. Oleh karena itu, pada kesempatan ini penulis ingin menyampaikan terima kasih sebesar-besarnya kepada:

- Kedua orangtua dan kedua kakak yang selalu mengingatkan, memberi dukungan finansial dan moral, serta mendoakan penulis dari awal masuk perkuliahan hingga bisa menyelesaikan Tugas Akhir ini
- Bapak Prof. Ir. Arif Djunaidy, M.Sc., Ph.D dan Bapak Faisal Mahananto, S.Kom, M.Eng, Ph.D yang telah memberikan saya bimbingan, arahan, dan motivasi dari awal pengerjaan proposal hingga Tugas Akhir ini selesai
- Bapak Ahmad Mukhlason, S.Kom, M.Sc, Ph.D dan Ibu Retno Aulia Vinarti, S.Kom, M.Kom, Ph.D yang telah menguji dan memberikan saran untuk memperbaiki Tugas Akhir
- Bapak Radityo Prasetianto W., S.Kom, M.Kom selaku dosen wali yang senantiasa mendampingi sejak penulis menjadi Mahasiswa Baru hingga penulis bisa menyelesaikan Tugas Akhir
- Seluruh dosen yang sudah mentransferkan ilmunya selama perkuliahan penulis

- Varian selaku teman penulis yang sudah banyak membantu penulis dan juga sebagai teman diskusi saat sedang kesulitan
- Gusti dan Yosh, teman yang selalu menemani saat di kosan dan selalu memberikan dukungan moral dan meredakan penat dalam mengerjakan Tugas Akhir
- Teman WW 41, terdiri dari Abdul, Beyo, Cica, Daffa, Gilang, Hazim, Icad, Upi, Syifa, dan yang lain yang selalu menemani malam-malam penulis selama pandemi ini dengan melakukan *group call* menggunakan Zoom dan Google Meet
- Pihak lain yang telah membantu dan mendukung penyelesaian Tugas Akhir yang tidak bisa disebutkan satu persatu

Penulis sadar bahwa pengerjaan ini masih jauh dari kata sempurna dan banyak kekurangan. Namun, penulis berharap semoga penyusunan Tugas Akhir ini bisa memberi manfaat untuk menambah pengetahuan para pembaca serta dapat menjadi acuan bagi instansi terkait untuk berbenah. Penulis juga akan terbuka terhadap saran dan masukan dari semua pihak.

Surabaya, 02 Juli 2020
Penulis

# DAFTAR ISI

# DAFTAR GAMBAR

*“Halaman ini sengaja dikosongkan”*

# DAFTAR TABEL

# DAFTAR KODE PROGRAM

*“Halaman ini sengaja dikosongkan”*

# BAB 1
# PENDAHULUAN

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

## 1.1. Latar Belakang

Pada era digital ini, media sosial merupakan suatu platform penting yang ikut mendorong kemajuan teknologi khususnya dalam kehidupan sosial manusia. Media sosial saat ini tidak hanya digunakan sebagai ruang berekspresi dan beropini seperti sedia kala. Namun, saat ini sudah bertransformasi menjadi suatu platform yang digunakan untuk segala jenis interaksi sosial di masyarakat. Salah satu contohnya yaitu penggunaan media sosial sebagai tempat menjual produk berupa barang dan juga jasa. Selain itu juga menggunakan media sosial sebagai platform untuk mengiklankan produk yang dijual. Contoh lainnya yaitu penggunaan media sosial sebagai *customer service* (layanan pelanggan) dan support.

Saat ini ada banyak *platform* media sosial yang digunakan, beberapa contohnya yaitu Facebook, Twitter, Youtube, Instagram, dan lain-lain. Berdasarkan infografis dari GlobalWebIndex, pengguna aktif media sosial di Indonesia mencapai 130 juta orang. Twitter sendiri merupakan salah satu platform dengan pengguna yang lumayan besar jumlahnya. Menurut data yang diberikan Twitter, ada total 330 juta pengguna Twitter yang aktif setiap harinya, dengan total 126 juta cuitan yang masuk. Twitter menjadi salah satu platform yang paling sering digunakan oleh perusahaan sebagai penyedia layanan pelanggan mereka. Berdasarkan survei yang dilakukan Aberdeen Group, 41% dari 170 perusahaan yang mereka survei sudah menggunakan media sosial sebagai media untuk layanan

pelanggan. Dari semua perusahaan yang memanfaatkan media sosial, 51% perusahaan menggunakan Twitter sebagai *platform* untuk layanan pelanggan mereka.

Penggunaan media sosial sebagai sarana pengaduan pelanggan merupakan tantangan tersendiri untuk suatu perusahaan. Banyak pelanggan tentunya mengharapkan penganan aduan secara *real-time* seperti pada saat pelanggan melakukan aduan dengan menghubungi *call center*. Namun dengan media sosial hal ini akan cukup sulit karena banyaknya aduan yang masuk dalam satu waktu tanpa ada filterisasi. Waktu respons selalu menjadi nilai penting bagi pelanggan. Oleh karena itu pelanggan mengharapkan pengaduan menggunakan media sosial dapat ditanggapi secara *real-time,* setidaknya ditanggapi pada hari yag sama. Northridge Group melaporkan bahwa 42 persen konsumen mengharapkan tanggapan atas permintaan layanan pelanggan mereka dalam satu jam. Dari kelompok ini, 17 persen mengharapkan tanggapan dalam hitungan menit. Waktu respons memang penting namun ada hal lain yang tidak kalah peting yaitu menyelesaikan aduan yang dilaporkan [1].

Kategorisasi pada aduan yang masuk melalui Twitter dapat mengurangi masalah-masalah diatas. Filterisasi aduan yang masuk secara manual tentu akan memakan waktu yang sangat lama, terutama jika aduan masuk secara bersamaan dengan jumlah banyak. Namun dengan komputasi kategorisasi teks, hal ini dapat ditangani dengan cepat. Aduan yang masuk akan terpilah-pilah. Dan ini akan memudahkan dalam penyelesaian aduan tersebut. Sehingga tidak hanya memberikan efek yang baik pada waktu respons aduan, namun juga perusahaan dapat menyelesaikan masalah yang diadukan oleh pelanggan.

Kategorisasi cuitan dapat dilakukan dengan berbagai cara. Saat ini sudah banyak tools kategorisasi Twitter yang dapat digunakan untuk mengelompokkan cuitan pada Twitter. Cara lain yang dapat digunakan adalah dengan melakukan klasifikasi teks. Klasifikasi teks ini dapat digunakan menggunakan ilmu text mining (penggalian teks). Klasifikasi teks merupakan sebuah proses pembelajaran mesin yang mampu

mengkategorikan dokumen teks ke dalam kelas-kelas yang sudah ditentukan sebelumnya secara tepat. Sehingga cuitan yang masuk akan dapat langsung dikategorikan ke dalam kelas yang sudah ditentukan secara otomatis. Dalam melakukan klasifikasi teks dapat dilakukan dengan berbagai metode, yaitu *Random Forest, Naive Bayes, Support Vector Machine* (SVM), dan *Neural Network*. *Neural Network* atau metode jaringan syaraf merupakan metode pembelajaran mesin yang berdasar pada pembelajaran yang menyerupai sistem otak manusia dalam menangkap pola. Metode ini sedang digandrungi dalam melakukan kalsifikasi teks dibandingkan metode konvensional lainnya. Ide penggunaan metode ini adalah untuk menyelesaikan permasalahan *sparsity* data dan representasi semantik dari data [2].

Model *neural network* sudah banyak digunakan untuk melakukan representasi kata, model representasi ini biasanya disebut *word embedding*. *Word embedding* merupakan suatu metode penyisipan kata, mengubah kata menjadi suatu vektor yang kontinyu dengan panjang yang sudah ditetapkan, sehingga tidak akan dibatasi dengan kosakata yang lebih banyak. Model ini juga kaya akan informasi semantik, sehingga memungkinkan kita untuk menghubungkan katakata yang serupa dengan mengukur jarak semantik antara vektor yang disematkan pada kata tersebut. Namun pada pengaplikasiannya, *word embedding* biasa tidak menaruh bobot pada tiap kata pada suatu kelas. Sedangkan, pada umumnya tiap kata akan memiliki makna dan kepentingan yang berbeda pada penggunaannya. Oleh karena itu, pada tugas akhir ini diusulkan menggunakan metode *weighted word embedding*. Dimana tiap kata akan memiliki bobot yang berbeda tergantung pada kelas mana kata tersebut dikategorikan.

*Neural network* memiliki banyak metode yang dapat digunakan untuk melakukan klasifikasi teks. Salah satu yang populer dan seing digunakan adalah metode *Recurrrent Neural Network*. Metode ini memiliki karakteristik tersendiri yaitu memiliki "memori" yang dapat menyimpan prediksi kata sebelumnya sebagai masukkan prediksi kata selanjutnya. Hal ini dapat

meningkatkan akurasi dalam melakukan prediksi berkelanjutan. Umumnya metode jaringan syaraf memiliki input dan output yang independen, tidak bergantung satu sama lain, namun beberapa kasus butuh kesinambungan antara satu kalimat dengan kalimat lain yang akan diprediksi. Dengan menggunakan *hidden layer* RNN masalah tersebut dapat diselesaikan. RNN mampu menangkap konteks secara luas dari data karena karakteristiknya yang memiliki dependensi antar kata jangka panjang [3].

Pada tugas akhir ini akan membuat sebuah model klasifikasi teks dengan menggunakan metode *recurrent neural network* yang akan dipadukan dengan *weighted word embedding* untuk menghasilkan model yang akurat. Model klasifikasi nantinya akan di aplikasikan pada data Twitter layanan pengaduan perusahaan trasportasi pada kasus ini yaitu PT. Kereta Api Indonesia (Persero), yaitu pada akun @KAI121.

## 1.2. Rumusan Masalah

Berdasarkan uraian latar belakang di atas, maka rumusan masalah yang menjadi fokus untuk diselesaikan dalam Tugas Akhir ini adalah sebagai berikut.

1. Bagaimana hasil pengukuran kinerja model pengklasifikasian menggunakan RNN?
2. Bagaimana seleksi fitur kata menggunakan metode Weighted Word Embedding dapat mempengaruhi kinerja hasil klasifikasi?
3. Bagaimana perbedaan hasil pengukuran kinerja model pengklasifikasian RNN antara yang menggunakan dan yang tidak menggunakan metode Weighted Word Embedding?

## 1.3. Batasan Masalah

Berikut ini adalah batasan permasalahan yang menjadi ruang lingkup pengerjaan Tugas Akhir ini.

1. Tugas akhir ini berfokus pada pembuatan model pengkategorian pengaduan layanan perusahaan transportasi.
2. Data yang digunakan pada tugas akhir ini dokumen Twitter PT Kereta Api Indonesia (Persero) yang masuk ke dalam akun @KAI121 dari bulan Januari 2015 hingga Maret 2016 dan hanya berupa teks bahasa Indonesia.
3. Metode klasifikasi yang digunakan adalah metode Recurrent Neural Network (RNN) dan metode Weighted Word Embedding.
4. Terdapat 3 kategori yang digunakan, yaitu kategori Informasi, Saran, dan Keluhan. Kategori tersebut tidak bersifat hirarki atau bersusun.

### 1.4. Tujuan Tugas Akhir

Tujuan dari tugas akhir ini adalah untuk membagun model pengkategorian pengaduan layanan Twitter pada perusahaan transportasi. Tujuan lain dari perspektif hubungan pelanggan yaitu memberikan waktu respons yang lebih cepat terhadap aduan dan menyelesaikan aduan pelanggan.

### 1.5. Manfaat Tugas Akhir

Manfaat yang akan diberikan dari dibuatnya tugas akhir ini adalah untuk membuat model yang akurat untuk melakukan klasifikasi teks pada dokumen layaan pengaduan melaui Twitter. Diharapkan nantinya model ini dapat menyelesaikan permasalahan pengkategorian cuitan yang masuk ke layanan pengaduan Twitter. Tugas akhir ini juga harapannya akan menjadi referensi untuk melakukan penelitian lanjutan agar ditemukannya metode akurat lainnya untuk melakukan klasifikasi teks dengan berbagai jenis dokumen data. Harapannya penggunaan weighted word embedding pun dapat dijadikan salah satu solusi untuk meningkatkan akurasi dari model.

## 1.6. Relevansi Tugas Akhir

Pada tugas akhir ini melakukan klasifikasi teks terhadap cuitan yang dilakukan di Twitter. Tugas akhir ini bertujuan untuk membuat metode yang akurat dalam mengklasifikasi dan mengkategorikan cuitan yang masuk ke dalam akun layanan pengaduan sebuah perusahaan. Metode yang digunakan yaitu recurrent neural network dengan ditambahkan bobot pada saat melakukan word embedding Topik ini sesuai dengan fokus bidang ilmu laboratorium di jurusan Sistem Informasi yaitu laboratorium Rekayasa Data dan Intelegensi Bisnis dan beberapa mata kuliah seperti mata kuliah tentang penggalian data dan analitika bisnis. Penelitian ini berkorelasi dengan salah satu fokus di laboratorium RDIB yaitu Intelegensi Bisnis. Dimana dalam penelitian ini mengaplikasikan ilmu penggalian teks untuk menyelesaikan permasalahan yang dibawa.

# BAB 2
# TINJAUAN PUSTAKA

Pada bab ini berisi penelitian sebelumnya yang dijadikan acuan dalam pengerjaan tugas akhir dan juga berisi dasar teori untuk menunjang penelitian pada tugas akhir.

## 2.1. Penelitian Sebelumnya

Pada sub-bab ini berisikan daftar penelitian sebelumnya yang dijadikan acuan dalam pengerjaan tugas. Terdapat 4 penelitian yang dijadikan acuan penulis. Penelitian ini dijadikan acuan penulis dalam memilih metode yang digunakan dalam pengerjaan tugas akhir.

### 2.1.1. Penelitian 1

Penelitian dengan judul "Improving text classification with weighted word embeddings via a multi-channel TextCNN model". Dibuat oleh Bao Guo, Chunxia Zhang, Junmin Liu, Xiaoyi Ma pada tahun 2019.

Pada penelitian ini bertujuan untuk meningkatkan performa model klasifikasi teks pada metode *TextCNN* dengan menambahkan bobot (*weight*) pada saat proses word embedding. Untuk meningkatkan akurasi klasifikasi, pendekatan menggunakan pembobotan telah terbukti cukup efektif. Tetapi selama ini, hampir semua metode hanya menetapkan satu bobot untuk setiap istilah (kata). Padahal kenyataannya tiap kata pada kalimat memiliki makna dan maksud yang berbeda dalam dokumen dengan label kelas yang berbeda. Penenlitian ini skema pembobotan baru untuk dikombinasikan dengan word embedding untuk meningkatkan kinerja klasifikasi CNN. Dalam metode ini, beberapa bobot diberikan untuk setiap istilah dan bobot ini diterapkan pada word embedding secara terpisah. Selanjutnya, fitur yang diubah dimasukkan ke dalam model CNN *multichannel* untuk memprediksi label kelas. Dengan membandingkan metode baru

dengan beberapa metode baseline lainnya dengan lima set data benchmark.

Hasilnya menunjukkan bahwa akurasi klasifikasi metode yang diusulkan melebihi metode lainnya dengan margin yang luar biasa. Selain itu, bobot yang diberikan oleh skema pembobotan yang berbeda juga dianalisis untuk mendapatkan lebih banyak perbandingan tentang mekanisme kerja meetode tersebut. Hasilnya cukup impresif karena menunjukkan peningkatan dibandingkan dengan hanya menggunakan CNN-static biasa tanpa pembobotan. Penelitian ini memiliki relevansi dengan tugas akhir yang akan dikerjakan, karena penelitian ini menggunakan metode *weighted word embedding*, yaitu menambahkan bobot yang berbeda pada tiap kata. Metode ini juga akan dipakai pada tugas akhir ini dengan tujuan yang sama yaitu meningkatkan akurasi model

### 2.1.2. Penelitian 2

Penelitian dengan judul "Recurrent neural networks with segment attention and entity description for relation extraction from clinical texts". Dibuat oleh Zhi Li, Jinshan Yang, Xu Gou, dan Xiaorong Qi pada tahun 2019.

Pada penelitian ini bertujuan untuk mengatasi kekurangan yang terdapat pada model yang sebelumnya yaitu mengalami degradasi performa dan hasil kinerja yang kurang baik saat menemukan kalimat yang panjang dan juga saat bertemu dua atau lebih entitas pada kalimat yang sama. Pada penelitian ini menggunakan arsitektur *neural network* berdasarkan *Bidirectional long short-term memory*. Arsitektur ini ditambahkan dengan 3 layer tambahan untuk meningkatkan performa dan mengatasi permasalahan diatas. Pertama, peneliti menggunakan mekanisme *concat-attention* untuk menangkap konteks kata yang paling penting untuk ekstraksi relasi dalam kalimat. Lalu, memanfaatkan mekanisme *segment attention* untuk meningkatkan kinerja model yang memproses kalimat

panjang. Dan terakhir, *tensor-based entity description* digunakan untuk mengatasi penurunan kinerja model ketika terdapat dua atau lebih entitas dalam kalimat.

Penelitian ini membandingkan hasil dari model sebelumnya dengan hasil penelitian yang dilakukan. Penelitian ini berhasil meningkatkan nilai f-measure sebesar 3% dari sebelumnya. Dari hasil percobaan menunjukkan bahwa model *BLSTM-SegAtt-Tensor* dapat mempelajari fitur-fitur yang ada dengan cukup baik dan mengungguli sebagian besar metode yang ada untuk mengekstraksi hubungan di antara entitas kata yang berhubungan dengan teks klinis.

Penelitian menggunakan model *BLSTM-SegAtt-Tensor* untuk meningkatkan performa pada saat menemukan kalimat panjang dan entitas yang sama dalam kalimat. Hal ini berkaitan dengan tugas akhir yang akan mengklasifikasi cuitan dari media sosial twitter. Pada umumnya masyarakat melakukan cuitan tanpa menggunakan kaidah kalimat yang tepat sehingga membuat kalimat menjadi panjang

### 2.1.3. Penelitian 3

Penelitian dengan judul "Comparative effectiveness of convolutional neural network (CNN) and recurrent neural network (RNN) architectures for radiology text report classification". Dibuat oleh Imon Banerjee, Yuan Ling, Matthew C. Chen, Sadid A. Hasan, Curtis P. Langlotz, Nathaniel Moradzadeh, Brian Chapman, Timothy Amrhein, David Mong, Daniel L. Rubina, Oladimeji Farri, dan Matthew P. Lungren pada tahun 2019.

Pada *paper* ini dilakukan penelitian komparasi performa antara model CNN Word - Glove dan *Domain phrase attention-based hierarchical recurrent neural network* (DPA-HNN). Model ini digunakan untuk mengklasifikasikan informasi tentang *Pulmonary Emboli* (PE) dari 7370 lebih laporan radiologi

bertipe *free-text*. Klasifikasi dikategorikan menjadi 2, yaitu *PEpositive/negative* dan *PEacute/chronic*. Selain mengkomparasi 2 model deep learning, pada penelitian ini juga melakukan komparasi terhadap model yang sudah dibuat sebelumnya yaitu PEFinder dan juga dengan *metode traditional machine learning* yang lain seperti SVM dan Adaboost. Model DPA-HNN yang diusulkan mengkodekan frase pada domain ke dalam mekanisme model dan mewakili laporan radiologi melalui struktur RNN yang terdiri dari representasi tingkat kata, kalimat, dan tingkat dokumen.

