TUGAS AKHIR – KI141502

IMPLEMENTASI ALGORITMA PEMBELAJARAN BACKPROPAGATION UNTUK KLASIFIKASI TWEET RESOLUSI

LINGGAR JUWITA HANDAYANI
NRP 5112 100 031

Dosen Pembimbing
Diana Purwitasari, S.Kom., M.Sc.
Dini Adni Navastara, S.Kom., M.Sc.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2016

[Halaman ini sengaja dikosongkan]

FINAL PROJECT – KI141502

IMPLEMENTATION OF BACKPROPAGATION LEARNING ALGORITHM FOR CLASSIFICATION OF NEW YEAR'S RESOLUTION TWEET

LINGGAR JUWITA HANDAYANI
NRP 5112 100 031

Advisor
Diana Purwitasari, S.Kom., M.Sc.
Dini Adni Navastara, S.Kom., M.Sc.

INFORMATICS DEPARTMENT
Faculty of Information Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2016

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

IMPLEMENTASI ALGORITMA PEMBELAJARAN BACKPROPAGATION UNTUK KLASIFIKASI TWEET RESOLUSI

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Rumpun Mata Kuliah Komputasi Cerdas dan Visi
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh :
LINGGAR JUWITA HANDAYANI
NRP : 5112 100 031

Disetujui oleh Dosen Pembimbing Tugas Akhir :

Diana Purwitasari, S.Kom., M.Sc. .....
NIP: 197804102003122001 (pembimbing 1)

Dini Adni Navastara, S.Kom., M.Sc. .....
NIP: 198510172015042001 (pembimbing 2)

SURABAYA
JULI 2016

[Halaman ini sengaja dikosongkan]

IMPLEMENTASI ALGORITMA PEMBELAJARAN BACKPROPAGATION UNTUK KLASIFIKASI TWEET RESOLUSI

Nama Mahasiswa	: LINGGAR JUWITA HANDAYANI
NRP	: 5112 100 031
Jurusan	: Teknik Informatika ITS
Dosen Pembimbing I	: Diana Purwitasari, S.Kom., M.Sc.
Dosen Pembimbing II	: Dini Adni Navastara, S.Kom., M.Sc.

Abstrak

Satu Januari mungkin akan datang dan pergi seperti yang terjadi pada tahun-tahun sebelumnya. Sosial media akan dibanjiri resolusi Tahun Baru pengguna. Pengguna secara masal berbagi harapan pribadi untuk meningkatkan kualitas diri menjadi lebih baik. Terbukti dengan kelas-kelas di pusat olahraga menjadi mendadak penuh dan terjual habis. Begitu juga jumlah puntung rokok pada asbak menjadi lebih sedikit di awal tahun. Beragamnya tweet resolusi yang dibagikan pengguna, maka dari itu dilakukan klasifikasi tweet resolusi ke dalam kelas sejenis.

Pada Tugas Akhir ini metode yang akan diimplementasikan adalah klasifikasi multi kelas dengan Algoritma Backpropagation, dan optimasi gradient descent. Model jaringan yang digunakan adalah Multi-Layer Perceptron dengan neuron input sebanyak jumlah fitur, dan neuron output adalah enam, di mana masing-masing neuron mewakili sebuah kelas. Di antaranya enam kelas tersebut adalah Career & Education, Finance, Health & Fitness, Personal Growth, Recreation & Leisure, dan Relationship.

Uji coba Tugas Akhir ini menggunakan data set berupa teks kicauan (tweet) yang telah dikelompokkan menjadi enam kelas, yang kemudian diimplementasikan pada sistem untuk melihat akurasi yang dihasilkan. Dengan penentuan nilai learning rate, jumlah neuron pada hidden layer, dan jumlah epoch, akurasi yang didapat pada tahap pengujian mencapai hingga 50%.

Kata kunci: tweet, klasifikasi, backpropagation, resolusi tahun baru.

[Halaman ini sengaja dikosongkan]

IMPLEMENTATION OF BACKPROPAGATION LEARNING ALGORITHM FOR CLASSIFICATION OF NEW YEAR'S RESOLUTION TWEET

Name	: LINGGAR JUWITA HANDAYANI
NRP	: 5112 100 031
Major	: Informatics Engineering Department – ITS
Supervisor I	: Diana Purwitasari, S.Kom., M.Sc.
Supervisor II	: Dini Adni Navastara, S.Kom., M.Sc.

Abstract

January 1st probably came and went the same way as it did last year: with a whole host of New Year's resolutions on social media. People shared their thoughts and hopes to be better man in the future. As results, class at the gym is suddenly sold out and that ring of co-workers around the ashtray seems a bit less packed. With a diverse tweet the resolution of a given user, this Final Project will do the classification to classify similar tweets resolution into clas.

In this Final Project will be implemented method is a multi-class classification with Back Propagation Algorithm, with optimization methods such as gradient descent. Used network model was implemented with number of features as input layer's neuron, and six output layer's neuron that reflect as each class. The six mentioned classes were Career & Education, Finance, Health & Fitness, Personal Growth, Recreation & Leisure, and Relationship.

Final trials using a data set of a collection of text tweets that are divided into six classes already by the ground truth data. The test is done by implementing a set of data on the system to see the resulting accuracy. By determining the right value of learning rate, number of neurons in the hidden layer, and sum of epoch, the accuracy obtained at the stage of testing up to 50%.

Keywords: tweet, classification, backpropagation, new year resolution.

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

Segala puji dan syukur ke hadirat Allah SWT yang telah memberikan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul "*Implementasi Algoritma Pembelajaran Backpropagation untuk Klasifikasi Tweet Resolusi*".

Dalam pelaksanaan Tugas Akhir ini tentu penulis sebagai makhluk sosial tidak dapat menyelesaikannya tanpa bantuan dari pihak lain. Tanpa mengurangi rasa hormat, penulis memberikan penghargaan serta ucapan terima kasih yang kepada:

1. Orang tua, saudara serta keluarga besar di kampung halaman yang senantiasa memberikan semangat dan doa agar penulis dapat menyelesaikan Tugas Akhir dengan tepat waktu.
2. Ibu Diana Purwitasari, S.Kom, M.Sc selaku dosen pembimbing Tugas Akhir pertama yang telah membimbing, memotivasi dan memberikan banyak masukan dalam pengerjaan Tugas Akhir ini.
3. Ibu Dini Adni Navastara, S.Kom., M.Sc. selaku dosen pembimbing Tugas Akhir kedua yang selalu memberikan masukan-masukan yang dapat penulis kembangkan dari Tugas Akhir ini.
4. Bapak dan Ibu dosen Jurusan Teknik Informatika ITS yang telah mengajarkan banyak ilmu berharga kepada penulis.
5. Bapak dan Ibu karyawan Jurusan Teknik Informatika ITS atas berbagai bantuan yang telah diberikan kepada penulis selama masa perkuliahan.
6. Untuk sahabat saya Dewi Maya dan Vessa Rizky yang memberi dukungan hingga akhir pengerjaan Tugas Akhir.
7. Untuk Wahyu Siwi dan Freeska Rochma yang telah mengabiskan waktu lebih banyak pada beberapa semester terakhir.

8. Teman-teman Teknik Informatika ITS angkatan 2012, yang telah memberikan warna-warni kehidupan mahasiswa mulai sejak mahasiswa baru hingga lulus.
9. Pihak-pihak lain yang tidak sempat penulis sebutkan, yang telah membantu kelancaran pengerjaan Tugas Akhir ini.

Penulis sangat berharap bahwa apa yang dihasilkan dari Tugas Akhir ini bisa memberikan manfaat bagi semua pihak, khususnya bagi diri penulis sendiri dan seluruh *civitas academica* Teknik Informatika ITS, serta bagi agama, bangsa, dan negara. Tak ada manusia yang sempurna sekalipun penulis berusaha sebaik mungkin dalam menyelesaikan Tugas Akhir ini. Karena itu, penulis memohon maaf apabila terdapat kesalahan, kekurangan, maupun kelalaian yang telah penulis lakukan. Kritik dan saran yang membangun sangat diharapkan oleh penulis untuk dapat disampaikan untuk perbaikan selanjutnya.