Pada hasil komparasi menunjukkan model *deep learning* (DPA-HNN dan RNN) memiliki performa kinerja yang lebih baik di antara model lainnya. Dari 4 data pusat kesehatan, RNN model memiliki performa yang baik dan dapat memprediksi kelas dengan tepat dibandingkan dengan model RNN. Penelitian yang dilakukan memiliki relevansi yang sama dengan pengerjaan tugas akhir ini yaitu klasifikasi teks dengan salah satu metode yang sama yaitu *recurrent nerural network*.

### 2.1.4 Penelitian 4

Penelitian dengan judul "The Impact of Features Extraction on the Sentiment Analysis". Penelitian dilakukan oleh Ravinder Ahuja pada tahun 2019.

Penelitian ini melakukan analisis sentimen pada data teks Twitter. Penelitian ini membandingkan hasil akurasi dalam melakukan ekstraksi fitur dengan dua cara berbeda, yaitu TF-IDF dan N-Gram. Dari hasil penelitian menunjukkan ekstraksi fitur menggunakan TF-IDF memiliki hasil performa yang lebih bagus yaitu sekitar 3-4%. Hal ini dikarenakan pada TF-IDF mempertimbangkan kepentingan setiap kata pada suatu dokumen.

Penelitian ini relevan dengan tugas akhir yang akan dikerjakan karena pada tugas akhir ini akan melalui proses TF-IDF pada saat melakukan *Weighted Word Embedding*.

## 2.2. Dasar Teori

Pada sub-bab ini menjelaskan teori-teori yang menjadi acuan dalam pengerjaan tugas akhir. Selain mengacu pada penelitian yang sudah dilakukan, pengerjaan tugas akhir ini juga mengacu pada teori yang sudah ada dan digunakan sebelumnya.

### 2.2.1. Layanan Pelanggan Perusahaan Transportasi

Klasifikasi merupakan sebuah teknik pembelajaran mesin yang digunakan untuk memprediksi atau mengelompokkan data ke dalam suatu kelas [4]. Klasifikasi teks atau merupakan suatu proses kategorisasi teks atau dokumen ke dalam sebuah kategori yang sudah ditentukan. Klasifikasi teks terdiri dari representasi dokumen, transformasi fitur dan / atau pemilihan fitur, konstruksi model ruang vektor, penerapan algoritma penggalian data, dan yang terakhir evaluasi algoritma penggalian data yang diterapkan [5]. Informasi yang sudah diekstraksi setelah klasifikasi teks dihubungkan satu dengan lainnya untuk menyajikan fakta atau hipotesis baru.

Klasifikasi teks dapat dilakukan secara manual dan otomatis. Pengklasifikasian secara otomatis memiliki tiga metode populer, yaitu menggunakan sistem *rule-based*, pembelajaran mesin, dan *hybrid* (Campuran *rule-based* dan pembelajaran mesin). Pada tugas akhir ini klasifikasi teks dilakukan dengan menggunakan pembelajaran mesin, khususnya yaitu dengan metode *deep learning*. Metode pembelajaran mesin menggunakan algoritma yang membangun pengklasifikasi dibagi menjadi dua bagian, yaitu *training* (pelatihan) dan testing (pengujian). Pada bagian *training* mesin akan mempelajari data

yang sudah diberikan label dan dilakukan testing untuk data yang tidak diketahui kelasnya.

### 2.2.2. Praproses Data

Tahap ini merupakan tahap awal dalam melakukan klasifikasi. Pada umumnya data mentah tidak dapat dilakukan atau kurang cocok untuk dilakukan klasifikasi sehingga hasilnya kurang baik. Sehingga data harus disiapkan dengan berbagai macam algoritma praproses, bahkan terkadang harus dilakukan secara manual [6].

***1. Text Cleaning***

Tahap pertama dari praproses data ini adalah melakukan *text cleaning* atau membersihkan data dari teks yang tidak dibutuhkan dan cenderung membuat model menjadi kurang optimal. Berikut beberapa tahap pembersihan teks:

a. *Special Character* dan *Punctuation*. Pada proses ini menghilangkan karakter seperti tanda petik (“) atau karakter lainnya dan juga tanda lainnya seperti tanda tanya (“?”), tanda seru (“!”), dan lain-lain.
b. *Case Folding*. Pada proses ini kita mengubah semua huruf kapital menjadi huruf kecil.
c. *Stemming*. Proses ini merupakan proses mengubah kata dengan imbuhan, awalan dan sisipan ke dalam bentuk kata dasar.
d. *Stop words*. Beberapa kata dalam teks tidak memiliki andil dalam proses prediksi nantinya. Contohnya seperti konjungsi, kata konjungsi tidak berpengaruh karena hanya sebagai penghubung antar kata. Oleh karena itu kata yang termasuk kriteria diatas akan dieliminasi.

**2. Tokenisasi dan Vektorisasi**

Tokenisasi dan Vektorisasi Pada tahap ini melakukan proses tokenisasi yaitu membagi teks menjadi beberapa bagian sub-teks yang lebih kecil atau ke dalam kata. Proses ini akan lebih memperjelas kategori yang diklasifikasikan. Selanjutnya setelah dilakukan tokenisasi, kata atau teks yang sudah terbagi-bagi akan diberikan vektor atau nilai untuk masig-masing.

### 2.2.3. *Weighted Word Embedding*

*Word embedding* merupakan suatu metode penyisipan kata, mengubah kata menjadi suatu vektor yang kontinyu dengan panjang yang sudah ditetapkan, sehingga tidak akan dibatasi dengan kosakata yang lebih banyak. Pembobotan kata merupakan suatu strategi praproses data dengan menetapkan bobot yang sesuai untuk setiap istilah sehingga bobot mewakili relevansi istilah dengan dokumen, model ini memainkan peran penting dalam meningkatkan kinerja klasifikasi teks. Model pembobotan tradisional, seperti *Term Frquency* (TF) dan *Term Frequency Inverse Document Frequency* (TF-IDF), bersifat *unsupervised* [7], yang berarti model tersebut tidak menggunakan informasi label dokumen untuk menghitung bobot. Semua metode pembobotan yang biasa digunakan mengikuti aturan yang sama, yaitu, satu istilah hanya memiliki satu bobot, yang secara implisit mengasumsikan bahwa istilah yang sama juga memiliki makna yang sama dalam dokumen yang berbeda. Namun, sangat umum bahwa istilah muncul dalam dokumen dengan label berbeda dan dalam tiap dokumen, masing-masing kata dapat memiliki makna (kepentingan) yang berbeda. Untuk mengatasi kekurangan tersebut, digunakanlah strategi pembobotan baru untuk klasifikasi teks dimana setiap istilah akan diberikan bobot yang berbeda tergantung pada kelasnya. Model TF-IDF tetap digunakan untuk melakukan pembobotan ini karena modelnya sederhana dan mempunyai kinerja yang baik [8]. Proses *weighted word embedding* dapat dibagi menjadi dua bagian proses, yaitu komputasi pemberian bobot ke masing-masing kata dan komputasi pembuatan

*weighted word embedding*. *Weighted word embedding* dapat dihasilkan dari perkalian antara matriks ***W***, hasil atau vektor dari pembobotan tiap kata dalam satu dokumen, dan matriks ***E***, matriks transpose yang berisikan word embedding yang didapat secara acak ataupun menggunakan pre-trained vectors yang dibuat oleh Word2Vec dan Glove Model.

2.2.3.1. TF-IDF

TF-IDF merupakan suatu pengukuran statistik yang digunakan secara luas untuk mengevaluasi seberapa pentingkah sebuah kata bagi sebuah dokumen dalam sebuah korpus [7]. Contoh, diberikan kosakata istilah $\mathfrak{V}$ dan kumpulan dokumen $\mathfrak{D}$, bobot TF-IDF untuk suatu istilah $t \in \mathfrak{V}$ dan dokumen $d \in \mathfrak{D}$ dikomputasikan dalam perhitugan berikut:

$$TF(t,d) = f_{t,d},$$

$$IDF(t,d) = \log\left(\frac{|\mathfrak{D}|}{m}\right) + 1, \tag{2.1}$$

$$TF\text{-}IDF(t,d) = TF(t,d) \times IDF(t,d)$$

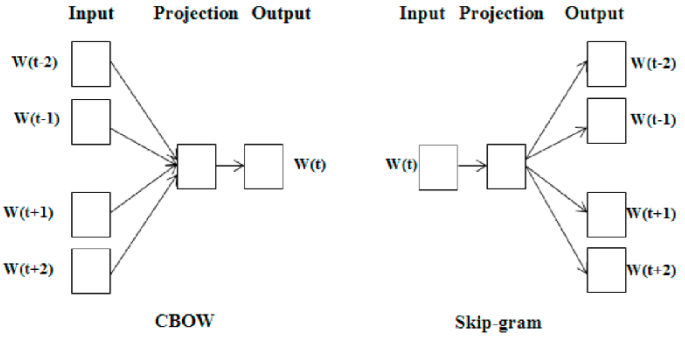
Dimana, $f_{t,d}$ = Frekuensi munculnya istilah t pada dokumen d, $|\mathfrak{D}|$ merepresentasikan jumlah dokumen sedangkan m adalah jumlah dokumen pada $\mathfrak{D}$ yang memiliki istilah t.

2.2.3.2. Word2Vec

Pada umumnya *word embedding* merupakan suatu permodelan bahasa dan fitur teknik pembelajaran di *Natural Language Processing* di mana kata atau frasa diwakili dalam bentuk vektor bilangan real. Konsep *word embedding* melibatkan rumus matematika. Model yang digunakan dalam word embedding sangat beragam, salah satunya adalah model Word2Vec. Word2Vec merepresentasikan kata-kata menjadi vektor berdasarkan beberapa fitur yang dimilikinya seperti ukuran dan dimensi vektor.

Model Word2Vec diusulkan oleh Mikolov dengan keunggulan representasi vektor ini mampu menangkap sintaks dan makna semantik dari kata menggunakan *NLP (Natural Language Programming)*. Selain itu, model Word2Vec adalah algoritma representasi vektor kata yang mampu mencapai kinerja terbaik di *NLP* dengan mengelompokan kata yang sama, yaitu kata-kata yang sama memiliki vektor yang sama atau vektor berdekatan. Nilai kesamaan dari vektor memiliki rentang dari -1 hingga 1 (tertinggi).

Arsitektur model Word2Vec dihitung dengan menggunakan *Neural Network* dengan kata teks sebagai input dan ruang vektor sebagai outputnya. Vektor kata yang dihasilkan adalah vektor ruang dimensi yang menangkap makna semantik kata tersebut. Ada dua jenis model arsitektur Word2Vec, yaitu model *Skip-Gram* dan *CBOW (Continuous Bag of Words)*. Model *Skip-gram* diperkenalkan sebagai metode yang efisien untuk mempelajari vektor kata dalam teks yang tidak terstruktur dalam jumlah yang besar. Arsitektur model *Skip-gram* membuat prediksi kata dengan hanya menggunakan sebagian kata atau bahkan satu kata dalam suatu teks yang tidak terstruktur, sedangkan model CBOW memprediksi kata berdasarkan kata konteks kata secara keseluruhan. Seperti pada gambar 2.1.




Gambar 2.1 Model Arsitektur Skip-gram dan CBOW

Model Word2Vec dapat dihitung menggunakan nilai vektor kata yang diperoleh dengan menggunakan persamaan *Cosine Similarity*. *Cosine Similarity* adalah perhitungan kesamaan antara dua vektor n-dimensi dengan mencari nilai kosinus dari sudut antara keduanya dan sering digunakan untuk membandingkan dokumen dalam penggalian teks. Rumus *Cosine Similarity* adalah sebagai berikut:

$$Cos\ \theta = \frac{\bar{x} \odot \bar{y}}{\|\bar{x}\|\|\bar{y}\|} \quad (2.2)$$

Dimana perkalian vektor x dan y dibagi dengan hasil perkalian dari masing-masing panjang vektornya.

2.2.3.3. FastText

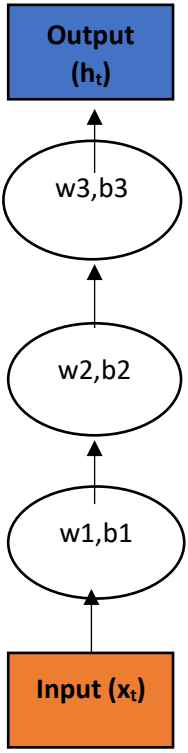
FastText merupakan suatu model *word embedding* yang dikembangkan oleh Facebook. FastText ini adalah pengembarngan dari model *word embedding* Word2Vec. FastText mampu mencapai kinerja yang sangat baik untuk representasi kata dan klasifikasi kalimat, khususnya dalam kasus kata-kata langka dengan memanfaatkan informasi tingkat karakter.

Pada model FastText setiap kata direpresentasikan dengan kumpulan karakter dengan panjang *n-gram*. Semisal kata ‘representasi’, dengan nilai *n-gram* = 3 maka pada model FastText akan diproses mejadi (‘rep’, ‘epr’, ‘pre’, ‘res’, ‘ese’,’sen’, ‘ent’, ‘nta’, ‘asi’). Dengan merepresentasikan kata menjadi karakter sejumlah *n-gram*, model ini mampu mengklasifikasikan kata yang langka dan juga mendeteksi apabila terdapat kesalahan penulisan dalam kata.

### 2.2.4. *Recurrent Neural Network*

*Recurrent neural network* merupakan metode pembelajaran mesin yang berdasar pada pembelajaran jaringan syaraf (*neural network*) yang menyerupai sistem otak manusia dalam menangkap pola. RNN adalah satu dari sekian banyak metode jaringan syaraf yang dapat digunakan. Berbeda dengan metode

jaringan syaraf lainnya RNN terkenal dengan kemampuannya melatih prediksi berdasarkan prediksi sebelumnya [2]. Hal ini dikarenakan RNN memiliki ”memori” yang menyimpan informasi dari hasil kalkulasi sebelumnya. Umumnya metode jaringan syaraf memiliki *input* dan *output* yang independen, tidak bergantung satu sama lain, namun beberapa kasus butuh kesinambungan antara satu kalimat dengan kalimat lain yang akan diprediksi. Dengan menggunakan *hidden layer* RNN masalah tersebut dapat diselesaikan. RNN mampu menangkap konteks secara luas dari data karena karakteristiknya yang memiliki dependensi antar kata jangka panjang [3].



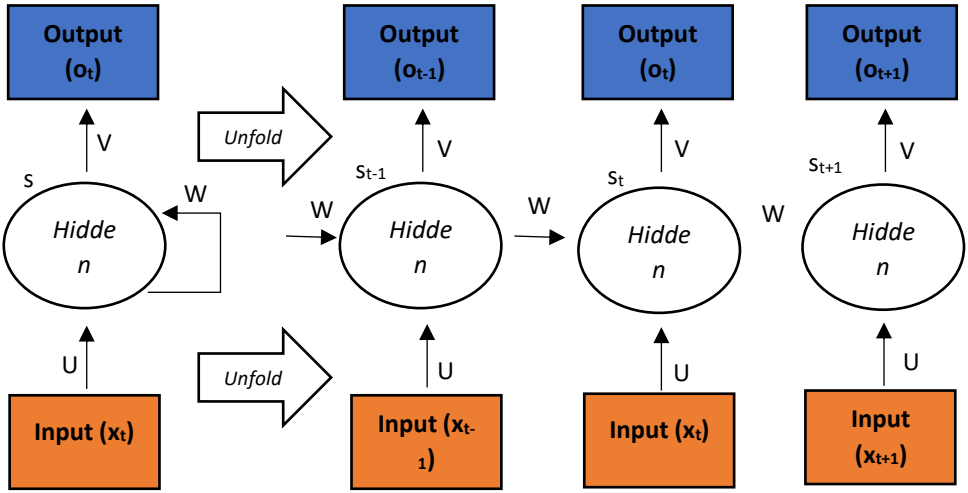

Gambar 2.2 Contoh Arsitektur RNN

Pada metode jaringan syaraf tradisional, umumnya terdapat banyak *hidden layer* dengan masing-masing bobot yang berbeda dapat dilihat pada gambar 2.2. Pada gambar 2.2 tiap *hidden layer* memiliki nilai bobot dan bias yang berbeda, sehingga *layer* independen antara satu dan lainnya dan tidak menyimpan "memori" dari hasil *output* sebelumnya.

RNN hanya terdapat satu *hidden layer*, dimana *hidden layer* tersebut melakukan perulangan. Perulangan pada *hidden layer* atau *recurrent layer* dapat memiliki bobot neuron yang sama maupun berbeda. Pada *layer* tersebut dilakukan *task* yang sama secara berulang pada tiap elemen secara berurut, setelah itu berdasarkan komputasi sebelumnya akan dilakukan hal yang sama untuk komputasi output selanjutnya. Hal ini membuat RNN dapat memproses data sekuensial lebih baik dan hasil yang signifikan saat melakukan klasifikasi teks [9].

Secara teoritis, RNN seharusnya dapat melakukan pembelajaran mesin secara sekuensial yang sangat panjang mungkin tak hingga. Namun, model ini hanya dapat melakukan perulangan kembali beberapa langkah saja dalam praktik nyatanya karena adanya gradien yang hilang sehingga membuatnya sulit untuk melakukan proses pembelajaran jangka panjang.

Pada gambar 2.3 merupakan penggambaran skema dari RNN. Setiap RNN memiliki perulangan, apabila dalam 1 kalimat terdapat 3 kata maka akan terdapat 3 sekuens *layer neural network* dengan masing-masing 1 *layer* untuk tiap kata, seperti gambar diatas saat skema tersebut dibuka (*unfold*). Pada setiap pembelajaran RNN akan dimulai dengan *input* (xt). Lalu selanjutnya dilakukan perhitungan untuk *hidden state* (s) dengan menggunakan *parameter* (U dan W).




Gambar 2.3 RNN 3 Hidden Layer

Dari perhitungan *hidden state* maka dapat digunakan untuk menghitung *ouput* (ot). Berikut rumus-rumus yang digunakan untuk perhitungan:

$$Current\ State\ (s_t) = f(Ux_t + Ws_{t-1}) \quad (2.3)$$

$$Ouput\ (o_t) = V.s_t \quad (2.4)$$

Parameter U,V,dan W diatas memiliki nilai yang sama pada tiap *layer*-nya. Hal inilah yang membedakan RNN dengan *deep neural network* lainnya. Fungsi f pada rumus perhitungan Current State 2.3 dapat menggunakan *tanh* atau *ReLu*.

### 2.2.5. Pengukuran Kinerja

Setelah membuat model klasifikasi, langkah selanjutnya melakukan pegujian terhadap model yang sudah dibuat. Dalam melakukan pengukuran kinerja, hal yang paling umum dilakukan adalah menghitung *precision, recall, f-measure, error,* dan akurasi [10]. Perhitungan diatas menggunakan variabel dari hasil *confussion matrix* yang dapat dilihat dari

tabel 2.1, yaitu *true positive*, *false positive*, *true negative*, dan *false negative.*

Tabel 2.1 *Confussion Matrix*

| Hasil Prediksi | Positive | Negative | |
|---|---|---|---|
| Positive | TP (*True Positive*) | FP (*False Positive*) | TP + FP (Jumlah dokumen yang dimasukkan ke dalam kelas) |
| Negative | FN (*False Negative*) | TN (*True Negative*) | FN + TN (Jumlah dokumen yang tidak termasuk dalam kelas) |
| | TP + FN (Jumlah dokumen yang tepat terklasifikasi) | FP + TN (Jumlah dokumen yang kurang tepat terklasifikasi) | n = TP + FP + TN + FN (jumlah total dokumen yang digunakan) |

Diberikan,

TP = Dokumen yang masuk ke dalam kelas yang tepat

FP = Dokumen yang masuk ke dalam kelas yang tidak tepat

TN = Dokumen yang tertolak ke dalam kelas dan tepat

FN = Dokumen yang tertolak ke dalam kelas dan kurang tepat

#### 2.2.5.1. *Precision*

Perhitungan ini mengukur ketepatan model dalam melakukan klasifikasi. Berikut rumus yang digunakan untuk mengukur,

$$precision = \frac{TP}{TP + FP} \times 100 \quad (2.5)$$

*Precision* merupakan hasil perbadingan dari dokumen yang relevan, tepat sasaran kelas, dengan total dokumen yang dimasukkan ke dalam kelas. Semakin tinggi presentase dari precision maka model memberikan hasil yang lebih relevan.

#### 2.2.5.2. *Recall*

Perhitungan ini bertujuan untuk mengukur tingkat keberhasilan model dalam memberikan hasil yang selalu relevan. Berikut rumus yang digunakan untuk mengukur,

$$recall = \frac{TP}{TP + FN} \times 100 \quad (2.6)$$

*Recall* merupakan hasil perbandingan antara dokumen yang relevan dengan dengan keseluruhan dokumen yang secara tepat teklasfikasi dalam model.

#### 2.2.5.3. *F-measure*

Perhitungan ini merupakan pengukuran evaluasi yang mengkombinasikan nilai *precision* dan *recall* dari model. Berikut rumus yang digunakan untuk mengukur,

$$f - measure = \frac{2 \times precision \times recall}{precision + recall} \quad (2.7)$$

#### 2.2.5.4. Akurasi

Akurasi merupakan perhitungan untuk mengukur kedekatan data hasil prediksi dengan data aktual yang dimiliki. Perhitungan ini mengukur ketepatan model untuk memprediksi dan membandingkan dengan data aktual. Berikut rumus yang digunakan untuk mengukur,

$$akurasi = \frac{TP + TN}{TP + FP + TN + FN} \times 100 \qquad (2.9)$$

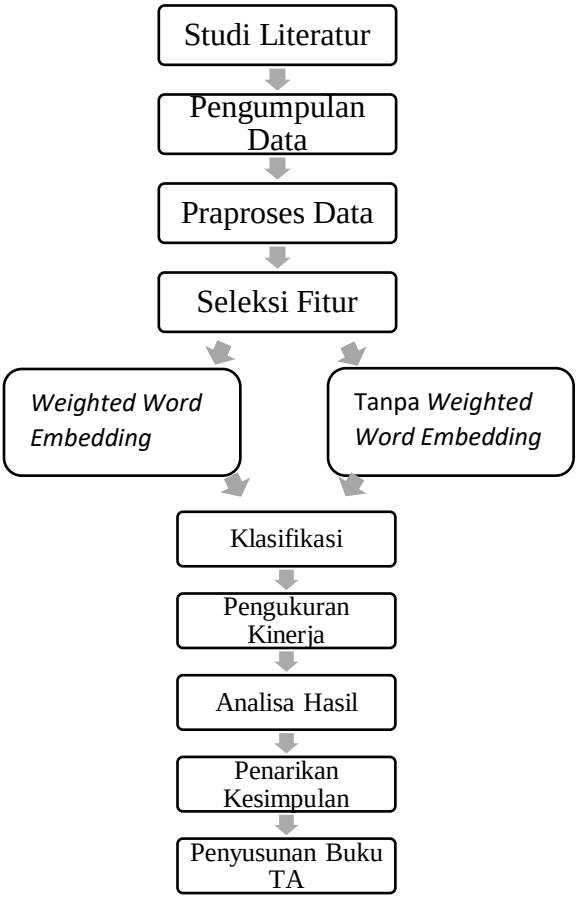
Dalam mengukur akurasi yang dilakukan adalah membandingkan dokumen yang relevan dengan total sampel. Semakin tinggi nilai akurasi menunjukkan model memiliki performa yang bagus.

# BAB 3
# METODE PENGERJAAN TUGAS AKHIR

Bab ini menjelaskan mengenai bagaimana alur pengerjaan dari penelitian yang dilakukan berserta penjelasan dari setiap tahap pengerjaan yang dilakukan.

## 3.1. Metodologi

Metodologi ini akan mensistematiskan langkah-langkah pengerjaan tugas akhir. Pada diagram tahapan dibawah ini merupakan alur dari pelaksanaan tugas akhir.




Gambar 3.1 Diagram Metodologi

## 3.2. Uraian Metodologi

Pada bagian ini akan diuraikan aktivitas yang dilakukan dalam pelaksanaan metodologi.

### 3.2.1 Studi Literatur

Pada tahapan pertama pengerjaan tugas akhir ini dilakukan studi literatur. Penulis mempelajari beberapa penelitian terdahulu yang terkait dengan penelitian yang akan dilakukan. Studi literatur ini diambil dari sumber valid dan kredibel berupa *paper*, jurnal, buku, berita dan sumber lainnya yang dianggap relevan dengan penelitian yang sedang peneliti lakukan. Studi literatur dilakukan untuk menambah wawasan ilmu untuk bidang minat topik tugas akhir yang diminati. Selain itu dengan melakukan studi literatur, membuka peluang kita untuk menemukan kemungkinan topik-topik lain yang dapat dijadikan topik tugas akhir. Penulis melakukan studi literatur mengenai klasifikasi terutama klasifikasi teks, metode klasifikasi *Recurrent Neural Network* (RNN), praproses teks, dll. Hasil dari studi literatur ini adalah dapat ditentukan tujuan dari penelitian yang dilakukan, topik penelitian yang akan diambil, dan tentu saja wawasan ilmu pengetahuan mengenai topik yang akan diambil. Dengan didapatkannya wawasan ilmu yang lengkap mengenai topik yang diambil, maka memungkinkan untuk menghasilkan penelitian yang balk. Dan dari hasil studi literatur ini pula, penulis menentukan bahwa topik tugas akhir yang diambil adalah mengenai penggalian data dari dokumen teks menggunakan metode klasifikasi *Recurrent Neural Network* (RNN).

### 3.2.2. Pengumpulan Data

Pada tahap ini penulis mengumpulkan data-data yang dibutuhkan dalam melakukan penelitian. Data yang digunakan yaitu dokumen teks dari akun Twitter layanan pengaduan milik PT Kereta Api Indonesia. Data Twitter ini merupakan data periode selama 15 bulan dari bulan Juni 2015 hingga Januari

2016, dengan total data sebanyak 20.204 cuitan. Data dibagi ke dalam tiga kategori, yaitu keluhan, saran, dan informasi. Data selanjutnya akan dilakukan proses praposes agar siap dan dapat diklasifikasi. Selain itu penulis juga membuat daftar *stopword* yang disesuaikan dengan studi kasus.

### 3.2.3. Praproses Data

Pada tahap ini dilakukan proses pengolahan data. Data pada penelitian terdapat duplikasi dan ada juga beberapa variabel pada data yang tidak diperlukan sehingga kedua jenis data tersebut dapat dihapus. Ada pula data yang bernilai *null*. Data seperti ini dapat diganti dengan cara dihapus atau dapat pula diisi dengan mencari nilai rataan, nilai tengah atau modus dari data untuk diisikan pada data yang bernilai *null* tersebut. Tahap selanjutnya data dibagi menjadi dua, yaitu data *training* dan data *testing*. Data *training* diberikan proporsi 2/3 dari keseluruhan data, sedangkan data *testing* sisanya yaitu 1/3. Data *training* ini yang nantinya akan digunakan untuk membuat model klasifikasi. Data *testing* akan digunakan untuk menguji model yang dibuat dari data *training*.

### 3.2.4. Praproses Teks

Pada tahap dilakukan praproses teks untuk membuat data teks siap diolah ke dalam mesin pembelajaran. Langkah-langkahnya seperti berikut,

1. Tokenisasi
   Pada tahap ini memisahkan suatu kalimat atau paragraf menjadi kata. Semisal, “Terjadi keterlambatan pada kereta Argo Bromo” maka setelah tokenisasi menjadi {“Terjadi”, keterlambatan”, “pada”, “kereta”, “Argo”, ”Bromo”}.
2. Normalisasi
   Pada tahap ini dilakukan pembersihan *special character* dan *punctuation*. Seperti, “@”, “!”, “?”, dan lain-lain.

3. *Case Folding*
   Pada tahap ini akan mengubah huruf-huruf yang kapital menjadi huruf kecil.
4. *Stemming*
   Proses ini merupakan proses mengubah kata dengan imbuhan, awalan dan sisipan ke dalam bentuk kata dasar. Misal, “kekurangan” menjadi “kurang”.
5. Membuat Daftar *Stop Words*
   Pada proses ini akan dibuat sebuah daftar kata-kata ang tidak digunakan. Beberapa kata dalam teks tidak memiliki andil dalam proses prediksi nantinya. Contohnya seperti konjungsi, kata konjungsi tidak berpengaruh karena hanya sebagai penghubung antar kata. Oleh karena itu kata yang termasuk kriteria diatas akan dieliminasi.
6. *Replace Slang Words*
   Pada proses ini akan melakukan penggantian kata-kata yang tidak sesuai dengan kamus, kembali ke dalam kata baku. Misal, “gak” menjadi “tidak”, “m4cet” menjadi “macet”, dan lain-lain. Daftar *slang words* didapatkan dari penelitian sebelumnya [11].

Pada tahap ini juga dilakukan pemilihan fitur kata yang akan digunakan dalam proses klasifikasi teks yang bertujuan untuk meningkatkan kinerja sistem dan akurasi proses klasifikasi teks. Karena dengan meningkatkan kinerja sistem pengklasifikasi, maka akan rnenghemat biaya dan waktu yang digunakan dalam proses tersebut. Praproses data dilanjutkan dengan melakukan pembobotan pada tiap kata. Proses ini pembobotan akan dilakukan sebanyak dua kali, yaitu pertama menggunakan pembobotan biasa dan yang kedua melakukan pembobotan dengan metode *weighted word embedding.*

Pada pembobotan biasa akan menggunakan metode yang sederhana yaitu TF-IDF. Pembobotan ini hanya akan

memberikan bobot yang sama pada tiap kata yang sama meskipun berada di kelas yang berbeda.
Selain itu, data juga akan diberikan pembobotan dengan metode *weighted word* embedding. Pembobotan ini dilakukan dengan cara memberikan bobot berbeda pada setiap kata. Meskipun kata tersebut sama, namun apabila terdapat di kelas yang berbeda maka akan mendapatkan bobot berbeda pula. Pembobotan pada tiap kata memiliki nilai yang berbeda bergantung kepada kepentingan kata tersebut pada kelas yang dikategorikan.

3.2.4.1. *Weighted Word Embedding*
Pada tahap ini, akan terbagi menjadi dua fase yang pertama yaitu pemberiat bobot pada kata menggunakan TF-IDF. Lalu, mengkalikan tiap bobot pada pemetaan vektor kata yang dilakuan dengan *Word2Vec Model*. Pada tahap pembobotan tiap kata akan diberikan *weight* (bobot). Pembobotan menggunakan metode TF-IDF, dimana pembobotan dilakukan berdasarkan frekuensi munculnya suatu kata dan jumlah kata penting yang terdapat dalam tiap dokumen.

Selanjutnya yaitu melakukan pemetaan tiap kata menjadi suatu vektor. Pada tahap ini menggunakan model *Word2Vec*. Dari bobot yang sudah didapat dari hasil kalkulasi TF-IDF diatas, bobot dari tiap kata tersebut dicocokkan dengan bobot dari vektor hasil dari model *Word2Vec*. Dari perkalian dua matriks diatas maka akan menghasilkan suatu kumpulan vektor atau *array* yang nantinya akan menjadi *input* dari model klasifikasi *Recurrent Neural Network*.

Selain menggunakan model Word2Vec, pada tugas akhir ini juga akan membandingkan dengan model vektorisasi menggunakan FastText. Sama seperti *word embedding* yang diberikan bobot. Hasil dari *word embedding* menggunakan FastText dikalikan hasil kalkulasi dari TF-IDF. Selanjutnya akan tercipta data vektor sesuai dengan panjang yang kita atur.

### 3.2.5. Klasifikasi

Pada tahap ini, diakukan proses klasifikasi dari dokumen Twitter yang telah diolah pada proses praproses teks sebelumnya. Proses klasifikasi akan menggunakan Python. Metode yang digunakan adalah metode RNN. Proses klasifikasi ini akan dilakukan sebanyak dua kali. Pertama, akan dilakukan klasifikasi menggunakan hasil pemobotan tradisional. Lalu, akan dilakukan klasifikasi dengan menggunakan pembobotan *weighted word embedding.*

Dalam proses klasifikasi, terdapat dua tahapan yang harus dilakukan. Yang pertama adalah tahap pelatihan (*training*) yang bertujuan untuk menemukan model dari klasifikasi yang dilakukan. Selanjutnya model yang didapatkan dari tahap pelatihan digunakan untuk melakukan klasifikasi pada tahap pengujian (*testing*). Kemudian dilakukan evaluasi terhadap model yang dibuat pada tahap pelatihan dan juga dilakukan evaluasi dari hasil klasifikasi yang dihasilkan dari tahap pengujian.

#### 3.2.5.1 Skenario Model

Pada pengerjaan tugas akhir ini akan dilakukan percobaan pada beberapa skenario model. Masing-masing skenario model akan memiliki parameter yang berbeda-beda yang bertujuan untuk mendapatkan hasil klasifikasi yang terbaik. Skenario model dengan hasil performa yang paling baik akan dipilih untuk melakukan prediksi model.

#### 3.2.5.2. Validasi Model

Setelah ditemukan skenario model dengan performa yang paling bagus. Selanjutnya model tersebut akan divalidasi terlebih dahulu menggunakan data validasi yang diambil dari data pelatihan.

### 3.2.6. Pengukuran Kinerja

Pada tahap ini dilakukan evaluasi terhadap model hasil klasifikasi apakah telah akurat dan optimal. Metode yang digunakan yaitu dengan menghitun akurasi, *precision*, *recall, f-measure,* dan *error*.

### 3.2.7. Analisa Hasil

Pada tahap ini melakukan analisa dari hasil pengujian yang dilakukan terhadap model yang dibuat. Berdasarkan dari penelitian yang dilakukan dapat dilakukan komparasi antar model yang dibuat dengan hasil dari penelitian sebelumnya. Analisa ini bertujuan untuk menentukan model yang paling akurat dan memiliki performa yang paling bagus diantara model lainnya.

### 3.2.8. Penarikan Kesimpulan

Pada tahap selanjutnya, dapat ditarik kesimpulan dari penelitian yang sudah dilakukan. Hasil analisa yang didapat dituangkan ke dalam bentuk kesimpulan dari penelitian. Diberikan juga saran apabila akan dilakukan penelitian lanjutan.

### 3.2.9. Penyusunan Buku TA

Tahapan akhir dari penelitian ini adalah pembuatan buku laporan tugas akhir sebagai bentuk dokumentasi serta pembelajaran dari penelitian. Di dalam laporan tugas akhir ini berisikan:

a. BAB I Pendahuluan

   Dalam bab pendahuluan, akan dipaparkan proses identifikasi masalah penelitian mengenai latar belakang masalah, perumusan masalah, batasan masalah, tujuan tugas akhir, manfaat tugas akhir, dan relevansi tugas akhir dengan bidang keilmuan sistem informasi. Berdasarkan uraian pada bab ini, harapannya gambaran umum permasalahan dan solusi masalah pada tugas akhir dapat dipahami.

b. BAB II Tinjauan Pustaka

Pada bab ini akan dilakukan pembahasan terkait dengan penelitian maupun studi literatur sebelumnya yang berkaitan dan dijadikan sebagai acuan selama pengerjaan tugas akhir serta landasan teori yang berkaitan dengan tugas akhir yang dapat membantu pemahaman selama pengerjaan tugas akhir ini.

c. BAB III Metodologi

Dalam bab metodologi ini akan dipaparkan tahapan yang akan dilaksanakan selama pengerjaan tugas akhir. Pada tiap metode yang digunakan bertujuan untuk pegangan agar pengerjaan tugas akhir berjalan secara sistematis. Dan juga dalam tahapan ini dijelaskan secara rinci input, proses dan *output* yang terkait pada tugas akhir.

d. BAB IV Perancangan

Pada bab ini akan dijelaskan mengenai perancangan pelaksanaan metodologi penelitian yang akan dilakukan.

e. BAB V Implementasi dan Hasil

Pada bab ini akan dijelaskan mengenai proses pelaksanaan metodologi penelitian yang telah dirancang dan juga mengenai hasil dan analisa dari implementasi metode dalam penelitian ini.

f. BAB VI Kesimpulan dan Saran

Bab ini akan berisikan kesimpulan dari seluruh aktivitas dan hasil dari penelitian ini serta saran-saran yang dapat diberikan untuk pengembangan penelitian selanjutnya.

## 3.3. Jadwal Pengerjaan

Dalam proses pengerjaan tugas akhir ini terdapat jadwal pengerjaan. Berikut rincian jadwal kegiatan selama penelitian tugas akhir berlangsung.

Tabel 3.1 Jadwal Pengerjaan

| No. | Kegiatan | Bulan 1 | | | | Bulan 2 | | | | Bulan 3 | | | | Bulan 4 | | | | Bulan 5 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| 1 | Studi Literatur | | | | | | | | | | | | | | | | | | | | |
| 2 | Pengumpulan Data | | | | | | | | | | | | | | | | | | | | |
| 3 | Praproses Data | | | | | | | | | | | | | | | | | | | | |
| 4 | Seleksi Fitur | | | | | | | | | | | | | | | | | | | | |
| 5 | Klasifikasi | | | | | | | | | | | | | | | | | | | | |
| 6 | Pengukuran Kinerja | | | | | | | | | | | | | | | | | | | | |
| 7 | Analisa Hasil dan Penarikan Kesimpulan | | | | | | | | | | | | | | | | | | | | |
| 8 | Penyusunan Buku Tugas Akhir | | | | | | | | | | | | | | | | | | | | |

*“Halaman ini sengaja dikosongkan”*

# BAB 4
# PRAPROSES DATA DAN PERANCANGAN MODEL *WEIGHTED WORD EMBEDDING*

Dalam bab ini menjelaskan rancangan penelitian tugas akhir dalam proses pengumpulan data, gambaran input dan output, serta proses pengolahan data menggunakan Python. Hasil pengolahan akan berupa model klasifikasi.

## 4.1. Pengumpulan Data

Data yang digunakan pada pengerjaan tugas akhir ini merupakan data pengaduan perusahaan transportasi. Data transportasi yang masuk melalui akun media sosial Twitter. Data yang digunakan pada studi kasus ini merupakan data pengaduan yang masuk pada akun media sosial Twitter PT. Kereta Api Indonesia (Persero) yaitu @KAI121. Batasan dari data yang digunakan adalah "cuitan" yang berbahasa Indonesia. Data yang dikumpulkan merupakan "cuitan" yang masuk dari rentang antara 2015 hingga 2016. Data berisikan 20.190 "cuitan" yang masih berupa data mentah dan belum di praproses. Data pengaduan sudah dikategorikan secara manual oleh tim *Customer Relation* dan *Digital Community* PT. Kereta Api Indonesia (Persero). Data dikategorikan menjadi 3, yaitu informasi, keluhan dan saran.