Surabaya, Juni 2016

Linggar Juwita Handayani

DAFTAR ISI

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

DAFTAR KODE SUMBER

[Halaman ini sengaja dikosongkan]

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

Bab ini membahas garis besar penyusunan Tugas Akhir yang meliputi latar belakang, tujuan pembuatan, rumusan dan batasan permasalahan, metodologi penyusunan Tugas Akhir, dan sistematika penulisan.

1.1. Latar Belakang

Twitter merupakan salah satu layanan mikroblog yang sedang tren di kalangan pengguna internet. Hingga Mei 2015, Twitter telah memiliki lebih dari 500 juta pengguna, 302 juta di antaranya adalah pengguna aktif (Wikipedia, 2006). Berbasis teks 140 karakter, lebih dikenal dengan kicauan (*tweet*), Twitter memungkinkan pengguna bertukar informasi dengan cepat, serta kebebasan untuk menuangkan segala hal yang diinginkan dalam teks singkat sebagai alat komunikasi (Tenuto, 2015). Jumlah kicauan meningkat pada peristiwa tertentu, seperti peristitwa tahun baru. Pengguna secara masif berkicau tentang resolusi yang ingin dicapai menjelang tahun baru. Twitter yang bersifat *real-time* dapat menjadi sumber informasi bagi penyedia layanan barang dan jasa dalam strategi bisnis.

Teks *tweet* resolusi akan menjadi bahan yang sangat penting untuk melakukan identifikasi terhadap keinginan atau target pencapaian pengguna di tahun mendatang, di mana pengguna berperan sebagai pelanggan dalam bisnis layann barang dan jasa. Teks *tweet* resolusi perlu diolah dan diklasifikasikan menjadi beberapa kelas yang telah ditentukan, di antaranya karir dan pendidikan, keuangan, kesehatan dan kebugaran, pengembangan diri, rekreasi dan hiburan, serta hubungan sosial. Data set yang tersedia telah memiliki teks *tweet* beserta kelas hasil klasifikasi secara *manual*. Oleh karena itu, proses klasifikasi dilakukan dengan metode *supervised learning* di mana sistem melakukan *training* terhadap data yang telah tersedia, dan kemudian bisa

dilakukan klasifikasi terhadap data baru untuk memunculkan prediksi pada tahap *testing*.

Supervised Learning merupakan suatu pembelajaran yang terawasi di mana jika *output* yang diharapkan telah diketahui sebelumnya. Beberapa metode populer yang masuk sebagai kategori *supervised learning*, di antaranya adalah *Single-Layer Perceptron* dan *Multi-Layer Perceptron*. Dalam kasus ini akan dipilih *Multi-Layer Perceptron* dengan Algoritma *Backpropagation* sebagai metode pembelajaran terhadap data pelatihan. *Multi-Layer Perceptron* merupakan metode penerapan *Neural Network* (NN). *Multi-Layer Perceptron* memiliki kelebihan, yaitu kemampuan untuk mendeteksi atau melakukan analisa untuk permasalahan yang sifatnya sangat kompleks (Prasetiyo, 2015), jika dibandingkan dengan *Single-Layer Perceptron*.

Oleh karena itu pada Tugas Akhir ini dibangun sebuah sistem yang dapat mengenali teks *tweet* resolusi. Di mana kemudian sistem akan mengklasifikasi teks tersebut ke dalam kelas-kelas yang telah ditentukan, di antaranya karir dan pendidikan, keuangan, kesehatan dan kebugaran, pengembangan diri, rekreasi dan hiburan, serta hubungan sosial.

1.2. Rumusan Masalah

Rumusan masalah yang diangkat dalam Tugas Akhir dapat dipaparkan sebagai berikut.

1. Bagaimana melakukan klasifikasi teks *tweet* ke dalam kategori yang telah ditentukan?
2. Bagaimana mengekstraksi fitur teks berdasarkan kemunculan kata dalam dokumen?
3. Bagaimana melakukan prediksi kategori teks *tweet* dengan model jaringan yang telah ada?

1.3. Batasan Masalah

Permasalahan yang dibahas dalam Tugas Akhir memiliki beberapa batasan, yakni sebagai berikut.

1. Sistem dibangun berbasis *desktop*, dengan basis data dan bahasa pemrograman C# serta kerangka kerja .NET.
2. Data *tweet* resolusi yang digunakan adalah dataset *tweet* resolusi Tahun Baru 2015.
3. Data *tweet* resolusi diambil berdasarkan penggunaan *#newyearsresolution* pada teks *tweet*.
4. Data *tweet* resolusi yang akan diproses adalah teks Bahasa Inggris.
5. Data masukan *tweet* resolusi telah dikompilasi dalam bentuk dokumen *excel spreadsheet*, dengan tipe berkas Microsoft Excel (ekstensi *.xls* atau *.xlxs*).
6. Model Jaringan Saraf Tiruan yang digunakan adalah jenis *Multi Layer Perceptron* dengan satu *hidden layer*.
7. Kelas *tweet* terdiri dari enam kelas, di antaranya, *Career & Education*, *Finance*, *Health & Fitness*, *Personal Growth*, *Recreation & Leisure*, dan *Relationship*.

1.4. Tujuan

Tujuan dari pembuatan Tugas Akhir ini adalah mengimplementasikan Algoritma *Backpropagation* untuk klasifikasi *tweet* resolusi dengan menggunakan Kerangka Kerja Microsoft .NET.

1.5. Manfaat

Manfaat dari Tugas Akhir ini adalah untuk membantu penyedia layanan dan jasa dalam memberikan rekomendasi terhadap strategi bisnis dan penjualan. Misalnya diketahui bahwa topik rekreasi banyak diperbincangkan, dan pengguna menunjukkan ingin melakukan rekreasi atau berlibur. Informasi ini membantu penyedia layanan jasa liburan, rekreasi, maupun tempat wisata serta pihak terkait dalam menentukan strategi selanjutnya untuk meningkatkan pengujung. Mengadakan promo-promo menarik serta menawarkan paket liburan yang lebih hemat dapat menjadi salah satu contoh strategi yang muncul setelah mengetahui informasi hasil klasifikasi.

1.6. Metodologi

Tahap yang dilakukan untuk menyelesaikan Tugas Akhir ini adalah sebagai berikut.

1. **Penyusunan Proposal Tugas Akhir**
   Di dalam proposal tugas akhir ini dijelaskan mengenai deskripsi pendahuluan dari Tugas Akhir yang akan dibuat. Pendahuluan berisi dari latar belakang alasan adanya usulan Tugas Akhir, rumusan masalah yang dibawa, batasan- batasan permasalahan, tujuan serta manfaat dari pembuatan Tugas Akhir ini. Sebagai bahan referensi pendukung dari usulan Tugas Akhir yang akan dibuat, maka dijelaskan pula tinjauan pustaka. Selain itu, terdapat metodologi sebagai langkah-langkah dalam pembuatan Tugas Akhir, dimulai dari penyusunan proposal hingga penyusunan buku Tugas Akhir dan sebagai haluan jadwal pelaksanaan Tugas Akhir dilampirkan pula jadwal kegiatan. Pada proposal ini, penulis mengajukan gagasan mengenai implementasi Algoritma *Backpropagation* untuk klasifikasi *tweet* resolusi.
2. **Studi Literatur**
   Pada tahap ini dilakukan pencarian informasi dan studi literatur terkait beberapa teori pendukung serta pustaka pendukungnya di antaranya adalah Algoritma *Back Propagation*, Kerangka Kerja Microsoft .NET, dan pemrograman aplikasi berbasis *desktop*.
3. **Implementasi**
   Pada tahap ini dilakukan implementasi sistem yang dibangun dengan bahasa pemrograman C#. Berdasarkan Algoritma *Back Propagation*, implementasi dilakukan dengan melalui beberapa fase, yakni fase *feed forward*, fase *back propagation*, dan fase memperbarui bobot. Sistem ini

dibangun menggunakan *Integrated Development Environment* (IDE) Microsoft Visual Studio Ultimate 2013 dan Kerangka Kerja Microsoft .NET.