## 4.2. Praproses Data

Pada tahap awal ini, data mentah akan melalui praproses data. Praproses data ini meliputi hal-hal mendasar seperti menghapus data yang duplikat, data *null*¸dan membagi data menjadi *train* dan *test set.* Lalu dilanjutkan dengan melakukan praproses teks yang meliputi *casefolding*, *Text Cleaning*, menghapus *stopword*, *stemming*, dan tokenisasi. Dalam melakukan praproses ini digunakan *library Pandas*, *library String*, dan juga *library Regular Expression*.

### 4.2.1. Casefolding

Pada tahap ini dilakukan perubahan huruf-huruf kapital menjadi huruf kecil. Hal ini dilakukan agar menyelaraskan seluruh kata agar tidak ada yang terduplikasi (misal: Saya dan saya). Syntax yang digunakan untuk *casefolding* yaitu `data["Tweet"]= data["Tweet"].str.lower()`. Kode tersebut merupakan kode yang digunakan untuk melakukan c*asefolding. Casefolding* ini menggunakan *method str.lower* milik *Pandas Library*. Berikut contoh hasil dari proses *casefolding* yang dapat dilihat pada tabel 4.1

Tabel 4.1 Hasil *Casefolding*

| **No** | **Sebelum** | **Sesudah** |
|---|---|---|
| 1. | Siang, di KAI bisa untuk intern ngga? masih kuliah semester akhir fakultas manajemen. Trims | siang, di kai bisa untuk intern ngga? masih kuliah semester akhir fakultas manajemen. trims |
| 2. | @KAI121 sy berniat beli tiket untk 6 org, mau tanya no kursi 6 penumpang (3 kursi) kereta eko progo yg berhadap2an no 17-18 atau 18-19? Tks | @kai121 sy berniat beli tiket untk 6 orang, mau tanya no kursi 6 penumpang (3 kursi) kereta eko progo yang berhadap2an no 17-18 atau 18-19? tks |
| 3. | @KAI121 Penasaran. Kalo penumpang telat kan tiket hangus. Nah kalo keretanya yg telat berangkat/tiba ada gak kompensasinya? Eks. lho ini. 2:30 PM - 20 Dec 2015 · | @kai121 penasaran. kalo penumpang telat kan tiket hangus. nah kalo keretanya yang telat berangkat/tiba ada gak kompensasinya? eks. lho ini. 2:30 PM – 20 Dec 2015 |
| 4. | Min @KAI121 ditemukan kasus sepertinya salah | min @kai121 ditemukan kasus sepertinya salah |

| | | |
|---|---|---|
| | tempel huruf kursi min. Di KA Harina sama Kertajaya http://www.kaskus.co.id/show_post/55f7cf4b92523364718b4569/6764/- ? 7:04 PM - 15 Sep 2015 · | tempel huruf kursi min. di ka harina sama kertajaya http://www.kaskus.co.id/show_post/55f7cf4b92523364718b4569/6764/- ? 7:04 pm - 15 Sep 2015 · |

### 4.2.2. Tweet Cleaning

Pada proses ini dilakukan pembersihan teks Twitter. Bentuk teks Twitter dapat berbeda dengan bentuk teks lainnya karena pada teks Twitter terdapat beberapa hal yang jarang ditemukan seperti terdapat *mention username*, *retweet tag* (RT), *hashtag*, tanggal dan waktu, *web url*, serta menghapus tanda baca (seperti: “.” , “?” , “!”,dll.). Format lain yang juga dihapus yaitu angka. Hal-hal tersebut nantinya tidak akan berguna pada saat proses pengklasifikasian karena tidak akan memiliki angka vektor yang signifikan untuk nantinya dimasukkan ke model klasifikasi.

Proses pertama yang dilakukan pada *Tweet Cleaning* ini adalah menghapus *mention username* dan juga *retweet tag*. Pada penggunaan media sosial Twitter penggunaan *mention* dan juga *retweet* sangat lumrah. Supaya “cuitan” yang kita sampaikan masuk ke notifikasi pengguna yang dituju maka biasanya pengguna Twitter akan menggunakan *mention username* pengguna yang dituju.

```
for row in sheet.iter_rows(min_col=2, max_col=2):
    for cell in row:
        text = str(cell.value)
        text = text+' '
        patt = re.compile('-[a-zA-Z]\w+(\s*)')
        rt_patt = re.compile('rt\s')
        cell.value = rt_patt.sub(' ', patt.sub(' ', text))
```

Kode Program 4.1 *Mention* dan *Retweet*

Pada kode di atas dilakukan iterasi pada kolom dua yaitu kolom “Tweet” yang berisikan pengaduan melalui Twitter. Untuk menghapus *username* Twitter, kode di atas menggunakan modul operasi *Regular Expression* (re). Modul ini menyediakan

operasi pencocokan pada *string Unicode* dan juga *string* 8-bit.. Fungsi-fungsi dalam modul ini memungkinkan untuk memeriksa apakah *string* tertentu cocok dengan *pattern* yang diberikan ataupun sebaliknya. Dari kode di atas dibuat 2 *pattern*, yaitu "patt", untuk mencari pola yang menyerupai *username* Twitter terdiri dari rangkaian huruf a-z dan juga angka serta karakter lainnya. *Pattern* kedua yaitu "rt_patt", untuk mencari pola *string* 'rt' pada setiap "cuitan". Nantinya, setiap pola tersebut akan digantikan oleh *whitespace* (karakter spasi putih). Hasil dari kode di atas dapat dilihat pada tabel 4.2.

Tabel 4.2 Hasil *Remove Username* dan *Retweet*

| **No** | **Sebelum** | **Sesudah** |
|---|---|---|
| 1. | siang, di kai bisa untuk intern ngga? masih kuliah semester akhir fakultas manajemen. trims | siang, di kai bisa untuk intern ngga? masih kuliah semester akhir fakultas manajemen. trims |
| 2. | @kai121 sy berniat beli tiket untk 6 orang, mau tanya no kursi 6 penumpang (3 kursi) kereta eko progo yang berhadap2an no 17-18 atau 18-19? tks | sy berniat beli tiket untk 6 orang, mau tanya no kursi 6 penumpang (3 kursi) kereta eko progo yang berhadap2an no 17-18 atau 18-19? tks |
| 3. | @kai121 penasaran. kalo penumpang telat kan tiket hangus. nah kalo keretanya yang telat berangkat/tiba ada gak kompensasinya? eks. lho ini. 2:30 PM – 20 Dec 2015 | penasaran. kalo penumpang telat kan tiket hangus. nah kalo keretanya yang telat berangkat/tiba ada gak kompensasinya? eks. lho ini. 2:30 PM – 20 Dec 2015 |
| 4. | min @kai121 ditemukan kasus sepertinya salah tempel huruf kursi min. di ka harina sama kertajaya http://www.kaskus.co.id/show_post/55f7cf4b925233 | min ditemukan kasus sepertinya salah tempel huruf kursi min. di ka harina sama kertajaya http://www.kaskus.co.id/show_post/55f7cf4b925233 |

| | | |
|---|---|---|
| | 64718b4569/6764/- ? 7:04 pm - 15 Sep 2015 · | 64718b4569/6764/- ? 7:04 pm - 15 Sep 2015 · |

Proses pembersihan selanjutnya yaitu menghapus tanda baca dan *URL*. Penghapusan tanda baca ini menggunakan *string method str.translate.* Sedangkan untuk menghapus *URL* digunakan *method str.replace.*

```
for row in sheet.iter_rows(min_col=2, max_col=2):
    for cell in row:
        text = str(cell.value)
        cell.value = text.translate(str.maketrans(' ',' ', string.punctuation))
```

Kode Program 4.2 Menghapus Tanda Baca

```
twit = data['Tweet']
data['Tweet'] = twit.replace('((www\.[\s]+)|(https?://[^\s]+))',' ',regex=True)
```

Kode Program 4.3 Menghapus URL

*Method maketrans()* mengubah dan memisahkan kalimat menjadi potongan kata yang selanjutnya disetor ke tabel. Lalu dengan *method translate*, akan mengganti *parameter* yang sudah ditentukan, dalam hal ini tanda baca dengan *whitespace.* Proses penghapusan URL dilakukan menggunakan *library Regular Expression* dengan *method replace*, yaitu mengganti URL dengan *whitespace* seperti pada kode di atas. Hasil dari kode di atas dapat dilihat pada tabel 4.3.

Tabel 4.3 Hasil Menghapus Tanda Baca dan URL

| **No** | **Sebelum** | **Sesudah** |
|---|---|---|
| 1. | siang, di kai bisa untuk intern ngga? masih kuliah semester akhir fakultas manajemen. trims | siang di kai bisa untuk intern ngga masih kuliah semester akhir fakultas manajemen trims |
| 2. | sy berniat beli tiket untk 6 orang, mau tanya no kursi 6 penumpang (3 kursi) kereta eko progo | sy berniat beli tiket untk 6 orang mau tanya no kursi 6 penumpang 3 kursi kereta eko progo yang |

| | | |
|---|---|---|
| | yang berhadap2an no 17-18 atau 18-19? tks | berhadap2an no 17 18 atau 18 19 tks |
| 3. | penasaran. kalo penumpang telat kan tiket hangus. nah kalo keretanya yang telat berangkat/tiba ada gak kompensasinya? eks. lho ini. 2:30 PM – 20 Dec 2015 | penasaran kalo penumpang telat kan tiket hangus nah kalo keretanya yang telat berangkat tiba ada gak kompensasinya eks lho ini 2:30 PM 20 Dec 2015 |
| 4. | min ditemukan kasus sepertinya salah tempel huruf kursi min. di ka harina sama kertajaya <url> 7:04 pm - 15 Sep 2015 · | min ditemukan kasus sepertinya salah tempel huruf kursi min di ka harina sama kertajaya 7:04 pm - 15 Sep 2015 · |

Selanjutnya, dilakukan proses pembersihan *date&timestamp*. Tanggal dan waktu biasanya terdapat pada data Twitter. Namun, tidak semuanya tercantum tanggal dan waktu pada setiap "cuitan".

```
for row in sheet.iter_rows(min_col=2, max_col=2):
    for cell in row:
        text = str(cell.value)
        text = text+' '
        time_patt = re.compile('[0-9]+\s(am|pm)\s')
        time_patt2 = re.compile('[0-9]+(m|s|h)[0-9]+\s(\w+)\sago\s')
        date_patt = re.compile('[0-9]+\s(\w+)\s20[0-9][0-9]\s')
        cell.value = date_patt.sub(' ', time_patt2.sub(' ', time_patt.sub(' ', text)))
```

Kode Program 4.4 Menghapus Date&Timestamp

Pada pembersihan *timestamp* ini juga menggunakan masih menggunakan *regular expression* (re). Setiap *string* yang cocok dengan *pattern* yang dibuat akan digantikan dengan *whitespace*. Hasil dari kode di atas dapat dilihat pada tabel 4.4.

Tabel 4.4 Hasil Menghapus Waktu dan Tanggal

| No | Sebelum | Sesudah |
|---|---|---|
| 1. | siang di kai bisa untuk intern ngga masih kuliah | siang di kai bisa untuk intern ngga masih kuliah |

| | | |
|---|---|---|
| | semester akhir fakultas manajemen trims | semester akhir fakultas manajemen trims |
| 2. | sy berniat beli tiket untk 6 orang mau tanya no kursi 6 penumpang 3 kursi kereta eko progo yang berhadap2an no 17 18 atau 18 19 tks | sy berniat beli tiket untk 6 orang mau tanya no kursi 6 penumpang 3 kursi kereta eko progo yang berhadap2an no 17 18 atau 18 19 tks |
| 3. | penasaran kalo penumpang telat kan tiket hangus nah kalo keretanya yang telat berangkat tiba ada gak kompensasinya eks lho ini 2:30 PM 20 Dec 2015 | penasaran kalo penumpang telat kan tiket hangus nah kalo keretanya yang telat berangkat tiba ada gak kompensasinya eks lho ini |
| 4. | min ditemukan kasus sepertinya salah tempel huruf kursi min di ka harina sama kertajaya 7:04 pm 15 Sep 2015 | min ditemukan kasus sepertinya salah tempel huruf kursi min di ka harina sama kertajaya |

Proses terakhir yaitu menghapus angka-angka yang terdapat pada data teks. Angka dianggap tidak penting dalam melakukan klasifikasi teks sehingga akan dieliminasi.

```
for row in sheet.iter_rows(min_col=2, max_col=2):
    for cell in row:
        text = str(cell.value)
        text = text+' '
        num_patt = re.compile('([0-9]+(\w+)) | ([0-9]+)')
        cell.value = num_patt.sub(' ', num_patt.sub(' ', text))
```

Kode Program 4.5 Menghapus Angka

Dengan menggunakan *regular expression* untuk mencari *pattern* angka, nantinya angka tersebut akan diganti dengan *whitespace*. Hasil dari kode di atas dapat dilihat pada tabel 4.5 berikut ini.

Tabel 4.5 Hasil Menghapus Angka

| No | Sebelum | Sesudah |
|---|---|---|
| 1. | siang di kai bisa untuk intern ngga masih kuliah semester akhir fakultas manajemen trims | siang di kai bisa untuk intern ngga masih kuliah semester akhir fakultas manajemen trims |
| 2. | sy berniat beli tiket untk 6 orang mau tanya no kursi 6 penumpang 3 kursi kereta eko progo yang berhadap2an no 17 18 atau 18 19 tks | sy berniat beli tiket untk orang mau tanya no kursi penumpang kursi kereta eko progo yang berhadap an no atau tks |
| 3. | penasaran kalo penumpang telat kan tiket hangus nah kalo keretanya yang telat berangkat tiba ada gak kompensasinya eks lho ini | penasaran kalo penumpang telat kan tiket hangus nah kalo keretanya yang telat berangkat tiba ada gak kompensasinya eks lho ini |
| 4. | min ditemukan kasus sepertinya salah tempel huruf kursi min di ka harina sama kertajaya | min ditemukan kasus sepertinya salah tempel huruf kursi min di ka harina sama kertajaya |

### 4.2.3. Menghapus *Stopword*

Tahap selanjutnya dalam praproses data adalah menghapus *stopword. Stopword* merupakan kata umum yang sering muncul dan tidak memiliki makna seperti konjungsi, kata tanya, dan lain-lain. Pertama yang harus dilakukan adalah dengan membuat daftar kata *stopword* terlebih dahulu. Nantinya daftar kata tersebut akan dicocokkan dengan kata yang terdapat pada data yang akan diklasifikasi beberapa contoh kata dapat *stopword* dapat dilihat pada tabel 4.6.

Tabel 4.6 Tabel Stopword

| idStopword | Stopword |
|---|---|
| 1 | ada |
| 2 | adalah |
| 3 | adanya |
| 4 | adapun |
| 5 | agak |
| 6 | Agaknya |
| 7 | Agar |
| 8 | Akan |

Proses penghapusan *stopword* dilakukan iterasi pada setiap kolom ‘Tweet’ untuk mencocokkan kata yang terdapat pada daftar *stopword* dengan “cuitan” pada tiap kolom. Nantinya, kata-kata tersebut akan diganti dengan *whitespace*.

```
stop_file = "D:/Tugas Akhir/Data/stopword.xlsx"
stop_data = openpyxl.load_workbook(stop_file)
ssheet = stop_data.active

for row in sheet.iter_rows(min_col=2, max_col=2):
    for cell in row:
        text = str(cell.value)
        text = text+' '
        for srow in ssheet.iter_rows(min_col=2, max_col=2):
            for scell in srow:
                sw = str(scell.value)
                pattern = re.compile(rf'\b{sw}\b')
                text = pattern.sub(' ', text)
        cell.value = text
```

Kode Program 4.6 Menghapus Stopword

Hal pertama yang harus dilakukan tentunya adalah dengan memuat daftar kata *stopword* terlebih dahulu. Setiap kata yang ada di daftar *stopwords* akan dicocokkan dengan setiap “cuitan”. Hasil dari kode di atas dapat dilihat pada tabel 4.7.

Tabel 4.7 Hasil Menghapus Stopword

| No | Sebelum | Sesudah |
|---|---|---|
| 1. | siang di kai bisa untuk intern ngga masih kuliah semester akhir fakultas manajemen trims | siang kai bisa intern kuliah semester fakultas manajemen |
| 2. | sy berniat beli tiket untk 6 orang mau tanya no kursi 6 penumpang 3 kursi kereta eko progo yang berhadap2an no 17 18 atau 18 19 tks | berniat beli tiket kursi penumpang kursi kereta eko progo |
| 3. | penasaran kalo penumpang telat kan tiket hangus nah kalo keretanya yang telat berangkat tiba ada gak kompensasinya eks lho ini | penasaran penumpang telat tiket hangus keretanya telat berangkat kompensasinya eks lho |
| 4. | min ditemukan kasus sepertinya salah tempel huruf kursi min di ka harina sama kertajaya | ditemukan salah tempel huruf kursi ka harina kertajaya |

### 4.2.4. Tokenisasi

Proses selanjutnya adalah proses tokenisasi yaitu proses memisahkan kalimat menjadi kumpulan token kata. Proses ini dapat menunjukkan kata mana saja yang paling sering muncul secara signifikan. Proses tokenisasi juga nantinya berfungsi agar kata dapat dilakukan pembobotan dan juga mengubah kata menjadi vektor. Tokenisasi dilakukan menggunakan *TweetTokenizer* milik *library* NLTK.

```
from nltk.tokenize import TweetTokenizer
token = TweetTokenizer()
datafr['Tweet'] = datafr['Tweet'].apply(token.tokenize)
```

Kode Program 4.7 Proses Tokenisasi

Dengan menggunakan NLTK dapat dengan mudah melakukan tokenisasi kata. Bahkan *library* ini juga menyediakan *tokenizer*

khusus untuk data teks Twitter, yaitu dengan *method TweetTokenizer*. Hasil dari tokenisasi dapat dilihat pada tabel 4.8 berikut.

Tabel 4.8 Hasil Tokenisasi

| No | Sebelum | Sesudah |
|---|---|---|
| 1. | siang kai bisa intern kuliah semester fakultas manajemen | [‘siang’, ‘kai’, ‘bisa’, ‘intern’, ‘kuliah’, ‘semester’, ‘fakultas’, ‘manajemen’] |
| 2. | berniat beli tiket kursi penumpang kursi kereta eko progo | [‘berniat’, ‘beli’, ‘tiket’, ‘kursi’, ‘penumpang’, ‘kursi’, ‘kereta’, ‘eko’, ‘progo’] |
| 3. | penasaran penumpang telat tiket hangus keretanya telat berangkat kompensasinya eks lho | [‘penasaran’, ‘penumpang’, ‘telat’, ‘tiket’, ‘hangus’, ‘keretanya, ‘telat’, ‘berangkat’, ‘kompensasinya’, ‘eks’, ‘lho’] |
| 4. | ditemukan salah tempel huruf kursi ka harina kertajaya | [‘ditemukan’, ‘salah’, ‘tempel’, ‘huruf’, ‘kursi’, ‘ka’, ‘harina’, ‘kertajaya’] |

### 4.2.5. Stemming

Proses *stemming* merupakan proses mengubah kata yang memiliki imbuhan menjadi kata dasar. Proses ini bertujuan agar tidak terdapat perbedaan antar kata yang seharusnya memiliki makna yang sama seperti contoh ‘menemukan’ dan ‘ditemukan’. Proses *stemming* ini menggunakan *library stemmer* yang menerapkan algortima dari Nazief dan Andriani [12].

```
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
factory = StemmerFactory()
stemmer = factory.create_stemmer()

datafr['Tweetx'] = datafr['Tweet'].apply(lambda x: [stemmer.stem(y) for y in x])
```

Kode Program 4.8 Sastrawi Stemmer

Tahapan dari algoritma *sastrawi stemmer* ini yang pertama yaitu mengecek terlebih dahulu apakah kata tersebut merupakan akar kata atau kata dasar, apabila iya maka proses berhenti pada tahap ini. Namun, jika bukan kata dasar maka akan dilakukan penghapusan imbuhan, baik itu awalan (Contoh: “meN-“, “ber-“, “di-“, “ter-“, “peN-“, “per-“, “se-“, dan “ke-“), akhiran (Contoh: “-kan”, “-an”, “-i”, dan “-nya”), kata ganti kepunyaan (Contoh: “-ku” dan “-mu”), dan yang terakhir sufiks infleksi (Contoh: “-lah”, “-kah”, “-ku”, “-mu”, atau “-nya”). Hasil dari *stemming* dapat dilihat pada tabel 4.9 berikut.

Tabel 4.9 Hasil Stemming

| No | Sebelum | Sesudah |
|---|---|---|
| 1. | [‘siang’, ‘kai’, ‘bisa’, ‘intern’, ‘kuliah’, ‘semester’, ‘fakultas’, ‘manajemen’] | [‘siang’, ‘kai’, ‘bisa’, ‘intern’, ‘kuliah’, ‘semester’, ‘fakultas’, ‘manajemen’] |
| 2. | [‘berniat’, ‘beli’, ‘tiket’, ‘kursi’, ‘penumpang’, ‘kursi’, ‘kereta’, ‘eko’, ‘progo’] | [‘niat’, ‘beli’, ‘tiket’, ‘kursi’, ‘tumpang’, ‘kursi’, ‘kereta’, ‘eko’, ‘progo’] |
| 3. | [‘penasaran’, ‘penumpang’, ‘telat’, ‘tiket’, ‘hangus’, ‘keretanya, ‘telat’, ‘berangkat’, ‘kompensasinya’, ‘eks’, ‘lho’] | [‘penasaran’, ‘tumpang’, ‘telat’, ‘tiket’, ‘hangus’, ‘kereta, ‘telat’, ‘berangkat’, ‘kompensasi’, ‘eks’, ‘lho’] |
| 4. | [‘ditemukan’, ‘salah’, ‘tempel’, ‘huruf’, | [‘temu’, ‘salah’, ‘tempel’, ‘huruf’, ‘kursi’, ‘ka’, ‘harina’, ‘kertajaya’] |

| | | |
|---|---|---|
| | ‘kursi’, ‘ka’, ‘harina’, ‘kertajaya’] | |

### 4.3. *Weighted Word Embedding*

Penyisipan kata (*word embedding*) adalah semacam representasi kata yang diperoleh dengan melatih sejumlah besar korpus yang tidak berlabel (*unsupervised*). *Word embedding* dapat dilihat sebagai representasi spasial dari sebuah kata. Namun, itu tidak dapat secara eksplisit mendefinisikan tingkat penting dan tidaknya dari setiap kata dalam teks.

Dalam melakukan klasifikasi teks, representasi teks merupakan langkah yang penting. Penggunaan model jaringan saraf (*neural network*) sudah menjadi hal yang umum dilakukan beberapa tahun ke belakang karena *word embedding* yang dipelajari dari data tidak terstruktur yang masif **[13]**. Fungsi komposisi digunakan dalam model saraf untuk mewakili teks sebagai vektor berdimensi kecil. Ini adalah proses matematis untuk menggabungkan beberapa *word embeddings* menjadi satu vector.

Model ini yang sering disebut sebagai *weighted word embedding*. Intuisi dari model ini adalah bahwa tidak semua kata dalam kalimat atau dokumen sama pentingnya untuk klasifikasi teks. Dengan demikian, model ini memperkenalkan skema bobot *term* untuk menginisialisasi nilai bobot kata **[14]**. Semakin penting kata, semakin tinggi nilai bobotnya. Model ini telah menunjukkan hasil yang memuaskan dalam arsitektur klasifikasi teks, seperti *Recurrent Neural Networks* (RNNs), *Recursive Neural Networks*, dan *Convolutional Neural Networks* (CNNs) **[15]**. Dengan kata lain, skema pembobotan istilah dapat secara eksplisit mengukur tingkat penting dari setiap kata.

#### 4.3.1. Pembobotan Kata

*Term weighting* (pembobotan kata) adalah prosedur yang terjadi selama proses pengindeksan teks untuk menilai nilai setiap istilah untuk dokumen. *Term weighting* adalah pemberian nilai numerik pada istilah yang mewakili kepentingannya dalam

dokumen untuk meningkatkan efektivitas pengambilan [16]. Dengan pembobotan ini merupakan cara yang memungkinkan untuk menentukan pentingnya suatu istilah (*term*) yang diberikan dalam suatu dokumen.

Pembobotan kata yang dilakukan pada pengerjaan tugas akhir ini menggunakan TF-IDF. TF-IDF (*Term Frequency, Inverse Document Frequency*) adalah adalah bobot yang sering digunakan dalam pencarian informasi dan penggalian teks. Bobot ini adalah ukuran statistik yang digunakan untuk mengevaluasi seberapa penting suatu kata bagi sebuah dokumen. Tingkat kepentingan suatu kata meningkat secara proporsional dengan berapa kali sebuah kata muncul dalam dokumen tetapi diimbangi dengan frekuensi kata dalam korpus.

Biasanya, bobot TF-IDF terdiri dari gabungan dua komputasi: yang pertama menghitung *Term Frequency* (TF) yang dinormalisasi, alias berapa kali sebuah kata muncul dalam dokumen, dibagi dengan jumlah total kata dalam dokumen itu. Komputasi yang kedua adalah *Inverse Document Frequency* (IDF), dihitung sebagai logaritma dari jumlah dokumen dalam korpus dibagi dengan jumlah dokumen tempat istilah tertentu muncul.

Sebagai contoh, pada satu cuitan yang masuk terdapat satu kata "Terlambat" dari panjang cuitan sebanyak 10 kata. Sehingga nilai TF dari kata terlambat adalah 1/10.

$$TF_{Terlambat} = 0{,}1$$

Lalu semisal dari total cuitan yang masuk yaitu 20.000. Kata "Terlambat" muncul pada 5.000 cuitan. Maka untuk IDF dari kata terlambat dihitung seperti berikut,

$$IDF_{Terlambat} = \log\left(\frac{20.000}{5.000}\right) = 0{,}602$$

Maka setelah diketahui masing-masing nilai TF dan IDF dari kata “Terlambat”. Hal yang selanjutnya dilakukan adalah mengalikan dua hasil tersebut menjadi seperti berikut,

$$TF - IDF_{Terlambat} = 0{,}1 \times 0{,}602 = 0{,}0602$$

Bobot yang didapatkan dari kata terlambat tersebut adalah 0,0602.

Terdapat cuitan yang masuk seperti ini, “info pemesanan tiket lokal pembelian tiket prameks go show menit sebelum berangkat layani surabaya turun solo”. Untuk mengubah dari siatu cuitan menjadi vektor maka harus diubah terlebih dahulu kedalam bentuk token kata seperti ini, [‘info’, ‘pemesanan’, 'tiket', 'lokal', 'pembelian', 'tiket', 'prameks', 'go', 'show', 'menit', 'sebelum', 'berangkat', 'layani', 'surabaya', 'turun', 'solo']. Dalam cuitan tersebut terdapat 16 kata. Semisal nilai TF dari kata info adalah 1/16 karena kemunculannya hanya 1 kali. Sedangkan dari 20.000 cuitan yang masuk terdapat 12.000 kata info, maka nilai IDF nya adalah 0.221.

Tabel 4.10 Hasil TF-IDF

| Kata | TF | IDF | TF-IDF |
|---|---|---|---|
| Info | 1/16 | 0.221 | 0.0138 |
| Pemesanan | 1/16 | 0.218 | 0.0136 |
| Tiket | 2/16 | 0.221 | 0.0138 |
| Lokal | 1/16 | 0.218 | 0.0136 |
| Pembelian | 1/16 | 0.016 | 0.001 |
| Prameks | 1/16 | 0.010 | 0.0062 |
| Go | 1/16 | 0.205 | 0.128 |
| Show | 1/16 | 0.107 | 0.0066 |
| Menit | 1/16 | 0.189 | 0.0118 |
| Sebelum | 1/16 | 0.264 | 0.0165 |
| Berangkat | 1/16 | 0.379 | 0.0237 |
| Layani | 1/16 | 0.254 | 0.0158 |
| Surabaya | 1/16 | 0.134 | 0.0083 |
| Turun | 1/16 | 0.100 | 0.0062 |

Tabel lengkap hasil TF-IDF yang dapat dilihat pada tabel 4.10. Dari hasil TF-IDF tersebut nantinya akan disimpan untuk selanjutnya dikalikan dengan hasil dari vektorisasi kata.

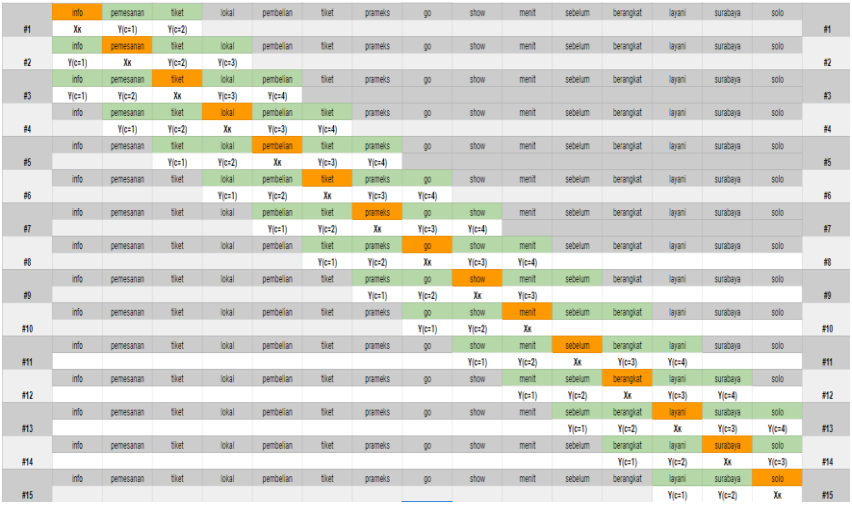
### 4.3.2. Vektorisasi Kata

Dalam melakukan pengolahan data teks. Sebelum kita dapat memasukkan data teks tersebut sebagai *input* untuk model yang kita buat, data teks tersebut harus diubah terlebih dahulu ke dalam suatu vektor dengan panjang yang ditentukan. Proses inilah yang disebut dengan *word embedding*. Pada proses vektorisasi ini menggunakan Word2Vec dan FastText milik *library* Gensim.

Vektorisasi atau *word embedding* menggunakan Word2Vec dan FastText ini merupakan suatu *neural network* yang masih dangkal tidak terlalu dalam karena hanya terdapat dua lapisan dan tanpa lapisan tersembunya (*hidden layer*). Proses *word embedding* ini seperti pembelajaran mesin yang masih sederhana. Word2Vec dan FastText melatih kata untuk memprediksi kata berdasarkan kata yang berada diantara kata tersebut (konteks). Proses kedua model mengubah kata menjadi suatu vektor adalah dengan menyandingkan fokus dari kata dengan konteks kata. Apabila fokus kata "Terlambat" dan kata "Telat" terdapat pada konteks kata yang bermiripan maka kedua kata ini akan memiliki nilai vektor yang yang hampir mendekati 1. Kedua model ini mentransformasikan kata menjadi vektor dan mencari kedekatan antar kata yang ada disekitarnya. Dengan parameter *windows size* akan menetukan berapa kata tetangga yang akan dicari kedekatannya. Kata yang berada di antara kata target itu disebut sebagai konteks.

Dari suatu cuitan, "info pemesanan tiket lokal pembelian tiket prameks go show menit sebelum berangkat layani surabaya turun solo". Untuk mengubah kata menjadi vektor, ditentukan *window size* sebesar 2. Maka tiap kata yang ada dalam cuitan

tersebut akan menjadi target dan konteks. Semisal target kata “info”, maka konteks katanya adalah “pemesanan” dan “tiket”. Lalu target kata “pemesanan” maka konteksnya adalah “info”, “tiket” dan “lokal”. Begitu terus hingga kata terakhir seperti pada gambar 4.1.




Gambar 4.1 Sliding Window

Dari *sliding window* tersebut tiap *window* dari 1-16 akan dilakukan *one-hot encoding*. Dari hasil *one-hot encoding* tersebut (input) nantinya akan dikalikan dengan *weight* (w1) yang sudah dirandomisasi . Dari hasil perkalian tersebut akan dihasilkan *hidden state*. Selanjutnya *hidden state* tersebut dikalikan dengan *weight* yang sudah dirandomisasi juga (w2). Setelah dikalikan dengan bobot yang kedua, maka akan dihasilkan output vektor dari tiap kata dalam cuitan tersebut.

### 4.3.3. Perancangan *Weighted Word Embedding*

Berdasarkan dari artikel yang menjadi rujukan dalam pembuatan model *weighted word embedding* [8]. *Pseudocode* dari model *weighted word embedding* dapat dilihat pada kode program 4.9.

**Input:** a training set $\mathcal{C}$ of $n$ classes.
**Output:** weighted word embeddings $\mathbf{I}_1, \ldots, \mathbf{I}_n$.
1: Use $\mathcal{C}$ to create a vocabulary $\mathcal{V}$ of words.
2: Cluster the documents in $\mathcal{C}$ with the same label as $\{\mathcal{C}_1, \ldots, \mathcal{C}_n\}$.
3: **for** each $i \in \{1, \ldots, n\}$ **do**
4: Predefine a weight vector $\mathbf{w}_i \in \mathbb{R}^{|\mathcal{V}|}$;
5: Obtain a TF-IDF term-document matrix $\mathcal{M}_i$ with $\mathcal{C}_i$;
6: For the words occurring in $\mathcal{C}_i$, assign the maximum value of each column of $\mathcal{M}_i$ to corresponding entry of $\mathbf{w}_i$;
7: For the words not occurring in $\mathcal{C}_i$, assign the minimum value of all maximum values to the rest entries of $\mathbf{w}_i$;
8: **end for**
9: For a sentence $\mathbf{s}$ in $\mathcal{C}$, pad or truncate it to length $k$;
10: Get the word embedding matrix $\mathbf{E} \in \mathbb{R}^{k \times l}$ of $\mathbf{s}$;
11: **for** each $i \in \{1, \ldots, n\}$ **do**
12: **for** each $j \in \{1, \ldots, k\}$ **do**
13: For the $j$th word in $\mathbf{s}$, find its corresponding weight $\hat{w}_{ij}$ from $\mathbf{w}_i$;
14: **end for**
15: Denote $\hat{\mathbf{w}}_i = (\hat{w}_{i1}, \ldots, \hat{w}_{ik})^{\mathrm{T}}$, i.e., a weight vector computed from the collection $\mathcal{C}_i$;
16: Let $\mathbf{W}_i$ be a $k \times l$ matrix whose each column is $\hat{\mathbf{w}}_i$;
17: Let $\mathbf{I}_i = \mathbf{W}_i \odot \mathbf{E}$, where $\odot$ denotes the element-wise multiplication of two matrices.
18: **end for**

Kode Program 4.9 *Pseudocode WWE*

Seluruh proses dapat secara luas dibagi menjadi dua fase, yaitu, perhitungan bobot istilah dan pembuatan model *weighted word embeddings*.

Diberikan korpus *C* yang terdiri dari n kelas, pertama-tama mengubah *tokenize* menjadi istilah yang merujuk pada kata-kata. Berdasarkan istilah yang diperoleh, algoritma *weighted word embedding* membuat kosakata *V* di mana setiap istilah unik. Untuk menyederhanakan notasi, *C* diwakili sebagai

koleksi $C = \{C1,. . . , Cn\}$, di mana *Ci* memasukkan semua dokumen dengan label i yang sama di dalam korpus. Untuk setiap koleksi *Ci*, TF-IDF *term-document matrix* Mi $\in R^{|Di|x|Vi|}$ dapat dihitung sesuai dengan kode program diatas Perhatikan bahwa | *Di* | menunjukkan jumlah dokumen dalam *Ci* dan | *Vi* | adalah jumlah istilah unik dalam *Ci*, i = 1,. . . , n. Selanjutnya, kita dapat menghasilkan vektor bobot dengan $w_i \in R^{|V|}$. Sehubungan dengan ketentuan dalam *Ci* dan *V*, bobot ditetapkan sebagai nilai maksimum dari setiap kolom Mi. Adapun istilah tidak dalam *Ci* tetapi dalam *V*, biarkan bobotnya menjadi nilai minimum dari nilai maksimum yang telah dihitung dengan Mi. Dengan mengulangi proses di atas koleksi *C1*,. . . , *Cn*, n vektor berat *w1*,. . . , *wn* bisa didapat. *Weighted word embedding* dapat dihasilkan dari perkalian antara matriks ***W***, hasil atau vektor dari pembobotan tiap kata dalam satu dokumen, dan matriks ***E***, matriks transpose yang berisikan word embedding yang didapat secara acak ataupun menggunakan pre-trained vectors yang dibuat oleh Word2Vec dan FastText.

*“Halaman ini sengaja dikosongkan”*

# BAB 5
# HASIL DAN PEMBAHASAN

Pada bab ini akan dijelaskan mengenai hasil eksekusi dari model *weighted word embedding* yang telah dijelaskan pada bab sebelumnya dan juga model klasifikasi *recurrent neural network*. Bagian ini akan juga akan membahas hasil yang didapat dari model yang dibuat.

## 5.1. Lingkungan Implementasi

Pengerjaan tugas akhir ini diimplementasi menggunakan laptop dan juga perangkat lunak yang mendukung penyelesaian tugas akhir. Spesifikasi laptop yang digunakan pada tugas akhir dapat dilihat pada tabel 5.1.

Tabel 5.1 Spesifikasi Laptop

| | |
|---|---|
| **Prosesor** | Intel i7-6500U |
| **Memori** | 8 GB DDR4 |
| **GPU** | NVIDIA GTX 950M |
| **Sistem Operasi** | Windows 10 |
| **Arsitektur** | 64-bit |

Sedangkan untuk perangkat lunak atau *tools* yang digunakan untuk mengolah data dapat dilihat pada tabel 5.2.

Tabel 5.2 Perangkat Lunak

| | |
|---|---|
| **Bahasa Pemrograman** | Python 3.7 |
| ***Distribution*** | Anaconda 3 |
| **IDE** | Spyder |

| | |
|---|---|
| ***Library*** | Pandas, Numpy, Regular Expression, NLTK, Gensim, SKLearn, Tesnsorflow |

## 5.2. Informasi Data

Data pengaduan yang masuk pada akun media sosial Twitter ini memiliki panjang 20.190 baris. Setelah melalui fase praproses pada pembahasan sebelumnya, yaitu Twitter *text cleaning*, menghapus data *null*, menghapus *stopword, stemming*, dan proses pembersihan data lainnya. Data bersih yang sudah di praproses memiliki panjang 19.447 baris. Data memiliki tiga buah label kategori yaitu informasi, keluhan dan saran/permintaan. Jumlah tiap label kategori dapat dilihat pada tabel 5.3.