4. **Uji Coba dan Evaluasi**
   Pada tahap ini dilakukan uji coba aplikasi dan evaluasi terhadap implementasi metode pada aplikasi. Tahap uji coba mengukur persentase kebenaran akurasi hasil klasifikasi dengan data *ground truth* yang telah tersedia.
5. **Penyusunan Buku Tugas Akhir**
   Tahap ini merupakan tahap dokumentasi dari Tugas Akhir. Buku Tugas Akhir berisi dasar teori, perancangan, implementasi dan hasil uji coba dan evaluasi dari aplikasi yang dibangun.

1.7. Sistematika Penulisan

Buku Tugas Akhir ini terdiri atas beberapa bab yang tersusun secara sistematis, yaitu sebagai berikut.

1. Bab I Pendahuluan
   Bab pendahuluan berisi penjelasan mengenai latar belakang masalah, rumusan masalah, batasan masalah, tujuan, manfaat dan sistematika penulisan Tugas Akhir.
2. Bab II Tinjauan Pustaka
   Bab tinjauan pustakan berisi penjelasan mengenai dasar teori yang mendukung pengerjaan Tugas Akhir.
3. Bab III Analisis dan Perancangan
   Bab analisis dan perancangan berisi penjelasan mengenai analisis kebutuhan, perancangan sistem dan perangkat yang digunakan dalam pengerjaan Tugas Akhir serta urutan pelaksanaan proses.
4. Bab IV Implementasi
   Bab implementasi berisi pembangunan implementasi dengan Algoritma *Back Propagation* untuk klasifikasi *tweet* resolusi,

sesuai dengan rumusan dan batasan yang sudah dijelaskan pada bagian pendahuluan.

5. Bab V Uji Coba dan Evaluasi
   Bab uji coba dan evaluasi berisi pembahasan mengenai hasil dari uji coba yang dilakukan terhadap sistem klasifikasi *tweet* resolusi dengan Algoritma *Back Propagation*.
6. Bab VI Kesimpulan dan Saran
   Bab kesimpulan dan saran berisi kesimpulan hasil penelitian. Selain itu, bagian ini berisi saran untuk pengerjaan lebih lanjut atau permasalahan yang dialami dalam proses pengerjaan Tugas Akhir.

BAB II
TINJAUAN PUSTAKA

Bab tinjauan pustaka berisi mengenai penjelasan teori yang berkaitan dengan implementasi perangkat lunak. Penjelasan tersebut bertujuan untuk memberikan gambaran mengenai sistem yang akan dibangun dan berguna sebagai pendukung dalam pengembangan perangkat lunak.

2.1. Data Tweet

Tweet merupakan Tulisan yang dapat diposting oleh pengguna di timeline Twitter. Dalam satu *tweet* hanya mampu menampung tidak lebih dari 140 karakter. *Tweet* biasanya dapat berisi informasi, obrolan, ungkapan hati atau kata-kata motivasi yang ditulis oleh pengguna dengan tujuan berbagi informasi tersebut dengan pengguna dunia maya (Wikipedia, 2006).

Gambar 2.1 *Tweet* yang Dikirimkan di Timeline Twitter

Meskipun secara *default tweet* tersebut dibatasi hanya 140 karakter, pengguna Twitter tetap bisa menulis tweet dengan panjang lebih dari batas karakter yang ditentukan atau bahkan tanpa batas. Namun, *tweet* yang dibagikan tidak akan sepenuhnya ditampilkan pada halaman *timeline*, hanya sebagian karakter, yang dibatasi sepanjang 140 karakter, dan akan disertai *link* untuk melihat *tweet* secara utuh.

Tabel 2.1 Klasifikasi Berdasarkan Konten *Tweet* Resolusi

No	*Tweet*	Topik	Kelas
1	*#NewYearsResolution Stop doubting myself*	*Be more positive*	*Personal Growth*
2	*#NewYearsResolution to eat buy and eat environmentally responsible. buy for local businesses and eat a plant based diet. #eatclean2015*	*Eat healthier*	*Health & Fitness*
3	*Adventures. I'm ready for more of those. #NewYearsResolution*	*Take a trip*	*Recreation & Leisure*
4	*#NewYearsResolution will be moving to @ExploreGeorgia #Altanta going to school and becoming a hair dresser or a actor!*	*Get dream job*	*Career & Education*
5	*I'm going to stop being honest and start telling people whatever they want to hear. So much easier. #NewYearsResolution*	*Be better at keeping in touch with loved ones or friends*	*Relationship*
6	*@jeffreymarshnow My New Years Resolution: Be brave and speak up about issues that matter! #NewYearsResolution*	*Be more confident*	*Personal Growth*
7	*"New Years Resolution Save Money don't Spend Money"*	*Save money*	*Finance*
8	*New Years resolution: more brunch.*	*Eat more*	*Health & Fitness*

Terdapat beberapa elemen yang sering digunakan oleh pengguna saat menuliskan sebuah *tweet*, di antaranya adalah RT (*Retweet*), *hashtag*, *mention*, dan *link*. RT digunakan apabila seorang pengguna ingin meneruskan *tweet* pengguna lain dengan tetap mencantumkan sumber asli yang pertama kali membuat *tweet*. *Hashtag* dilambangkan dengan tanda pagar (#) digunakan agar pengguna Twitter dapat mencari kata kunci tertentu dengan mudah dan cepat. *Mention* merupakan suatu tindakan untuk memberi tahu pengguna Twiter dengan mencantumkan *username*nya pada *tweet* (*@username*) sehingga pengguna tersebut akan ikut membaca *tweet* yang dikirimkan (Manroe, 2014). Pengguna Twitter bisa mengirim *link* secara langsung melalui *tweet*, namun pada umumnya *link* muncul secara otomatis karena teks yang dikirimkan melalui *tweet* melebihi 140 karakter. Contoh *tweet* ditunjukkan Gambar 2.1.

Selanjutnya akan ditunjukkan contoh *tweet-tweet* yang akan digunakan sebagai data set pada sistem klasifikasi Tugas Akhir ini. Contoh data hasil klasifikasi berdasarkan konten *tweet* resolusi ditunjukkan pada Tabel 2.1. Kolom *Tweet* berisi teks *tweet* yang digunakan sebagai data set awal pada sistem ini. Kolom Topik merupakan isi atau intisari dari teks *tweet* tersebut. Dan, kolom Kelas menujukkan termasuk kelas apa teks *tweet* tersebut.

2.2. Praproses Teks

Pembangunan sistem klasifikasi *tweet* resolusi ini mengimplementasikan konsep *text mining*, yaitu suatu kegiatan penggalian informasi yang dibutuhkan, dari sekumpulan dokumen teks. Penggalian informasi tidak dapat dilakukan pada sumber daya teks mentahan dengan format teks yang masih bercampur antara karakter, kata, dan kalimat (Eldira, et al., 2010). Dan preproses merupakan proses fundamental atau dasar yang harus dilakukan untuk mendapatkan informasi inti dari dokumen teks. Informasi inti inilah yang akan diteruskan dan diolah ke tahapan proses selanjutnya (Gurusamy & Gurusamy, 2015).

Sumber data dokumen teks awal dikatakan mentahan karena pasti berisi hal format angka, tanggal, kata-kata umum yang tidak bermakna, serta kata-kata inti sudah berubah menjadi kata lain karena preposisi dan imbuhan-imbuhan yang ada. Dan hal-hal tersebut menutupi informasi inti dari suatu dokumen teks (Gurusamy & Gurusamy, 2015).

Tahapan praproses teks dalam menggali informasi inti pada *tweet* resolusi ini dibutuhkan empat langkah, yaitu *case folding*, *tokenizing*, *filtering*, dan *analyzing*. Penjelasan lebih detail adalah sebagai berikut.