Tabel 5.3 Kategori Pengaduan

| **Kategori** | **Jumlah** |
|---|---|
| Informasi | 15.745 |
| Keluhan | 3.140 |
| Saran/Permintaan | 562 |

### 5.2.1. Split Data

Setelah semua data selesai di praproses dan sudah bersih siap untuk di proses. Hal yang harus dilakukan pertama adalah membagi data menjadi dua, yaitu data *training* (latih) dan data *test* (uji). Perbandingan data latih dan uji adalah 80:20. Data latih mengambil 80% dari data awal, sedangkan data uji mengambil sisanya yaitu 20%. Jumlah data latih terdapat 15.558 baris sedangkan data uji sebanyak 3.889 baris. Data diambil secara acak dan tidak runut dari *dataset* awal. Jumlah label pada data latih dan data uij dapat dilihat pada tabel 5.4 dan tabel 5.5

Tabel 5.4 Kategori Pengaduan Data Latih

| **Kategori** | **Jumlah** |
|---|---|
| Informasi | 12.599 |
| Keluhan | 2.521 |
| Saran/Permintaan | 438 |

Tabel 5.5 Kategori Pengaduan Data Uji

| **Kategori** | **Jumlah** |
|---|---|
| Informasi | 3.146 |
| Keluhan | 619 |
| Saran/Permintaan | 124 |

Data latih dan uji yang sudah bersih akan siap untuk di proses *embedding* menggunakan *weighted word embedding.*

### 5.2.2. Hasil Praproses Data

Data mentah yang didapat dari Twitter harus dilakukan praproses terlebih dahulu. Hal ini bertujuan agar *dataset* lebih bersih dan juga teratur hingga nantinya dapat dilakukan klasifikasi dan memberikan akurasi yang bagus. Data Twitter yang sudah bersih dapat dilihat pada tabel 5.6.

Tabel 5.6 Hasil Praproses Data

| **Teks Sebelum** | **Teks Sesudah** |
|---|---|
| Siang, di KAI bisa untuk intern ngga? masih kuliah semester akhir fakultas manajemen. Trims | [‘siang’, ‘kai’, ‘bisa’, ‘intern’, ‘kuliah’, ‘semester’, ‘fakultas’, ‘manajemen’] |

| | |
|---|---|
| @KAI121 sy berniat beli tiket untk 6 org, mau tanya no kursi 6 penumpang (3 kursi) kereta eko progo yg berhadap2an no 17-18 atau 18-19? Tks | [‘niat’, ‘beli’, ‘tiket’, ‘kursi’, ‘tumpang’, ‘kursi’, ‘kereta’, ‘eko’, ‘progo’] |
| @KAI121 Penasaran. Kalo penumpang telat kan tiket hangus. Nah kalo keretanya yg telat berangkat/tiba ada gak kompensasinya? Eks. lho ini. 2:30 PM - 20 Dec 2015 · | [‘penasaran’, ‘tumpang’, ‘telat’, ‘tiket’, ‘hangus’, ‘kereta, ‘telat’, ‘berangkat’, ‘kompensasi’, ‘eks’, ‘lho’] |
| Min @KAI121 ditemukan kasus sepertinya salah tempel huruf kursi min. Di KA Harina sama Kertajaya <URL> 7:04 PM - 15 Sep 2015 · | [‘temu’, ‘salah’, ‘tempel’, ‘huruf’, ‘kursi’, ‘ka’, ‘harina’, ‘kertajaya’] |

Tahap praproses data yang dilakukan yaitu *casefolding* (mengubah huruf kapital menjadi kecil), menghapus data *null*, tanda baca, angka, *mention*, URL, tanggal, waktu, *stopword*, dan yang terakhir melakukan *stemming*.

### 5.3 Model *Word Embedding*

Pada *word embedding*, seperti yang ssudah dijelaskan pada pembahasan sebelumnya menggunakan *weighted word embedding*. *Weighted Word Embedding*. Data latih dan uji yang sudah bersih dilakukan *embedding* menggunakan model *weighted word embedding*. Hal ini bertujuan agar model dapat berjalan dengan efektif. Model ini merupakan model word embedding berbasis Word2Vec dan FastText. Dan juga TF-IDF *Library* yang digunakan dalam membuat model ini yaitu TfidfVectorizer dan gensim.

### 5.3.1. Pembuatan Model *Weighted Word Embedding* Word2Vec

```
data_vectors = []
row=0

for sent in datafr['Tweetx']:
    sent_vec = np.zeros(50)
    weight_sum = 0
    for word in sent:
        if word in w2v_words and word in tfidf_feat:
            vec = w2v_m.wv[word]
            tf_idf = dictionary[word]*(sent.count(word)/len(sent))
            sent_vec += (vec * tf_idf)
            weight_sum += tf_idf
    if weight_sum != 0:
        sent_vec /= weight_sum
    data_vectors.append(sent_vec)
    row += 1
```

Kode Program 5.1 Kode *Weighted Word Embedding*

Mengikuti dari penjelasan artikel jurnal yang menjadi rujukan [8] model *weighted word embedding* ini merupakan gabungan antara model Word2Vec dan juga TF-IDF. Sebelum masuk ke dalam perulangan *for* seperti di atas. Hal yang pertama dilakukan yaitu melakukan inisiasi model dari TF-IDF dan juga Word2Vec. Selanjutnya kita membuat kamus kata yang berisikan *list* kata yang terdapat pada data. Kata dalam kamus kata selanjutnya dipanggil pada perulangan. Pertama akan dilakukan penghitungan nilai TF-IDF dari setiap kata. Lalu mengalikan nilai TF-IDF dengan kata yang sesuai dan dibagi dengan jumlah nilai TF-IDF. Selanjutnya mengalikan hasil TF-IDF dengan hasil dari Word2Vec. Hasil dari *word embedding* ini adalah berupa *numpy array* yang memiliki panjang dimensi 50 karena sudah diatur demikian pada saat inisialisasi.isi

### 5.3.2. Hasil Model *Weighted World Embedding* Word2Vec

Hasil dari model *weighted embedding* ini merupakan berupa *array* dengan 50 dimensi. Berikut contoh hasil dari *embedding* data latih dapat dilihat pada tabel 5.7.

Tabel 5.7 Hasil Weighted Word Embedding

| **Index** | 0 | 1 | 2 | ... | 48 | 49 |
|---|---|---|---|---|---|---|
| 0 | -0.367121 | -0.267898 | 0.248966 | ... | 0.154694 | 0.225441 |
| 1 | -0.172453 | -0.437697 | 0.248209 | ... | -0.142336 | 0.501239 |
| 2 | -0.159872 | -0.372715 | -0.045302 1 | ... | 0.043996 6 | 0.348656 |
| ..... ... | ..... | ..... | ..... | ... | ..... | ..... |
| 15557 | -0.430925 | -0.319547 | 0.233689 | ... | 0.179252 | 0.247613 |
| 15558 | -0.167254 | -0.539361 | 0.331606 | ... | 0.030482 5 | 0.511976 |

Hasil *embedding* diatas nantinya perlu dilakukan perubahan dimensi terlebih dahulu sebelum masuk ke dalam model klasifikasi.

### 5.3.3. *Model Weighted Word Embedding* FastText

Model ini sama halnya dengan model *weighted word embedding* hanya saja disini menggunakan FastText dalam mengubah kata menjadi vektor. FastText ini merupakan model yang lebih kompleks dibandingkan dengan model Word2Vec.

FastText membagi kata ke dalam *n-gram*. Sehingga model ini dapat memberikan performa yang lebih bagus karena memproses kata hingga ke tingkat karakter. Tidak ada yang berbeda pada kode program model ini, *library* yang digunakan pun masih sama yaitu Gensim.

```
fast_vectors = []
row=0

for sent in datafr['Tweetx']:
    fast_vec = np.zeros(100)
    weight_sum = 0
    for word in sent:
        if word in fastm_word and word in tfidf_feat:
            f_vec = fast_m.wv[word]
            fast_tf_idf = dictionary[word]*(sent.count(word)/len(sent))
            fast_vec += (f_vec * fast_tf_idf)
            weight_sum += fast_tf_idf
    if weight_sum != 0:
        fast_vec /= weight_sum
    fast_vectors.append(fast_vec)
    row += 1
```

Kode Program 5.2 Model WWE FastText

Sama dengan model yang sebelumnya, pada model ini menggabungkan antara hasil dari FastText dan TF-IDF. Masing-masing hasil nantinya akan dikali dan selanjutnya disimpan dalam `fast_vectors`.

### 5.3.4. Model Word2Vec

Model *word embedding* ini hanya menggunakan Word2Vec saja tanpa ada pembobotan menggunakan TF-IDF. Sehingga model ini lebih sederhana. Model ini dibuat sebagai perbandingan performa antara penggunaan bobot dan tanpa bobot. Model ini nantinya akan menjadi pembanding. Seharusnya model ini akan memberikan hasil performa yang lebih buruk dibandingkan dua model yang sebelumnya.

```
data_w2v = []
row=0


for sent in datafr['Tweetx']:
    sent_w2v = np.zeros(50)
    for word in sent:
        if word in w2v_words:
            sent_w2v = w2v_m.wv[word]

    data_w2v.append(sent_w2v)
    row += 1

data_wv = pd.DataFrame(data_w2v)
```

Kode Program 5.3 Model Word2Vec

*Word embedding* menggunakan Word2Vec ini hanya perlu memanggil *library* Gensim dan membuat kamus kata yang nantinya akan dipanggil pada perulangan seperti kode program diatas.

### 5.4. Model Klasifikasi RNN

Setelah data dilakukan *embedding* yang selanjutnya dilakukan adalah melakukan proses klasifikasi dengan model RNN (*Recurrent Neural Network*). Pada RNN ini akan terdapat beberapa *layer*. Sebelum memasukkan data ke dalam model klasifikasi ada beberapa tahap yang dilakukan terlebih dahulu yaitu melakukan *reshape input data* agar cocok dengan input model klasifikasi dan juga melakukan *encoding* pada label kategori.

Pada tugas akhir ini penulis menggunakan *library* Tensorflow dan API Keras dalam membuat dan melakukan model klasifikasi menggunakan RNN.

#### 5.4.1. Menyiapkan Data Input

Data hasil *embedding* yang sudah dilakukan sebelumnya berisikan *array* dengan 50 dimensi. Data tersebut tidak akan

cocok untuk masuk ke model klasifikasi. Oleh karena itu kita mengubah dimensi dari data dengan menggunakan metode *reshape*.

Selain mengubah dimensi data hal lain yang perlu dilakukan ialah *encoding* atau mengubah label yang awalnya masih berbentuk *string* diubah menjadi *array*. *Encoding* dilakukan menggunakan LabelEncoder. Berikut Label beserta hasil *encode* pada tabel 5.8.

Tabel 5.8 Hasil Label Encoding

| **Label** | **Hasil *Encode*** |
|---|---|
| Informasi | 0 |
| Keluhan | 1 |
| Saran/Permintaan | 2 |

Setelah label di-*encoding* maka didapatkan hasil seperti di atas. Label informasi diganti menjadi ‘0’, Keluhan menjadi ‘1’, dan Saran/Permintaan menjadi ‘2’.

### 5.4.1. Skenario Model Klasifikasi

Pada pengerjaan tugas akhir ini akan membandingkan beberapa skenario model untuk mencari mana hasil yang terbaik. Pada pembuatan skenario ini ada beberapa parameter yang akan diubah-ubah, yaitu *size* dan *length* dari vektor, *activation* pada masing-masing *layer* RNN, dan yang terakhir yaitu model *word embedding*.

Terdapat tiga model *word embedding* yang akan dicoba pada tugas akhir ini yaitu *Weighted Word Embedding* menggunakan Word2Vec (WWE-W2V), *Weighted Word Embedding* menggunakan FastText (WWE-FT), dan model *word embedding* menggunakan Word2Vec saja. Sedangkan untuk

*size* dan *length* dari vektor hasil *word embedding* akan dibandingkan tiga hasil yaitu dengan panjang 50, 100, dan 200. Selain skenario dari *word embedding*, penulis juga membandingkan parameter *activation* yang ada pada tiap *layer* RNN. Arsitektur RNN yang digunakan adalah arsitektur LSTM (Long-Short Term Memory). Pada skenario fungsi aktivasi ini akan mencoba dua jenis fungsi aktivasi yaitu *relu* dan *softmax.* Pada masing-masing *layer* akan dilakukan skenario pemberian fungsi aktivasi *relu* dan aktivasi *softmax*. Detail skenario dapat dilihat pada tabel 5.9.

Tabel 5.9 Skenario Model Klasifikasi

| | **Model *Word Embedding*** | **Panjang Vektor** | **Fungsi Aktivasi** |
|---|---|---|---|
| **Skenario 1** | WWE-W2V | 50 | Aktivasi *softmax* pada tiap layer |
| | | 100 | |
| | | 200 | |
| **Skenario 2** | WWE-FT | 50 | |
| | | 100 | |
| | | 200 | |
| **Skenario 3** | Word2Vec | 50 | |
| | | 100 | |
| | | 200 | |
| **Skenario 4** | WWE-W2V | 50 | Aktivasi *relu* pada tiap layar dan aktivasi *softmax* pada *output layer* |
| | | 100 | |
| | | 200 | |
| **Skenario 5** | WWE-FT | 50 | |
| | | 100 | |
| | | 200 | |
| **Skenario 6** | Word2Vec | 50 | |
| | | 100 | |
| | | 200 | |

Terdapat enam skenario yang nantinya akan dibandingkan hasilnya. Sebelum membandingkan antar skenario, hal pertama yang dilakukan adalah membandingkan masing-masing

percobaan yang terdapat di dalam skenario. Lalu membandingkan mana yang terbaik antara skenario 1 hingga skenario 6.

### 5.4.2. Membuat Model RNN Awal

Model Klasifikasi RNN ini menggunakan Keras yang ada pada Tensorflow. Karena pada tahap sebelumnya sudah dilakukan *embedding* sehingga tidak perlu lagi menambahkan *layer embedding*. Oleh karena itu kita hanya perlu menambahkan parameter *input_shape* untuk memberitahu dimensi yang akan masuk pada model.

```
model = tf.keras.Sequential([
    tf.keras.layers.LSTM(50, input_shape=(1,50), activation='relu'),
    tf.keras.layers.Dense(50, activation='relu'),
    tf.keras.layers.Dense(3, activation='softmax'),
])
```

Kode Program 5.4 Model RNN Awal

Model dibangun menggunakan tf.keras.sequential. Model tidak membutuhkan model *embedding* sebagai gantinya pada *input layer* perlu dimasukkan *input_shape()*. Hal ini karena *embedding* sudah dilakukan menggunakan model *weighted word embedding*. RNN memproses input *sequence* dengan mengulangi melalui elemen. RNN meneruskan output dari satu cap waktu ke *input* mereka dan kemudian ke yang berikutnya.

```
model.compile(loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
              optimizer='adam',
              metrics=['accuracy'])
```

Kode Program 5.5 *Model Compiler*

Untuk mengukur *loss* antara label dan prediksi menggunakan SparseCategoricalCrossentropy. Penggunaan *cross entropy* tipe ini karena terdapat lebih dari dua kelas label dan berbentuk *integer*. Untuk optimasi menggunakan adam karena optimasi adam dikenal cukup efektif dan kerap dipakai pada model jaringan syaraf karena memberikan hasil yang bagus dan waktu eksekusi yang cepat.

## 5.5. Hasil Percobaan Skenario

Pada tugas akhir ini ingin menunjukkan bahwa penggunaan model *weighted word embedding* menghasilkan performa klasifikasi yang lebih baik dibanding *word embedding* tanpa pembobotan. Kami membandingkan antara Skenario 1, 2, 3, 4, 5, dan 6. yang sudah ditentukan pada subbab sebelumnya. Parameter yang pembedanya adalah model *word embedding*, panjang vektor hasil *word embedding*, dan fungsi aktivasi pada *layer* RNN.

### 5.5.1. Skenario 1

Pada skenario 1 akan melakukan perbandingan antara model *word embedding* dengan *Weighted Word Embedding* menggunakan Word2Vec (WWE-W2V) yang memiliki panjang 50, 100, dan 200. Pada skenario 1 ini menggunakan fungsi aktivasi *softmax* pada seluruh lapisannya kecuali lapisan ke-2.

Tabel 5.10 Parameter Skenario 1

| | **Parameter 1** | **Parameter 2** | **Parameter 3** |
|---|---|---|---|
| ***Word Embedding*** | WWE-W2V | WWE-W2V | WWE-W2V |
| ***Fold*** | 10 | 10 | 10 |
| ***Vector Length*** | 50 | 100 | 200 |
| ***Layer 1*** | Softmax | Softmax | Softmax |
| ***Layer 2*** | relu | relu | relu |
| ***Layer 3*** | Softmax | Softmax | Softmax |
| ***Epoch*** | 10 | 10 | 10 |

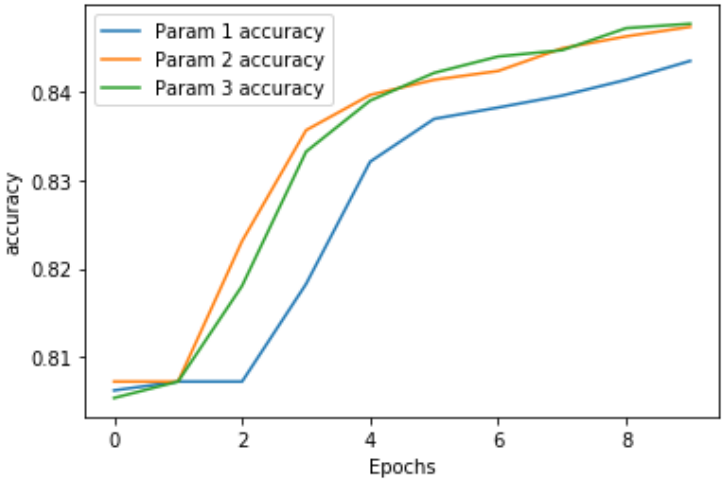
Berikut hasil akurasi model dari pelatihan skenario 1 yang dapat dilihat pada tabel 5.11

Tabel 5.11 Hasil Akurasi Skenario 1

| | **Parameter 1** | **Parameter 2** | **Parameter 3** |
|---|---|---|---|
| **1** | 0.8062 | 0.8072 | 0.8053 |

| | | | |
|---|---|---|---|
| **2** | 0.8072 | 0.8072 | 0.8072 |
| **3** | 0.8072 | 0.8231 | 0.8180 |
| **4** | 0.8183 | 0.8357 | 0.8333 |
| **5** | 0.8321 | 0.8397 | 0.8390 |
| **6** | 0.8370 | 0.8414 | 0.8422 |
| **7** | 0.8382 | 0.8424 | 0.8440 |
| **8** | 0.8396 | 0.8450 | 0.8447 |
| **9** | 0.8414 | 0.8463 | 0.8472 |
| **10** | 0.8435 | 0.8474 | 0.8477 |

Dari Tabel 5.11 bisa dilihat bahwa tidak ada peningkatan akurasi yang signifikan antara data dengan panjang vektor 50, 100, dan 200. Terjadi perbedaan kenaikan pada tiap *epoch* pada masing-masing parameter namun perbedaan tersebut tidak terlalu signifikan. Hasil akurasi tertinggi didapat pada *epoch* 10 menggunakan pada model dengan Parameter 3 yaitu dengan panjang vektor 200, hasil akurasi 0,8477. Kenaikan akurasi dari tiap *epoch* dapat dilihat pada gambar 1.