- *Case Folding*
  Case folding merupakan tahapan yang mengubah semua huruf dalam dokumen menjadi huruf kecil. Hanya huruf 'a' sampai dengan 'z' yang diterima. Sehingga yang tampak berubah hanya format huruf, karakter selain huruf tidak akan terlihat perubahan.
- Tokenizing
  Tokenizing adalah proses penghilangan tanda baca pada kalimat yang ada dalam dokumen sehingga menghasilkan kata-kata yang berdiri sendiri-sendiri, secara istilah disebut *token*. Hasil dari proses ini tentu dokumen teks akan bersih dari tanda baca, angka, atau karakter-karakter lain yang tidak bermakna, dan antar *token* akan terpisah dengan spasi (Usmaida, 2007).
- *Filtering*
  Banyaknya kemunculan kata-kata dalam dokumen dapat diartikan sebagai kata yang mewakili dokumen, namun hal ini akan menjadi rusak dan mengaburkan inti dokumen dengan kata-kata penghubung yang juga akan sering muncul, seperti '*and*', '*then*', '*but*', '*after*', dan lain-lain. Kata-kata inilah yang akan menjadi hambatan untuk mengambil intisari dokumen, dan harus dihilangkan untuk dapat menyaring kata-kata bermakna yang mewakili dokumen. Hasil dari proses *tokenizing* akan disaring dan dihilangkan dari kata-kata tak bermakna yang masuk dalam daftar kata *stopword* (Usmaida,

2007). Hasil dari tahapan ini tentu daftar kata yang mewakili intisari dokumen.

- *Analyzing*
  Analyzing merupakan tahap di mana dokumen yang telah dipisah menjadi kata yang independen, akan dianalisa seberapa terkaitnya kata tersebut terhadap kata lain maupun dokumen. Pada tahap ini akan dihitung frekuensi sebuah kata pada masing-masing dokumen. Informasi ini sangat penting dan dibutuhkan untuk pembobotan yang akan diaplikasikan pada tahap selanjutnya.

2.3. Algoritma TF-IDF

Proses penggalian informasi dokumen teks atau lebih khususnya disebut teknologi sistem temu kembali dokumen teks, memiliki 2 tahapan awal yang penting. Selain praproses teks yang telah dijabarkan di atas, proses selanjutnya yang harus dilakukan adalah representasi teks. Tahap representasi teks ini biasa dikenal dengan tahapan pembobotan teks. Telah banyak penelitian-penelitian yang mengusulkan metode-metode baru untuk pembobotan teks, namun metode pembobotan yang sampai saat ini masih sering digunakan (*popular*) dan hasilnya masih dianggap yang terbaik karena efisien, sederhana, dan akurat adalah Metode *Term Frequency – Inverse Document Frequency*, atau lebih dikenal dengan metode pembobotan Algoritma TF-IDF.

Algoritma TF-IDF mempertimbangkan seringnya kemunculan *term* (kata) dalam dokumen dan rasio panjang dokumen tersebut di dalam *corpus* (sekumpulan dokumen teks) (Saadah, et al., 2012). Bobot dari perhitungan TF-IDF inilah yang menggambarkan seberapa pentingnya *term* (kata) dalam sebuah dokumen dan *corpus*.

Frekuensi kemunculan (*term frequency*) merupakan petunjuk sejauh mana *term* tersebut mewakili isi dokumen atau secara formula diartikan sebagai ukuran seringnya kemunculan sebuah *term* dalam sebuah dokumen dan juga dalam seluruh dokumen di dalam *corpus*. Semakin besar kemunculan suatu *term* dalam

dokumen akan memberikan nilai kesesuian yang semakin besar. *Term frequency* ini dihitung menggunakan Persamaan (2.1) dengan $tf_i(d_j)$ adalah notasi frekuensi kemunculan *term* ke-*i* dalam dokumen ke-*j*.

Faktor kebalikan yang diperhatikan juga dalam pemberian bobot TF-IDF adalah kejarangmunculan *term* (*Inverse Document Frequency*) dalam koleksi atau secara formula diartikan sebagai logaritma dari rasio jumlah seluruh dokumen dalam *corpus* dengan jumlah dokumen yang memiliki *term* yang dimaksud seperti yang dituliskan secara matematis pada Persamaan (2.2), di mana idf_i adalah frekuensi kemunculan *term* ke-i dalam seluruh dokumen atau satu *corpus*. *Term* yang muncul pada sedikit dokumen harus dipandang sebagai *term* yang lebih penting (*uncommon terms*) daripada *term* yang muncul pada banyak dokumen (Karmayasa & Mahendra, 2012).

Bobot TF-IDF sendiri menggabungkan kedua faktor penting di atas, dan secara formula nilai TF-IDF didapatkan dengan mengalikan nilai TF dengan nilai IDF, ditunjukkan pada Persamaan (2.3) di mana $(tf\text{-}idf)_{ij}$ adalah nilai bobot *term* ke-*i* dalam dokumen ke-*j*.

$$tf_i(d_j) = \frac{freq_i(d_j)}{\sum_{i=1}^{k} freq_i(d_j)} \tag{2.1}$$

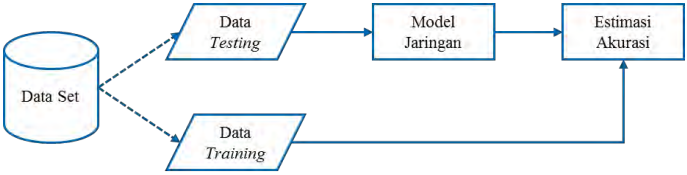
$$idf_i = \log \frac{|D|}{|\{d: t_i \in d\}|} \tag{2.2}$$

$$(tf - idf)_{ij} = tf_i(d_j) \,.\, idf_i \tag{2.3}$$

Di mana,
$freq_i(d_j)$ adalah frekuensi *term* ke- i dalam dokumen ke- j.
$\sum_{i=1}^{k} freq_i(d_j)$ adalah jumlah *term* pada dokumen ke- j.
$|D|$ adalah jumlah dokumen dalam *corpus*.
$|\{d: t_i \in d\}|$ adalah dokumen yang mengandung *term* ke- i.

2.4. Metode *Hold-out*

Metode *Hold-out* merupakan salah satu metode estimasi akurasi yang sering digunakan dan cukup populer dalam membantu menghitung nilai akurasi pada permasalahan klasifikasi maupun *clustering*. Dalam metode ini data dibagi menjadi dua kelompok data independen, yaitu data *training* dan data *testing*, secara acak (Sano, 2013). Secara khusus sesuai persentase yang ditentukan dari data dialokasikan dalam kelompok data *training*, dan sisanya ke dalam kelompok data *testing*. Data *training* digunakan untuk memperoleh model, dan akurasinya diestimasi menggunakan data *testing*. Estimasinya bersifat pesimis karena hanya sebagian data awal digunakan untuk memperoleh model.



Gambar 2.2 Estimasi Akurasi dengan Metode *Hold-out*

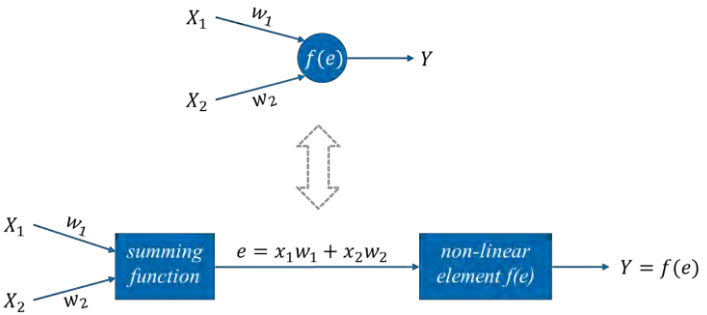
Gambar 2.2 menunjukkan ilustrasi metode *hold-out* dalam mengestimasi akurasi untuk diterapkan pada permasalahan klasifikasi. Pembagian data *training* dan data *testing* ditentukan oleh jumlah persentase yang ditetapkan. Misalkan diambil contoh untuk data *testing* adalah data yang diambil 10% dari data set total, artinya 90% dari data set total akan digunakan sebagai data *training*. Contoh lain ditunjukkan oleh Tabel 2.2.

Tabel 2.2 Pembagian Data Set dengan Metode *Hold-out*

No	Total Data Set	Persentase	Jumlah Data	
			Training	*Testing*
1	5000	10%	4500	500
2	7500	20%	6000	1500
3	7500	40%	4500	3000
4	10000	80%	8000	2000

2.5. Jaringan Saraf Tiruan

Jaringan Saraf Tiruan (JST) (Bahasa Inggris: *Artificial Neural Network* (ANN), atau juga disebut *Simulated Neural Network* (SNN), atau umumnya hanya disebut *Neural Network* (NN)), adalah jaringan dari sekelompok unit pemroses kecil yang dimodelkan berdasarkan jaringan saraf manusia. Jaringan Saraf Tiruan merupakan sistem adaptif yang dapat mengubah strukturnya untuk memecahkan masalah berdasarkan informasi eksternal maupun internal yang mengalir melalui jaringan tersebut. Oleh karena sifatnya yang adaptif, Jaringan Saraf Tiruan juga sering disebut dengan jaringan adaptif (Wikipedia, 2013).



Gambar 2.3 Model Jaringan Saraf Tiruan

Jaringan Saraf Tiruan merupakan salah satu metode yang cukup populer untuk klasifikasi. Beberapa alasan yang mendasari banyak peneliti mempelajari Jaringan Saraf Tiruan karena memiliki kemampuan untuk menyelesaikan data yang kompleks atau tidak tepat, serta kemampuan untuk menemukan pola untuk dikenali oleh manusia atau teknik komputasi lain. Berikut ini adalah dari penggunaan Jaringan Saraf Tiruan (Setiawan, 2011).

1. Dapat menyelesaikan tugas yang tidak dapat dikerjakan oleh program linear.
2. Saat sebuah elemen Jaringan Saraf Tiruan gagal, jaringan dapat terus berjalan tanpa masalah.

3. Sebuah model Jaringan Saraf Tiruan dapat belajar, sehingga tidak perlu diprogram ulang.
4. Dapat diterapkan pada sembarang aplikasi.
5. Dapat diterapkan tanpa banyak kendala.

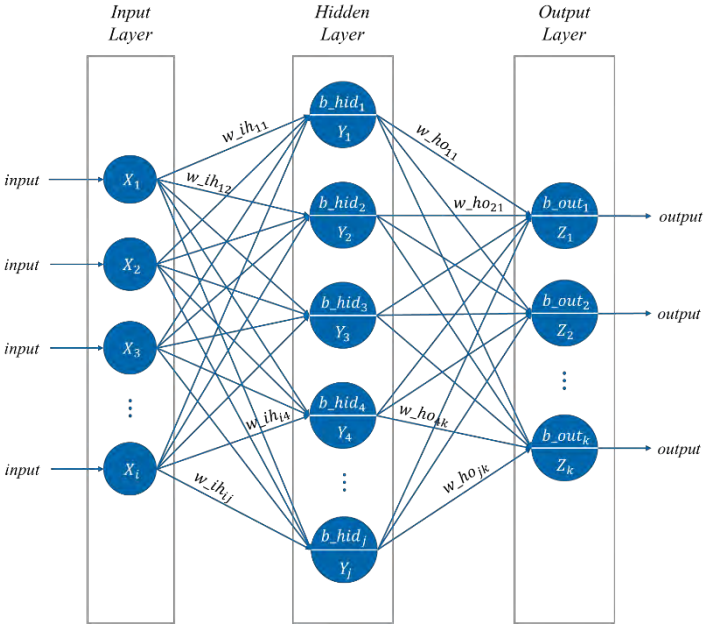
Namun, ada pula beberapa kekurangan yang dimiliki Jaringan Saraf Tiruan, di antaranya adalah memerlukan *training* pada data agar dapat beroperasi, dan memerlukan proses waktu yang lama untuk model jaringan yang besar dan rumit.

Secara sederhana, Jaringan Saraf Tiruan adalah sebuah alat pemodelan data statistik non-linier. Jaringan Saraf Tiruan dapat digunakan untuk memodelkan hubungan yang kompleks antara *input* dan *output* untuk menemukan pola-pola pada data (Wikipedia, 2013).

Model pada Jaringan Saraf Tiruan pada dasarnya merupakan fungsi model matematika yang mendefinisikan fungsi $f: X \rightarrow Y$. Istilah "jaringan" pada Jaringan Saraf Tiruan merujuk pada interkoneksi dari beberapa *neuron* yang diletakkan pada lapisan yang berbeda. Model Jaringan Saraf Tiruan ditunjukkan pada Gambar 2.3. Untuk penjelasan makna simbol akan dijelaskan pada subbab 2.5.1 pada Tabel 2.3. Secara umum, lapisan pada Jaringan Saraf Tiruan dibagi menjadi tiga bagian:

- Lapis masukan (*input layer*) terdiri dari *neuron* yang menerima data masukan dari variabel X. Semua *neuron* pada lapis ini dapat terhubung ke *neuron* pada lapisan tersembunyi atau langsung ke lapisan luaran jika jaringan tidak menggunakan lapisan tersembunyi.
- Lapisan tersembunyi (*hidden layer*) terdiri dari *neuron* yang menerima data dari lapisan masukan.
- Lapisan luaran (*output layer*) terdiri dari *neuron* yang menerima data dari lapisan tersembunyi atau langsung dari lapisan masukan yang nilai luarannya melambangkan hasil kalkulasi dari X menjadi nilai Y.

2.5.1. Algoritma *Backpropagation*



Gambar 2.4 Arsitektur Jaringan *Multi-Layer Perceptron*

Backpropagation merupakan salah satu metode pembelajaran yang diterapkan dalam Jaringan Saraf Tiruan. *Backpropagation* melakukan dua tahap komputasi, yaitu tahap umpan maju (*feedforward*) untuk menghitung *error* antara nilai *output* secara aktual dan nilai target, serta tahap propagasi balik (*backpropagation*) untuk menghitung mundur nilai *error* tersebut yang digunakan untuk memperbarui nilai bobot dan bias setiap *neuron* pada jaringan (Wikipedia, 2013).

Algoritma *Backpropagation* adalah salah satu algoritma yang menggunakan metode pembelajaran terawasi (*supervised learning*), serta termasuk jaringan *Multi-Layer Perceptron*. Metode *supervised learning* adalah teknik pembelajaran mesin dengan membuat suatu fungsi dari data *training*. Data *training*

terdiri dari pasangan nilai *input* dan *output* yang diharapkan dari *input* yang bersangkutan. Tugas dari *supervised learning* adalah untuk memprediksi nilai fungsi untuk nilai semua *input* yang ada (Mulyawati, 2012). Sedangkan, *Multi-Layer Perceptron* merupakan salah satu model Jaringan Saraf Tiruan yang memiliki satu atau lebih *hidden layer* yang terletak antara *input layer* dan *output layer*, seperti yang ditunjukkan oleh Gambar 2.4.

Tabel 2.3 Keterangan Rumus Algoritma *Backpropagation*

Notasi	Keterangan
i	Jumlah *neuron* pada *input layer*.
j	Jumlah *neuron* pada *hidden layer*.
k	Jumlah *neuron* pada *output layer*.
X_i	Nilai masukan ke-*i* pada *input layer*.
Y_j	Nilai aktivasi *neuron* ke-*j* tahap umpan maju pada *hidden layer*.
Z_k	Nilai aktivasi *neuron* ke-*k* tahap umpan maju pada *output layer*.
Y_in_j	Hasil penjumlahan sinyal masukan *neuron* ke-*j* pada *hidden layer*.
Z_in_k	Hasil penjumlahan sinyal masukan *neuron* ke-*k* pada *output layer*.
b_hid_j	Nilai bias *neuron* ke-*j* pada *hidden layer*.
b_out_k	Nilai bias *neuron* ke-*k* pada *output layer*.
w_ih_{ij}	Nilai bobot antara *neuron* ke-*i* pada *input layer* dan *neuron* ke-*j* pada *hidden layer*.
w_ho_{jk}	Nilai bobot antara *neuron* ke-*j* pada *hidden layer* dan *neuron* ke-*k* pada *output layer*.
t_k	Nilai target node ke-*k* pada *output layer*.
δ_hid_j	Nilai *gradient descent neuron* ke-*j* pada *hidden layer*.
δ_out_k	Nilai *gradient descent neuron* ke-*k* pada *output layer*.
α	Nilai *learning rate*.