Gambar 5.1 Plot Grafik Akurasi Skenario 1

### 5.5.2. Skenario 2

Pada skenario 2 akan melakukan perbandingan antara model *word embedding* dengan *Weighted Word Embedding* menggunakan FastText (WWE-FT) yang memiliki panjang 50, 100, dan 200. Pada skenario 1 ini menggunakan fungsi aktivasi *softmax* pada seluruh lapisannya kecuali pada lapisan ke-2.

Tabel 5.12 Parameter Skenario 2

| | **Parameter 4** | **Parameter 5** | **Parameter 6** |
|---|---|---|---|
| ***Word Embedding*** | WWE-FT | WWE-FT | WWE-FT |
| ***Fold*** | 10 | 10 | 10 |
| ***Vector Length*** | 50 | 100 | 200 |
| ***Layer 1*** | Softmax | Softmax | Softmax |
| ***Layer 2*** | Relu | Relu | Relu |
| ***Layer 3*** | Softmax | Softmax | Softmax |
| ***Epoch*** | 10 | 10 | 10 |

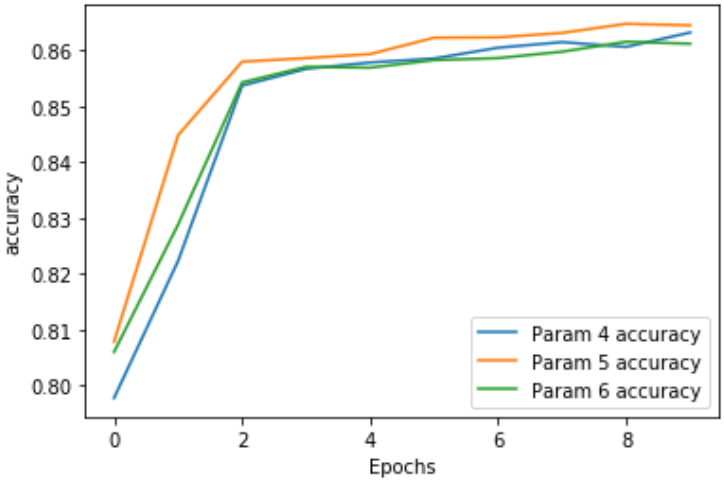
Berikut hasil akurasi model dari pelatihan model skenario 2 yang dapat dilihat pada tabel 5.13

Tabel 5.13 Hasil Akurasi Skenario 2

| | **Parameter 4** | **Parameter 5** | **Parameter 6** |
|---|---|---|---|
| **1** | 0.7978 | 0.8079 | 0.8060 |
| **2** | 0.8223 | 0.8448 | 0.8288 |
| **3** | 0.8537 | 0.8579 | 0.8542 |
| **4** | 0.8567 | 0.8586 | 0.8570 |
| **5** | 0.8578 | 0.8593 | 0.8569 |
| **6** | 0.8585 | 0.8622 | 0.8582 |
| **7** | 0.8604 | 0.8623 | 0.8586 |
| **8** | 0.8614 | 0.8631 | 0.8597 |
| **9** | 0.8606 | 0.8463 | 0.8615 |
| **10** | 0.8632 | 0.8644 | 0.8612 |

Dari Tabel 5.13 dapat dilihat bahwa performa akurasi model tidak memiliki perbedaan yang signifikan antara panjang 50,

100, dan 200. Masih sama seperti skenario 1 hasil yang didapat masih belum maksimal. Namun dengan penggunaan model *Weighted Word Embedding* dengan FastText memberikan hasil akurasi yang sedikit lebih baik yaitu bertambah 0,167. Berbeda dengan Skenario 1, pada Skenario 2 hasil akurasi yang lebih baik dilakukan oleh model Parameter 4, yaitu vektor dengan panjang 50. Kenaikan skor akurasi dapat dilihat pada gambar 2




Gambar 5.2 Plot Grafik Akurasi Skenario 2

### 5.5.3. Skenario 3

Pada skenario 3 akan melakukan perbandingan antara model *word embedding* dengan menggunakan Word2Vec yang memiliki panjang 50, 100, dan 200. Pada skenario 1 ini menggunakan fungsi aktivasi *softmax* pada seluruh lapisannya kecuali pada lapisan ke-2.

Tabel 5.14 Parameter Skenario 3

| | **Parameter 7** | **Parameter 8** | **Parameter 9** |
|---|---|---|---|
| ***Word Embedding*** | Word2Vec | Word2Vec | Word2Vec |
| ***Fold*** | 10 | 10 | 10 |

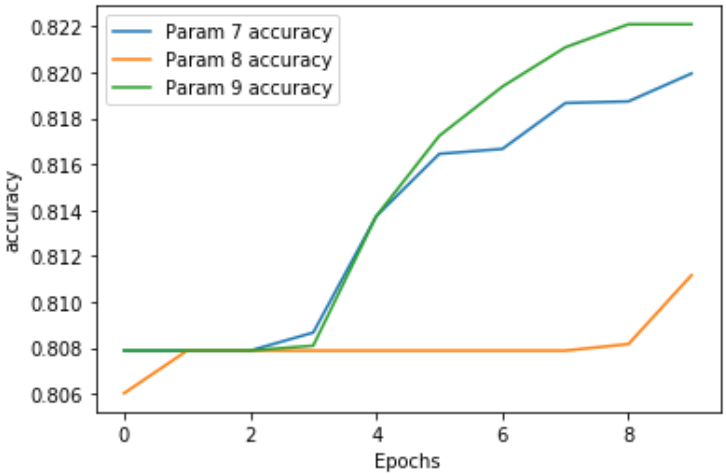
| ***Vector Length*** | 50 | 100 | 200 |
|---|---|---|---|
| ***Layer 1*** | Softmax | Softmax | Softmax |
| ***Layer 2*** | relu | relu | relu |
| ***Layer 3*** | Softmax | Softmax | Softmax |
| ***Epoch*** | 10 | 10 | 10 |

Berikut hasil akurasi dari pelatihan model skenario 2 yang dapat dilihat pada tabel 5.15

Tabel 5.15 Hasil Akurasi Skenario 3

| | **Parameter 7** | **Parameter 8** | **Parameter 9** |
|---|---|---|---|
| **1** | 0.8079 | 0.8060 | 0.8079 |
| **2** | 0.8079 | 0.8079 | 0.8079 |
| **3** | 0.8079 | 0.8079 | 0.8079 |
| **4** | 0.8087 | 0.8079 | 0.8081 |
| **5** | 0.8137 | 0.8079 | 0.8137 |
| **6** | 0.8165 | 0.8079 | 0.8172 |
| **7** | 0.8167 | 0.8079 | 0.8194 |
| **8** | 0.8187 | 0.8079 | 0.8211 |
| **9** | 0.8187 | 0.8082 | 0.8221 |
| **10** | 0.8200 | 0.8112 | 0.8221 |

Model pada skenario 3 memberikan hasil performa akurasi yang paling buruk di antara dua skenario lainnya yang menggunakan fungsi aktivasi yang sama. Penggunaan Word2Vec sebagai *word embedding* tidak memberikan hasil yang memuaskan saat melakukan pelatihan model. Hasil Akurasi tertinggi yang didapat hanya 0,8221, lebih kecil 0,25 dibanding dengan menggunakan model Word2Vec yang diberikan bobot (WWE-W2V). Kenaikan akurasi dari skenario 3 dapat dilihat pada gambar 3




Gambar 5.3 Plot Grafik Akurasi Skenario 3

### 5.5.4. Skenario 4

Pada skenario 4 akan melakukan perbandingan antara model *word embedding* dengan *Weighted Word Embedding* menggunakan Word2Vec (WWE-W2V) yang memiliki panjang 50, 100, dan 200. Pada skenario 1 ini menggunakan fungsi aktivasi *relu* pada seluruh lapisannya kecuali lapisan luar atau *output* yang menggunakan *softmax*.

Tabel 5.16 Parameter Skenario 4

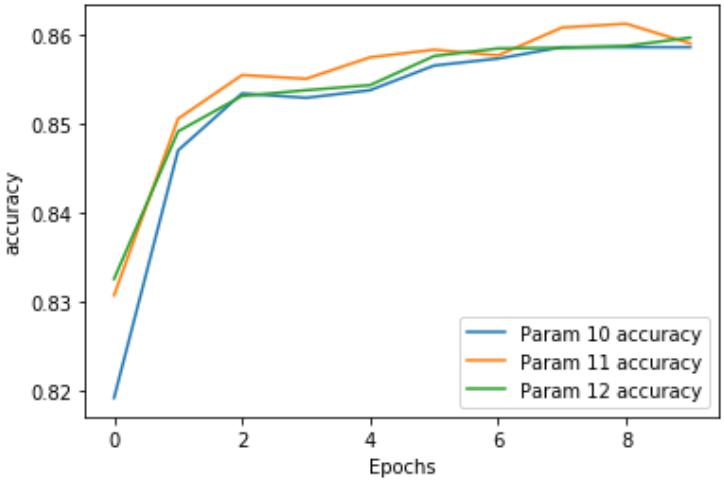
| | **Parameter 10** | **Parameter 11** | **Parameter 12** |
|---|---|---|---|
| ***Word Embedding*** | WWE-W2V | WWE-W2V | WWE-W2V |
| ***Fold*** | 10 | 10 | 10 |
| ***Vector Length*** | 50 | 100 | 200 |
| ***Layer 1*** | Relu | Relu | Relu |
| ***Layer 2*** | Relu | Relu | Relu |
| ***Layer 3*** | Softmax | Softmax | Softmax |
| ***Epoch*** | 10 | 10 | 10 |

Berikut hasil akurasi dari pelatihan model skenario 2 yang dapat dilihat pada tabel 5.17

Tabel 5.17 Hasil Akurasi Skenario 4

| | **Parameter 10** | **Parameter 11** | **Parameter 12** |
|---|---|---|---|
| **1** | 0.8191 | 0.8307 | 0.8325 |
| **2** | 0.8470 | 0.8506 | 0.8492 |
| **3** | 0.8534 | 0.8555 | 0.8532 |
| **4** | 0.8529 | 0.8551 | 0.8538 |
| **5** | 0.8538 | 0.8575 | 0.8544 |
| **6** | 0.8566 | 0.8584 | 0.8577 |
| **7** | 0.8574 | 0.8577 | 0.8585 |
| **8** | 0.8587 | 0.8609 | 0.8586 |
| **9** | 0.8587 | 0.8613 | 0.8588 |
| **10** | 0.8587 | 0.8591 | 0.8597 |

Tabel 5.17 menunjukkan adanya kenaikan performa dari penggunaan fungsi aktivasi *relu* pada *layer* pertema dibandingkan dengan menggunakan aktivasi *softmax* seperti pada skenario 1. Meskipun kenaikan akurasi tidak begitu banyak, namun setidaknya nilai akurasi dapat mencapai 0,86 dibanding dengan skenario 1 yang hanya 0,84. Panjang vektor pada skeario ini tidak terlalu memberikan pengaruh yang signifikan karena perbandingan akurasi antar parameter tidak terlalu jauh. Kenaikan akurasi dari skenario 4 dapat dilihat pada gambar 4




Gambar 5.4 Plot Grafik Akurasi Skenario 4

### 5.5.5. Skenario 5

Pada skenario 5 akan melakukan perbandingan antara model *word embedding* dengan *Weighted Word Embedding* menggunakan FastText (WWE-FT) yang memiliki panjang 50, 100, dan 200. Pada skenario 1 ini menggunakan fungsi aktivasi *relu* pada seluruh lapisannya kecuali lapisan luar atau *output* yang menggunakan *softmax*.

Tabel 5.18 Parameter Skenario 5

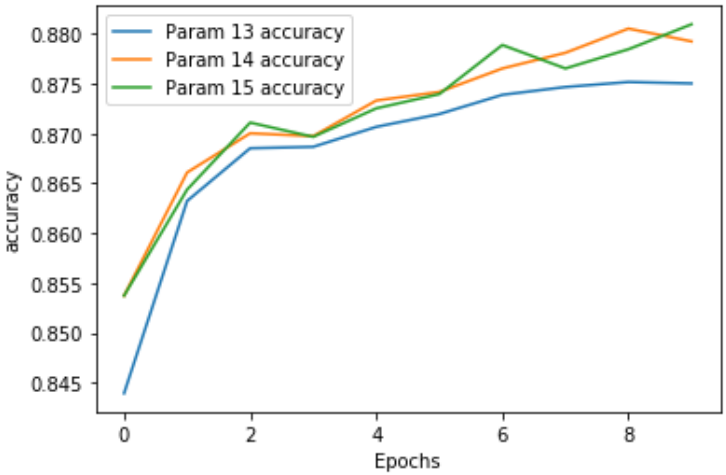
| | **Parameter 13** | **Parameter 14** | **Parameter 15** |
|---|---|---|---|
| ***Word Embedding*** | WWE-FT | WWE-FT | WWE-FT |
| ***Fold*** | 10 | 10 | 10 |
| ***Vector Length*** | 50 | 100 | 200 |
| ***Layer 1*** | Relu | Relu | Relu |
| ***Layer 2*** | Relu | Relu | Relu |
| ***Layer 3*** | Softmax | Softmax | Softmax |
| ***Epoch*** | 10 | 10 | 10 |

Berikut hasil akurasi dari pelatihan model skenario 2 yang dapat dilihat pada tabel 5.19

Tabel 5.19 Hasil Akurasi Skenario 5

| | **Parameter 13** | **Parameter 14** | **Parameter 15** |
|---|---|---|---|
| **1** | 0.8440 | 0.8537 | 0.8537 |
| **2** | 0.8632 | 0.8661 | 0.8644 |
| **3** | 0.8685 | 0.8700 | 0.8711 |
| **4** | 0.8687 | 0.8697 | 0.8697 |
| **5** | 0.8707 | 0.8733 | 0.8725 |
| **6** | 0.8719 | 0.8742 | 0.8739 |
| **7** | 0.8739 | 0.8765 | 0.8789 |
| **8** | 0.8747 | 0.8781 | 0.8765 |
| **9** | 0.8752 | 0.8805 | 0.8784 |
| **10** | 0.8750 | 0.8792 | 0.8809 |

Dari Tabel 5.19 di atas dapat dilihat hasil dari akurasi skenario 5. Dengan menggunakan model WWE-FT, hasil akurasi dari pelatihan model bertambah hingga menghasilkan akurasi paling tinggi yaitu pada *epoch* 10 sebesar 0,8809 yang memiliki panjang vektor 200. Panjang vektor yang dijadikan perbandingan tidak memberikan dapak yang signifikan pada model klasifikai, namun memang memberikan hasil akurasi yang lebih konsisten dibandingkan dengan yang lain. Kenaikan akurasi dari skenario dapat dilihat pada gambar 5




Gambar 5.5 Plot Grafik Akurasi Skenario 5

### 5.5.6. Skenario 6

Pada skenario 6 akan melakukan perbandingan antara model *word embedding* dengan menggunakan Word2Vec yang memiliki panjang 50, 100, dan 200. Pada skenario 1 ini menggunakan fungsi aktivasi *relu* pada seluruh lapisannya kecuali lapisan luar atau *output* yang menggunakan *softmax*.

Tabel 5.20 Parameter Skenario 6

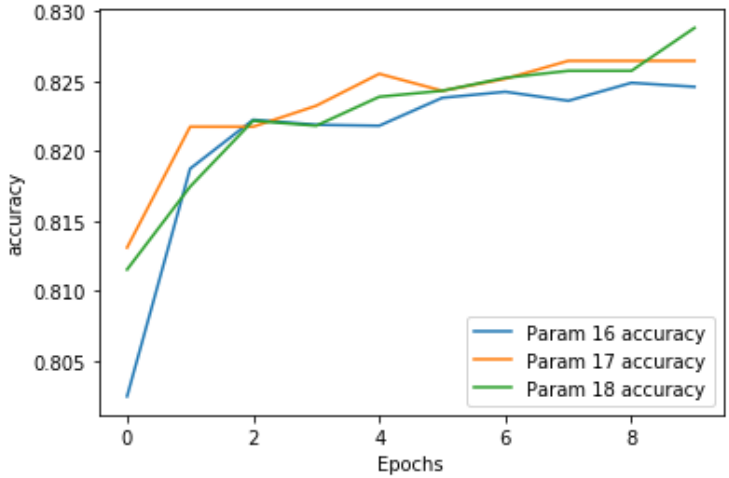
| | **Parameter 16** | **Parameter 17** | **Parameter 18** |
|---|---|---|---|
| ***Word Embedding*** | Word2Vec | Word2Vec | Word2Vec |
| ***Fold*** | 10 | 10 | 10 |
| ***Vector Length*** | 50 | 100 | 200 |
| ***Layer 1*** | Relu | Relu | Relu |
| ***Layer 2*** | Relu | Relu | Relu |
| ***Layer 3*** | Softmax | Softmax | Softmax |
| ***Epoch*** | 10 | 10 | 10 |

Berikut hasil akurasi dari pelatihan model skenario 2 yang dapat dilihat pada tabel 5.21

Tabel 5.21 Hasil Akurasi Skenario 6

| | **Parameter 13** | **Parameter 14** | **Parameter 15** |
|---|---|---|---|
| **1** | 0.8115 | 0.8086 | 0.8138 |
| **2** | 0.8192 | 0.8207 | 0.8188 |
| **3** | 0.8205 | 0.8226 | 0.8233 |
| **4** | 0.8223 | 0.8244 | 0.8253 |
| **5** | 0.8250 | 0.8254 | 0.8282 |
| **6** | 0.8272 | 0.8258 | 0.8303 |
| **7** | 0.8259 | 0.8286 | 0.8305 |
| **8** | 0.8263 | 0.8304 | 0.8290 |
| **9** | 0.8283 | 0.8294 | 0.8313 |
| **10** | 0.8275 | 0.8316 | 0.8315 |

Dari Tabel 5.21 menunjukkan hasil dari skenario 6 yang hasilnya tidak sebagus performa skenario lainnya. Penggunaan model *word embedding* Word2Vec tanpa bobot cenderung memiliki performa hasil yang tidak sebagus menggunakan bobot. Hasil akurasi tertinggi adalah 0,8316. Kenaikan akurasi dapat dilihat pada gambar 6

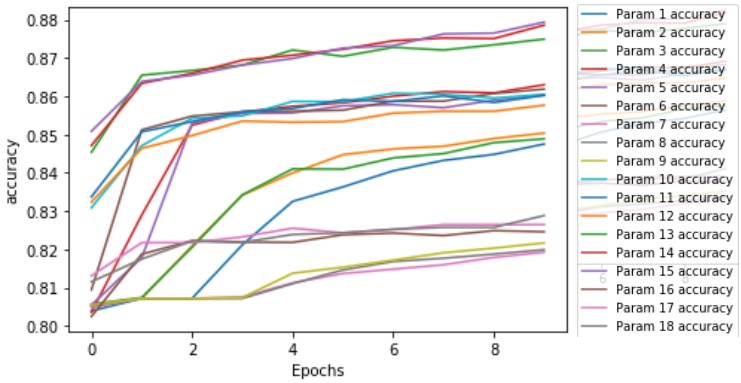



Gambar 5.6 Plot Grafik Akurasi Skenario 6

### 5.5.7. Perbandingan Skenario

Setelah menjalankan semua skenario yang telah dibuat. Selanjutnya yang harus dilakukan adalah melakukan perbandingan performa akurasi dari setiap skenario. Tujuannya untuk mencari model mana yang paling bagus dan akan dipilih untuk melakukan prediksi menggunakan data uji.