Algoritma pembelajaran *backpropagation* dibagi menjadi dua fase, yaitu fase propagasi dan fase modifikasi bobot. Fase propagasi terdiri dari propagasi maju dari masukan data *training* untuk menghasilkan keluaran pada *output layer*, dan propagasi mundur di mana nilai keluaran pada *output layer* digunakan untuk menghitung *delta* (selisih nilai target dan nilai keluaran) setiap *neuron* pada *hidden layer* dan *output layer*. Sedangkan, yang termasuk pada fase modifikasi bobot adalah mengalikan hasil *delta* dan nilai masukan untuk mendapat nilai *gradient descent*, serta memperbarui nilai bobot dengan mengurangi nilainya sebanyak persentase dari nilai *gradient descent* (Wikipedia, 2013). Secara detail, langkah-langkah pengerjaan dan perhitungan Algoritma *Backpropagation* ditunjukkan pada persmaan-persamaan seperti di bawah ini. Dan, keterangan notasi serta simbol rumus algoritma dapat dilihat pada Tabel 2.3.

- Langkah 0
  Inisialisasi bobot secara acak. Inisialisasi bobot bertujuan untuk mengisi nilai *neuron* yang masih kosong pada model jaringan. Untuk menetukan banyak bobot yang perlu diinisialisasi dapat digunakan Persamaan (2.5).

  $$total_{bobot} = ij + jk + j + k \quad (2.5)$$

  Kemudian, inisialisasi nilai $epoch = 0$ dan menentukan nilai maksimum $epoch$ yang diinginkan.
- Langkah 1
  Kerjakan Langkah 2 selama $(epoch < epoch_{maksimum})$. Setiap Langkah 1 selesai dilakukan, perbarui nilai $epoch$ menjadi $epoch = epoch + 1$.
- Langkah 2
  Untuk setiap vektor dokumen yang masuk sebagai data *training* dan disimpan pada variabel X, lakukan Langkah 3 hingga Langkah 9. Jadi, pada langkah ini akan mulai dilakukan *training* untuk masing-masing vektor dokumen.

- Langkah 3
  Setiap *neuron input* $(X_i, i = 1, 2, \dots, p)$ menerima sinyal X_i dari vektor dokumen, kemudian meneruskan sinyal tersebut ke semua *neuron* pada *hidden layer*.
- Langkah 4
  Setiap *neuron* pada *hidden layer* $(Y_j, j = 1, 2, \dots, q)$ menjumlahkan sinyal-sinyal masukan dari *input layer* yang telah dibobotkan dengan Persamaan (2.6). Kemudian hitung nilai aktivasi dengan Persamaan (2.7).

$$Y\_in_j = b\_hid_j + \sum_{i=1}^{p} X_i * w\_ih_{ij} \quad (2.6)$$

$$Y_j = \frac{1}{1 + e^{Y\_in_j}} \quad (2.7)$$

- Langkah 5
  Setiap *neuron* pada *output layer* $(Z_k, k = 1, 2, \dots, r)$ menjumlahkan sinyal-sinyal masukan yang telah dibobotkan dengan Persamaan (2.8). Kemudian hitung sinyal nilai aktivasi seperti Persamaan (2.9).

$$Z\_in_k = b\_out_k + \sum_{i=1}^{p} X_i w\_ho_{jk} \quad (2.8)$$

$$Z_k = \frac{1}{1 + e^{Z\_in_k}} \quad (2.9)$$

- Langkah 6
  Setiap *neuron* pada *output layer* $(Z_k, k = 1, 2, \dots, r)$ menerima target pola yang berhubungan dengan pola *neuron* pada *input layer*. Kemudian, hitung informasi *error* dengan Persamaan (2.10). Selanjutnya, hitung selisih bobot dengan Persamaan (2.11), serta selisih bias dengan Persamaan (2.12).

$$\delta\_out_k = (t_k - Z_k) f'(Z\_in_k) \quad (2.10)$$

$$\Delta w\_ho_{jk} = \alpha \delta\_out_k Y_j \quad (2.11)$$

$$\Delta b\_out_k = \alpha \delta\_out_k \quad (2.12)$$

- Langkah 7
  Setiap *neuron* pada *hidden layer* $(Y_j, j = 1, 2, \dots, q)$ menjumlahkan perkalian nilai *gradient descent* dengan bobot seperti pada Persamaan (2.13). Kemudian hitung selisih bobot dan bias dengan Persamaan (2.14) dan Persamaan (2.15).

$$\delta\_hid_j = \left(\sum_{k=1}^{r} \delta\_out_k w\_ho_{jk} \right) f'(Y\_in_j) \quad (2.13)$$

$$\Delta w\_ih_{ij} = \alpha \delta\_hid_j Y_j \quad (2.14)$$

$$\Delta b\_hid_j = \alpha \delta\_hid_j \quad (2.15)$$

- Langkah 8
  Setiap *neuron* pada *output layer* $(Z_k, k = 1, 2, \dots, r)$ perbarui bias dan bobotnya $(j = 1, 2, \dots, q)$ dengan Persamaan (2.16) dan Persamaan (2.17). Untuk setiap *neuron* pada *hidden layer* $(Y_j, j = 1, 2, \dots, p)$ perbarui bias dan bobotnya $(i = 1, 2, \dots, p)$ dengan Persamaan (2.18) dan Persamaan (2.19).

$$w\_ho_{jk}(baru) = w\_ho_{jk}(lama) + \Delta w\_ho_{jk} \quad (2.16)$$

$$b\_out_k(baru) = b\_out_k(lama) + \Delta b\_out_k \quad (2.17)$$

$$w\_ih_{ij}(baru) = w\_ih_{ij}(lama) + \Delta w\_ih_{ij} \quad (2.18)$$

$$b\_hid_j(baru) = b\_hid_j(lama) + \Delta b\_hid_j \quad (2.19)$$

- Langkah 9
  Hitung *MSE* dengan Persamaan (2.20).

$$MSE = \frac{\sum_{k=1}^{r} (t_k - Z_k)^2}{i} \quad (2.20)$$

2.6. Perhitungan Kinerja Sistem

Perhitungan kinerja sistem ini adalah dengan menggunakan akurasi. Akurasi digunakan untuk mengukur tingkat kualitas keberhasilan klasifikasi *tweet* pada tahap pengujian. Kategori uji coba dinyatakan benar jika hasil prediksi tahap pengujian sama

dengan target dari data *ground truth*. Sedangkan kategori uji coba dinyatakan salah jika hasil prediksi tahap pengujian tidak sama dengan target dari data *ground truth*. Persamaan untuk pengukuran akurasi dapat dilihat pada Persamaan (2.21). Perhitungan akurasi akan dilakukan pada kedua tahap yakni, tahap pelatihan dan tahap pengujian untuk mengetahui bahwa pada tahap pengujian sistem telah belajar dan mengenali pola data dengan baik.

$$akurasi = \frac{uji\ coba\ benar}{uji\ coba\ benar + uji\ coba\ salah} \tag{2.21}$$

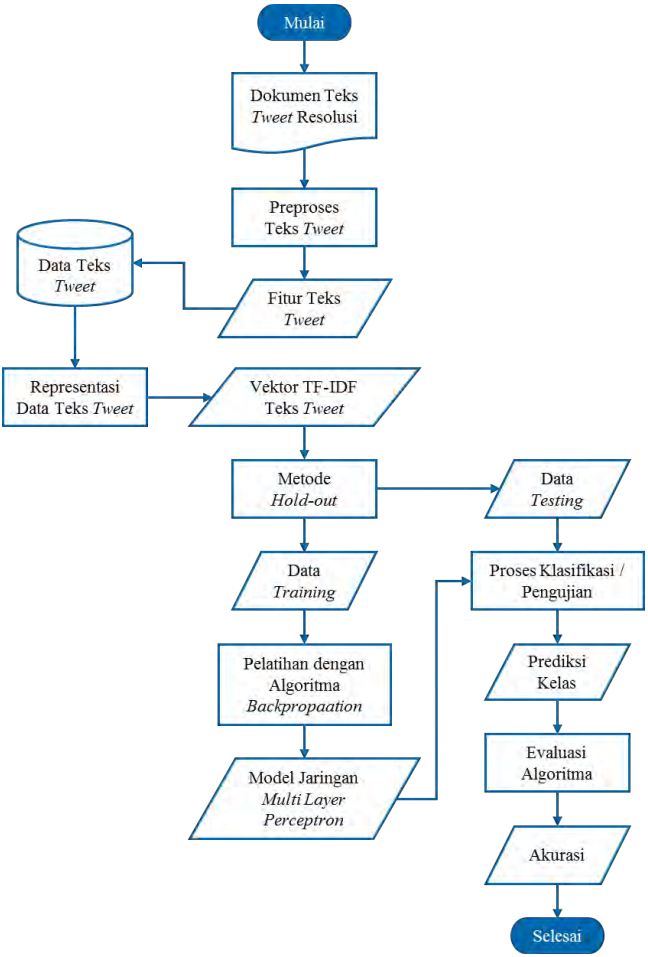
[Halaman ini sengaja dikosongkan]

BAB III
ANALISIS DAN PERANCANGAN

Pada Bab 3 ini akan dijelaskan mengenai analisis dan perancangan perangkat lunak untuk mencapai tujuan dari Tugas Akhir. Perancangan ini meliputi perancangan data, perancangan proses, dan perancangan antar muka, serta juga akan dijelaskan tentang analisis implementasi metode secara umum pada sistem.

3.1. Analisis Implementasi Metode Secara Umum

Pada Tugas Akhir ini akan dibangun sebuah sistem untuk melakukan klasifikasi *tweet* resolusi menggunakan Algoritma *Backpropagation* dengan model Jaringan Saraf Tiruan *Multi-Layer Perceptron*, serta menerapkan metode *supervised learning*. Proses-proses yang terlibat di dalam implementasi sistem ini meliputi tahap praposes teks, tahap pelatihan (*training*), tahap pengujian (*testing*), dan tahap evaluasi kinerja sistem. Tahap praproses teks merupakan tahap pertama yang dilakukan untuk mengolah data masukan sebelum memasuki tahap utama, yaitu tahap *training*. Praproses teks dilakukan dengan tujuan penyeragaman dan pembersihan elemen-elemen kata yang tidak diperlukan dan tidak bermakna. Pada tahap ini, terdapat tiga subproses, yaitu *case folding* untuk mengubah semua huruf dalam dokumen menjadi huruf kecil (*lower case*), *tokenizing* atau tokenisasi untuk memotong teks berdasarkan setiap kata yang penyusunnya, *filtering* atau penyaringan untuk mengambil kata-kata penting hasil tokenisasi, dan *analyzing* atau analisa untuk menentukan seberapa jauh hubungan antarkata dalam dokumen dan antardokumen. Nilai perhitungan pada tahap praproses teks kemudian akan digunakan sebagai data set untuk tahap *training*, maupun tahap *testing*. Untuk selanjutnya, hasil praproses teks akan disebut sebagai fitur dan banyak data merupakan dokumen. Sebelum masuk pada tahap *training*, data set akan dibagi menjadi bagian, yaitu bagian data *training* dan data *testing*. Diagram alir keseluruhan sistem dapat dilihat pada Gambar 3.1.



Gambar 3.1 Diagram Alir Implementasi Sistem

Pada tahap *training*, Algoritma *Backpropagation* diberikan sejumlah data *training*, yang telah melalui praproses teks di awal, di mana terdiri dari nilai pembobotan fitur dan nilai target *output*. Untuk masing-masing vektor dokumen pada data *training* akan menjadi masukan awal jaringan pada tahap *training*. Setelah

seluruh dokumen melalui tahap *training*, maka pada akhir tahap ini akan didapat model jaringan yang telah "pintar" dan mengenali pola data. Sehingga, selanjutnya model jaringan ini mampu memberikan prediksi kelas terhadap data *testing* yang masuk pada tahap *testing*. Pada tahap *testing*, *output* yang dihasilkan adalah prediksi kelas terhadap data dokumen *testing*.

Keluaran pada tahap *testing* akan dibandingkan dengan nilai target *output* yang telah disimpan di awal sebagai sebuah data set utuh. Apabila nilai *output* atau prediksi sama dengan nilai target, maka hasil prediksi dihitung sebagai nilai benar, dan sebaliknya jika hasil prediksi tidak sama dengan nilai salah maka akan dihitung sebagai nilai salah. Pada akhir proses pembandingan, dapat dihitung nilai akurasi prediksi dengan data *ground truth*.

3.2. Perancangan Data

Pada subbab ini akan dibahas mengenai perancangan data yang merupakan bagian penting karena data sebagai objek yang akan diolah oleh sistem dalam Tugas Akhir ini dan menghasilkan sebuah informasi. Data yang akan digunakan pada sistem ini adalah data masukan (*input*), data proses, dan data keluaran (*output*) yang memberikan hasil pengolahan sistem ini untuk pengguna.

3.2.1. Data Masukan

Data masukan merupakan data awal yang akan diproses oleh sistem untuk melakukan pelatihan dan menghasilkan model jaringan yang akan digunakan pada tahap pengujian. Data masukan tersebut berupa kumpulan dokumen teks *tweet* diambil dari twitter berdasarkan kata kunci *hashtag*, yakni *#newyearsresolution*. Seluruh dokumen *tweet* akan disimpan dalam bentuk berkas berekstensi *.xls* untuk mempermudah proses *import* data ke dalam sistem. Di mana data akan selanjutnya dikirim ke dalam basis data sebagai media penyimpanan sebelum melalui tahapan-tahapan pada sistem. Berikut ini adalah contoh teks *tweet* yang digunakan sebagai data masukan.

- *Self improvement!.. Mentally, physically, and financially. #NewYearsResolution*
- *social media purging > #NewYearsResolution*
- *to actually work out and not be lazy about it #NewYearsResolution*

3.2.2. Data Proses

Data proses adalah data yang digunakan selama proses berjalannya sistem yang merupakan hasil pengolahan dari data masukan untuk diproses kembali menjadi data keluaran di tahap selanjutnya. Data proses yang dimaksud adalah data masukan yang telah tersimpan di dalam basis data dan menyimpan nilai pembobotan fitur serta nilai target masing-masing dokumen. Pembobotan fitur dilakukan secara otomatis di dalam basis data setelah data masukan tersimpan secara utuh. Implementasi pembobotan fitur akan menggunakan Algoritma TF-IDF yang akan dibahas pada subbab 3.3.2.

Tabel 3.1 Data Proses Matrix TF-IDF

		fitur ke-						
		6	7	8	9	10	11	12
dokumen ke-	1	0	2,21	2,91	2,91	2,91	3,03	0
	72	0	2,21	2,91	0	0	0	0
	475	0	0	0	2,91	2,91	0	0
	1150	0	0	0	0	0	0	1,60
	1817	0	0	0	0	0	3,03	0
	2341	0	0	0	2,91	2,91	0	0
	3159	0	0	0	0	2,91	0	0

Setelah data keseluruhan diambil dari basis data, data akan disimpan dalam bentuk matriks selama sistem berlangsung. Contoh matriks dapat dilihat pada Tabel 3.1. Matriks tersebut menjelaskan besar nilai fitur pada suatu dokumen. Pada matriks ini dapat diketahui bahwa fitur ke-7 pada dokumen ke-72 memiliki nilai bobot sebesar 2,21; dan untuk fitur ke-11 pada dokumen ke-1817 memiliki nilai bobot sebesar 3,03. Sedangkan nilai bobot

sebesar 0 menujukkan bahwa suatu dokumen tidak memiliki fitur tertentu.