Dari 6 skenario dan 18 parameter yang dibandingkan. Terdapat 1 skenario yang memiliki skor akurasi yang paling tinggi yaitu skenario 5. Pada skenario 5 dengan panjang vektor 200 memiliki akurasi hingga 0,88. Berikut grafik perbandingan tiap parameter yang dapat dilihat pada gambar 7




Gambar 5.7 Grafik Perbandingan Akurasi

Dari gambar di atas dapat dilihat kalau skenario 5 memiliki akurasi yang paling unggul diantara semua skenario. Parameter 18 mengahasilkan akurasi tertinggi yaitu 0,88. Penggunaan FastText yang diberikan bobot memang memberikan dampak pada performa model klasifikasi. Vektor dengan panjang 200 memang tidak memberikan kenaikan akurasi yang signifikan dibandingkan dengan vektor dengan panjang 50 dan 100. Namun memberikan hasil yang sedikit lebih konsisten dan memberikan hasil akurasi yang paling tinggi dibandingkan yang lainnya. Selanjutnya Skenario 5 dengan parameter 18 akan

dijadikan model yang dipilih untuk melakukan prediksi namun sebelum itu akan dilakukan validasi terlebih dahulu.

## 5.6. Percobaan FastText

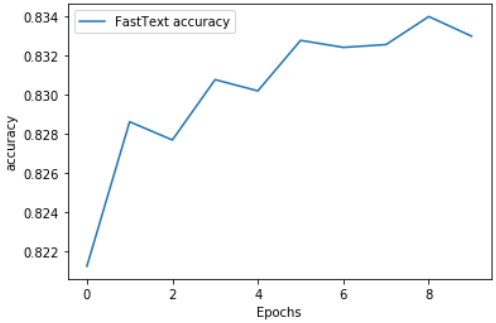
Setelah dilakukan beberapa percobaan model *word embedding* ditemukan model terbaik yaitu model FastText yang diberikan bobot. Pada percobaan kali ini akan dicoba bagaimana hasil apabila hanya menggunakan FastText saja tanpa menggunakan bobot. Dengan menggunakan parameter yang sama seperti pada skenario 5 yaitu dengan panjang vektor 200.

Dari percobaan menggunakan FastText ini, memberikan hasil performa model seperti pada tabel 5.22.

Tabel 5.22 Tabel Performa FastText

| | |
|---|---|
| ***Precision*** | 0.790 |
| ***Recall*** | 0.834 |
| ***Accuracy*** | 0.834 |
| ***F-measure*** | 0.793 |

Dan berikut grafik kenaikan akurasi pada tiap *epoch* dapat dilihat pada gambar 5.8




Gambar 5.8 Grafik Akurasi Menggunakan FastText

## 5.6. 10-*Fold Cross Validation*

Untuk memvalidasi model klasifikasi yang dibuat pada tugas akhir ini penulis menggunakan *K-fold Cross Validation* dengan nilai k sebesar 10. *K-fold cross validation* yang dilakukan menggunakan *library* dari sklearn.

```
acc_per_fold = []
loss_per_fold = []

kfold = KFold(n_splits=num_folds, shuffle=True)
# K-fold Cross Validation model evaluation
fold_no = 1
for train, test in kfold.split(inputs, targets):
    model19 = tf.keras.Sequential([
    tf.keras.layers.LSTM(200, input_shape=(1, 200), activation='relu'),
    tf.keras.layers.Dense(200, activation='relu'),
    tf.keras.layers.Dense(3, activation='softmax'),
    ])

    model19.compile(loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
              optimizer='adam',
              metrics=['accuracy'])

    print('------------------------------------------------------------------------')
    print(f'Training for fold {fold_no} ...')

    history19 = model19.fit(inputs[train], targets[train],
              epochs=10)

    scores = model19.evaluate(inputs[test], targets[test], verbose=0)
    print(f'Score for fold {fold_no}: {model19.metrics_names[0]} of {scores[0]};
          {model19.metrics_names[1]} of {scores[1]*100}%')
    acc_per_fold.append(scores[1] * 100)
    loss_per_fold.append(scores[0])

    fold_no = fold_no + 1
```

Kode Program 5.6 K-Fold Cross Validation

Karena jumlah *fold* yang ingin digunakan sebanyak 10, maka nilai `num_folds` yang dimasukkan adalah 10. Sehingga nantinya data akan terbagi ke dalam 10 *fold.*

Validasi model dilakukan menggunakan data validasi yang sudah diambil 10% dari data pelatihan. Hasil dari validasi model ini dapat dilihat dari tabel 5.23 berikut.

Tabel 5.23 Hasil Cross Validaton dengan 10 Fold

| ***Fold*** | ***Accuracy*** | ***Loss*** |
|---|---|---|
| 1 | 85.15% | 0.4025 |
| 2 | 84.51% | 0.3889 |
| 3 | 86.82% | 0.3643 |
| 4 | 85.79% | 0.3875 |
| 5 | 83.68% | 0.4285 |
| 6 | 85.66% | 0.3781 |
| 7 | 85.28% | 0.3911 |
| 8 | 86.05% | 0.3837 |
| 9 | 86.17% | 0.3573 |
| 10 | 86.10% | 0.3955 |

Dari validasi model dapat kita lihat akurasi dari model agak memurun dari dibandingkan dengan pelatihan model sebelumnya. Skor *loss* pada validasi model ini menunjukkan hasil yang relatif kecil sehingga dapat dikatakan model ini tidak *overfittin* dan siap untuk dilakukan prediksi. Rerata dari hasil akurasi dan *loss* secara berturut-turut adalah 85,54% dan 0,3895.

## 5.7. Prediksi Model

Setelah mendapatkan model yang paling bagus dari mencoba setiap skenario dan melakukan validasi pada model tersebut. Hal selanjutnya yang dilakukan yaitu melakukan uji prediksi pada model menggunakan data uji (*test*). Sebelum melakukan prediksi data kita melakukan evaluasi model terlebih dahulu untuk melihat hasil akurasi dan *loss* pada model. Berikut hasil evaluasi model yang dapat dilihat pada tabel 5.24.

Tabel 5.24 Hasil Evaluasi Model

| | |
|---|---|
| **Akurasi** | 0.8683 |
| ***Loss*** | 0.3674 |

Hasil akurasi yang didapatkan pada data pengujian adalah 0,8683. Akurasi ini lebih kecil dibanding pada data pelatihan.

Skor *loss* yang dihasilkan dari evaluasi ini dapat dibilang kecil dan ini menandakan bahwa model tidak *overfitting*.

Selanjutnya kita langsung melakukan prediksi pada model yang dapat dilihat pada tabel 5.25.

Tabel 5.25 Hasil Prediksi Model

| Teks | Label | Prediksi |
|---|---|---|
| siang kai bisa intern kuliah semester fakultas manajemen | Informasi | Informasi |
| berniat beli tiket kursi penumpang kursi kereta eko progo | Informasi | Informasi |
| penasaran penumpang telat tiket hangus keretanya telat berangkat kompensasinya eks lho | Keluhan | Keluhan |
| ditemukan salah tempel huruf kursi ka harina kertajaya | Informasi | Informasi |

Prediksi model pada data pengujian memberikan hasil yang cukup baik dalam memprediksi teks ke setiap kategorinya. Ketepatan model dalam memprediksi dapat dilihat pada tabel 5.26

Tabel 5.26 Confussion Matrix Prediksi

| Data Aktual | Prediksi | | |
|---|---|---|---|
| | Informasi | Keluhan | Saran |
| Informasi | 2984 | 161 | 1 |
| Keluhan | 230 | 387 | 2 |
| Saran | 61 | 57 | 6 |

Dari 3889 data pengujian, terdapat 3146 berkategori Informasi, 619 berkategori Keluhan dan 124 berkategori Saran/Permintaan. Model RNN dengan metode *word embedding* menggunakankan *weighted* FastText ini memberikan hasil akurasi sebesar 0,868. Model ini dapat dengan secara tepat memprediksi 2984 kategori Informasi, 387 kategori Keluhan, dan 6 kategori Saran/Permintaan.

Pada kategori Saran/Permintaan hanya sedikit teks yang secara tepat diprediksi. Hal ini menurut penulis dikarenakan oleh sedikitnya data pelatihan pada kategori tersebut. Selain itu adanya ambiguitas kata pada kategori tersebut yang juga terdapat pada kategori Informasi dan kategori Keluhan yang notabene memiliki lebih banyak data pelatihan. Secara kategori, model dapat mempediksi kategori Informasi secara tepat dengan skor presisi 0,911 sedangakn untuk kategori Keluhan dan Saran/Permintaan secara bertutrut-turut adalah 0,640 dan 0,667.

Hasil performa data pengujian secara keseluruhan dapat dilihat pada tabel 5.27.

Tabel 5.27 Performa Model

| | |
|---|---|
| ***Precision*** | 0.860 |
| ***Recall*** | 0.868 |
| ***Accuracy*** | 0.868 |
| ***F-measure*** | 0.855 |

Hasil *precison, recall, accuracy,* dan *f-measure* dari model RNN dengan *Weighted Word Embedding* FastText secara berturut-turut adalah 0,860; 0,868; 0,868; dan 0,855.

### 5.8. Uji Coba Lainnya

Pada uji coba ini, akan dilakukan pengujian model yang sudah dibuat diatas untuk dipastikan model benar-benar melakukan prediksi secara tepat. Uji coba ini dilakukan dengan menggunakan salah satu data latih. Sehingga pada uji coba ini akan dilakukan prediksi menggunakan hanya 1 data saja. Data ini merupakan data berlabel ‘Informasi’. Data berisi seperti ini [‘info’, ‘pemesanan’, 'tiket', 'lokal', 'pembelian', 'tiket', 'prameks', 'go', 'show', 'menit', 'sblom', 'berangkat', 'layani', 'surabaya', 'turun', 'solo'].

| Index | Kelas | Tweet |
|---|---|---|
| 0 | Informasi | info pemesanan tiket lokal pembelian tiket prameks go show menit sblom berangkat layani surabaya turun solo |

Gambar 5.9 Tweet Data Latih

Pada gambar 5.8 dapat dilihat bahwa cuitan masuk kedalam kelas informasi. Setelah dilakukan *word embedding* dengan menggunakan bobot seperti pada model dengan skenario 5. Hasil yang berupa vektor dimensi ini nantinya akan dimasukkan ke dalam model klasifikasi. Model klasifikasi ini akan melihat jarak antar vektor dimensi untuk menentukkan kategori dari setiap cuitan.

Hasil dari pelatihan ditunjukkan pada gambar 5.9. Hasil prediksi dari model ini berupa probabilitas dari tiap kategori. Angka 0 merepresentasikan kategori Informasi, angka 1 untuk kategori Keluhan, dan 2 untuk kategori Saran/Permintaan. Dari gambar dapat dilihat bahwa cuitan ini lebih condong pada probabilitas kategori Informasi oleh karenanya dia masuk ke dalam kategori Informasi

| | 0 | 1 | 2 |
|---|---|---|---|
| 0 | 0.981632 | 0.0116687 | 0.00669912 |

Gambar 5.10 Hasil Uji Coba

Dari hasil percobaan uji ini didapatkan hasil yang disajikan dalam Tabel 5.28

Tabel 5.28 Hasil Percobaan Lain

| **Prediksi Kelas** | Informasi |
|---|---|
| **Akurasi** | 100% |
| ***Loss*** | 0.0045 |
| ***Precision*** | 100% |
| ***Recall*** | 100% |
| ***F-measure*** | 100% |

Dari hasil diatas dapat ditarik kesimpulan bahwa model yang dibuat sudah melakukan prediksi sesuai dengan yang diharapkan karena memberikan hasil yang tepat dalam melakukan prediksi pada salah satu data latihnya.

*“Halaman ini sengaja dikosongkan”*

# BAB 6
# KESIMPULAN DAN SARAN

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

## 7.1. Kesimpulan

Kesimpulan yang didapatkan dari pengerjaan tugas akhir yang telah dilakukan adalah sebagai berikut:

a. Tugas Akhir ini telah berhasil mengimplementasikan model pengklasifikasi teks berbasis *recurrent neural network* (RNN) yang dikombinasikan dengan model *weighted word embedding* (WWE) untuk melakukan klasifikasi jenis pengaduan secara otomatis di sebuah perusahaan transportasi. Penggabungan kedua model ini mampu memberikan hasil akurasi klasifikasi yang lebih baik dibandingkan dengan penggabungan model pengklasifikasi RNN dan model *word embedding* biasa.

b. Hasil uji coba kinerja implementasi model penggabungan RNN dan WWE menunjukkan bahwa implementasi WWE baik yang menggunakan model *FastText* (*Weighted FastText*) maupun model *Word2Vec* (*Weighted Word2Vec*) memberikan hasil yang lebih baik dibandingkan dengan hasil kinerja yang menggabungkan RNN dan model *word embedding* biasa. Dengan menggunakan metode evaluasi berbasis 10-*fold cross validation*, model gabungan *RNN-Weighted FastText* dan *RNN-Weighted Word2Vec* berturut-turut memberikan hasil akurasi sebesar 88,2% dan 87,5%. Di lain pihak, dengan menggunakan metode evaluasi yang sama, model gabungan *RNN- FastText* dan *RNN-Word2Vec* berturut-turut memberikan hasil akurasi sebesar 83,4% dan 83,4%.

c. Panjang dimensi vektor hasil *word embedding* juga mempengaruhi kinerja dari model. Semakin panjang dimensi yang digunakan, maka semakin tinggi akurasi yang

dapat diperoleh, walaupun perbedaannya tidak begitu signifikan. Sebagai contoh, hasil uji coba menunjukkan bahwa penggunaan *word embedding FastText* dengan panjang dimensi sebesar 200, 100, dan 50 berturut-turut memberikan hasil akurasi *RRN-Weighted FastText* sebesar 88,2%, 87,9%, dan 87,5%.

## 7.2. Saran

Seperti dijelaskan dalam kesimpulan bahwa model Pengklasifikasi gabungan RRN-WWE yang diimplementasikan dalam Tugas Akhir ini mampu memberikan hasil klasifikasi yang relatif tinggi. Namun demikian, berikut diberikan beberapa saran untuk meningkatkan kinerja dari model yang diimplementasikan:

a. Diperlukan data pengaduan yang lebih banyak agar data latih yang digunakan dalam membangun model pengklasifikasi dapat mewakili sebagian besar keberagaman jenis pengaduan yang disampaikan oleh pengguna perusahaan transportasi yang menjadi objek dari Tugas Akhir ini. Saran ini didasarkan pada adanya kesulitan untuk memperoleh data yang lebih banyak karena kendala masa pandemi covid-19.
b. Perlu penegasan perbedaan antara jenis pengaduan berlabel “Informasi” dan jenis pengaduan “Saran/Permintaan”. Hal ini dikarenakan adanya ambiguitas teks yang termasuk dalam kedua jenis pengaduan tersebut, sehingga banyak teks yang seharusnya diklasifikasi sebagai jenis pengaduan “Saran/Permintaan” yang diklasifikasi sebagai jenis keluhan “Informasi”.

# DAFTAR PUSTAKA

[1] M. C. &. Q. Monitoring, “State of Customer Service Experience 2018,” The Northridge Group, Inc., 2018.

[2] H.-K. Poon, W.-S. Yap, Y.-K. Tee, W.-K. Lee dan B.-M. Goi, “Hierarchical Gated Recurrent Neural Network with Adversarial and Virtual Adversarial Training on Text Classification,” *Neural Networks,* vol. 119, pp. 299-312, 2019.

[3] Y. Bengio, P. Simard dan P. Frasconi, “Learning long-term dependencies with gradient descent is difficult,” *IEEE Transactions on Neural Networks,* vol. 5, no. 2, pp. 157-166, 1994.

[4] S. Neelamegam dan D. E. Ramaraj, “Classification algorithm in Data mining: An Overview,” *International Journal of P2P Network Trends and Technology (IJPTT),* vol. 3, no. 5, 2013.

[5] R. Jindal, R. Malhotra dan A. Jain, “Techniques for text classification : Literature review and current trends,” *Webology,* vol. 12, no. 2, pp. 1-28, 2015.

[6] S. B. Kotsiantis, “Supervised Machine Learning: A Review of Classification,” *Informatica,* vol. 31, pp. 249-268, 2007.

[7] K. S. Jones, “A statistical interpretation of term specificity,” *Journal of Documentation,* vol. 28, no. 1, pp. 11-21, 1972.

[8] B. Guo, C. Zhang, J. Liu dan X. Ma, "Improving text classification with weighted word embeddings via a multi-channel TextCNN model," *Neurocomputing,* vol. 363, pp. 366-374, 2019.

[9] P. Liu, X. Qiu dan X. Huang, "Recurrent Neural Network for Text Classification with Multi-Task Learning," dalam *International Joint Conference on Artificial Intelligence*, New York, 2016.

[10] Y. Yang, "An Evaluation of Statistical Approaches to Text Categorization," *Information Retrieval ,* vol. 1, pp. 69-90, 1999.

[11] I. D. P. Wijana, "The Use of English in Indonesian Adolescent's Slang," *Journal of Language and Literature,* vol. 14, no. 1, pp. 8-19, 2014.

[12] M. Adriani, B. Nazief, J. Asian, S. Tahaghogh dan H. E. Williams, "Stemming Indonesian: A confix-stripping approach," *ACM Transactions on Asian Language Information Processing,* vol. 6, no. 4, pp. 1-33, 2007.

[13] D. Shen, G. Wang, W. Wang, M. R. Min, Q. Su, Y. Zhang, C. Li, R. Henao dan L. Carin, "Baseline Needs More Love: On Simple Word-Embedding-Based Models and Associated Pooling Mechanisms," *Computation and Language,* pp. 1-15, 2018.

[14] H. Ren, Z. Zeng, Y. C. Q. Du, Q. Li dan H. Xie, "A Weighted Word Embedding Model for Text Classification," dalam *International Conference on*

*Database Systems for Advanced Applications*, Chiang Mai, 2019.

[15] I. Sutskever, O. Vinyals dan Q. V. Le, "Sequence to Sequence Learning with Neural Networks," *Neural Information Processing Systems (NIPS),* 2014.

[16] G. Salton dan M. J. Grill, "Introduction to Modern Information Retrieval," *Library Management,* vol. 32, no. 4, pp. 373-374, 2011.

# BIODATA PENULIS

**Muhammad David Rahman,** biasa dipanggil dengan nama David. Penulis anak ke-3 dari tiga bersaudara, lahir di Bogor, 25 Juli 1998. Penulis lahir dan dibesarkan di Kota Hujan Bogor. Riwayat pendidikan yang pernah ditempuh penulis di antaranya SD Negeri Polisi 5 Bogor, SMP Negeri 1 Bogor, dan SMA Negeri 1 Bogor. Setelah itu, penulis melanjutkan studinya ke perguruan tinggi. Penulis masuk Departemen Sistem Informasi ITS melalui Jalur SBMPTN. Selama perkuliahan aktif menjadi panitia ISE! 2016 dan ISE! 2017, menjadi pengurus Badan Eksekutif Mahasiswa Fakultas Teknologi Informasi dan Komunikas (BEM FTIK). Selain itu penulis juga mengikuti beberapa kepanitian lainnya di lingkup jurusan, fakultas maupun tingkat institut. Penulis juga Penulis berkesempatan untuk menjadi asisten praktikum mata kuliah Desain dan Manjemen Jaringan Komputer. Pada tingkat akhir, penulis memutuskan untuk mengerjakan Tugas Akhir yang berkaitan dengan laboratorium RDIB. Untuk kepentingan penelitian penulis, silahkan menghubungi muhdavidrahman@gmail.com.