3.2.3. Data Keluaran

Data keluaran dari sistem ini adalah berupa informasi evaluasi kinerja sistem melalui hasil tahap *training* dan tahap *testing*. Untuk menentukan hasil *testing*, model jaringan hasil tahap *training* diaplikasikan pada data *training* untuk menghasilkan prediksi kelas untuk masing-masing vektor dokumen masukan. Beberapa data yang ditampilkan adalah jumlah data dan fitur, maksimum *epoch*, *learning rate*, akurasi tahap *training* dan *testing*, serta *running time*.

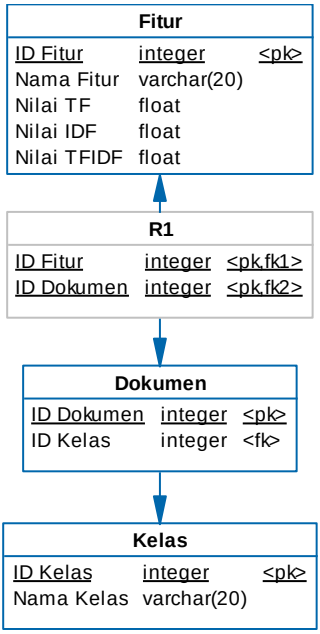
Tabel 3.2 Data Keluaran Sistem Klasifikasi *Tweet*

Hasil		Nilai
Data	:	3914
Fitur	:	4551
Maks. *Epoch*	:	10
Learning Rate	:	0,05
- Hasil Pelatihan -		
Waktu	:	435876 milisekon
MSE	:	0,06690
Benar	:	3301
Salah	:	222
Akurasi	:	0,93698
- Hasil Pengujian -		
Benar	:	362
Salah	:	29
Akurasi	:	0,92583

Pada Tabel 3.2, Data mewakili jumlah dokumen yang masuk sebagai *input*. Dokumen yang dimaksud merupakan keseluruhan data yang diambil dari basis data sebelum dibagi menjadi data *training* dan data *testing*. Fitur merupakan banyak teks yang dihasilkan dari tahap praproses teks. Maks. *Epoch* yakni jumlah iterasi yang ingin diaplikasikan untuk melakukan *training*.

Waktu yang dimaksud merupakan waktu yang dibutuhkan sistem selama melakukan *training*. MSE merupakan besar *error* yang dihitung pada setiap akhir tahap *training*. Benar dan Salah secara berturut-turut merupakan jumlah nilai benar dan salah hasil perbandigan nilai *output* dengan nilai target.

3.2.4. Basis Data



Gambar 3.2 Rancang Tabel pada Basis Data

Pada Tugas Akhir ini akan dilakukan perancangan basis data sebagai media penyimpanan fitur, yakni yang berupa teks. Pada basis data, setiap fitur akan mengalami proses pembobotan untuk mendapatkan nilai masing-masing fitur agar dapat diolah dalam system klasifikasi *tweet* resolusi. Rancangan basis data yang dibangun meliputi tabel untuk menyimpan hasil pembobotan dan tabel untuk menyimpan kelas klasifikasi yang didapat dari data masukan. Untuk rancangan tabel dalam basis data dapat dilihat

pada Gambar 3.2. Ditunjukkan pada gambar, terdapat tiga jenis tabel yang dirancang, yakni Tabel Fitur, Tabel Dokumen, dan Tabel Kelas. Untuk penjelasan mengenai atribut tabel, dapat dilihat pada Tabel 3.3.

Tabel 3.3 Keterangan Atribut Rancangan Basis Data

No	Nama Atribut	Keterangan
1	ID Fitur	Menyimpan *primary key* Tabel Fitur.
2	Nama Fitur	Menyimpan nama fitur atau kata yang masuk untuk setiap dokumen.
3	Nilai TF	Menyimpan nilai tf untuk fitur dimaksud.
4	Nilai IDF	Menyimpan nilai idf untuk fitur dimaksud.
5	Nilai TFIDF	Menyimpan hasil kali nilai tf dan idf untuk fitur dimaksud.
6	ID Dokumen	Menyimpan *primary key* Tabel Dokumen.
7	ID Kelas	Menyimpan *primary key* Tabel Kelas.
8	Nama Kelas	Menyimpan nama kelas yang masuk untuk setiap dokumen.

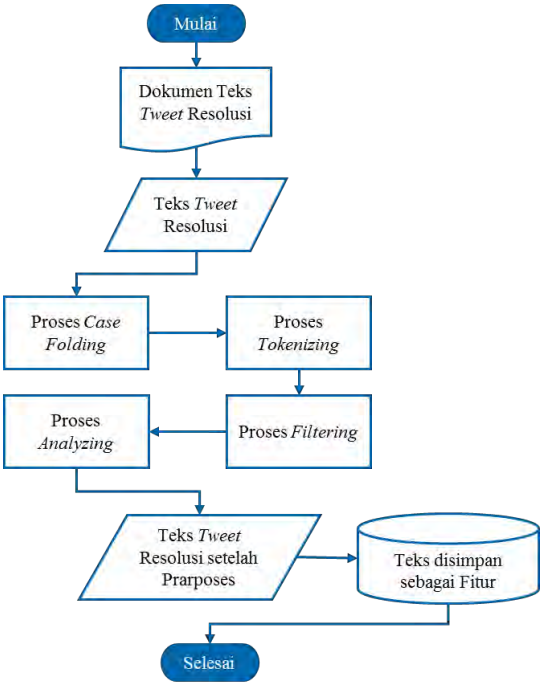
3.3. Perancangan Proses

Pada subbab ini akan dibahas mengenai perancangan proses yang dilakukan untuk memberikan gambaran secara rinci pada setiap alur implementasi metode pada. Alur tersebut nantinya akan digunakan dalam tahap implementasi.

3.3.1. Praproses Teks

Tahap praproses teks merupakan tahap paling awal sebelum memulai tahap lain yang akan diaplikasikan pada sistem ini. Praproses teks berperan penting dan sangat mempengaruhi hasil pada tahap selanjutnya. Hasil praproses teks yang baik mampu meningkatkan performansi sistem. Pada Tugas Akhir ini, tahap praproses teks terdiri dari empat proses utama, yakni *case*

folding (mengubah teks menjadi *lower case*), *tokenizing* (pemotongan teks berdasarkan kata penyusunnya), *filtering* (pemilihan kata-kata yang memiliki makna, dan membuang yang tidak), dan *analyzing* (penentuan hubungan antarkata dalam dokumen dan antardokumen). Dalam hal ini, teks merupakan bentuk dari sebuah dokumen *tweet*. Data yang masuk dalam sistem adalah berupa berkas berekstensi *.xls* yang berisi kumpulan dokumen teks *tweet* sebagai data masukan.



Gambar 3.3 Diagram Alir Praproses Teks

Masing-masing dokumen *tweet* pada Tugas Akhir ini memiliki maksimal 140 karakter, baik huruf, angka, maupun bentuk karakter lainnya. Seperti yang ditunjukkan pada Gambar 3.3, setelah masing-masing dokumen teks melalui tahap praproses dari proses *case folding* hingga proses *analyzing*, setiap teks akan

disimpan ke dalam basis data sebagai fitur. Dari poin ini hingga tahapan selanjutnya, potongan-potongan teks ini akan disebut sebagai fitur.

3.3.1.1. Proses Case Folding

Proses *case folding* merupakan proses pertama yang dilakukan pada tahap preproses teks. Proses ini merupakan sebuah proses untuk mengubah format seluruh teks dalam dokumen menjadi huruf kecil (*lower case*), tanpa huruf kapital. Proses ini bertujuan untuk menghindari ambiguitas dua teks yang sama namun berbeda format.

Contoh hasil proses *case folding* ditunjukkan Tabel 3.4. Pada kolom masukan contoh nomor 1, kata *Self* memiliki huruf *S* dalam bentuk kapital. Setelah melalui proses *case folding* bentuk huruf kapital berubah menjadi huruf kecil. Kata *Self* berubah menjadi *self*. Sama halnya seperti penjelasan sebelumnya, kata *#NewYearsResolution* berubah menjadi *#newyearsresolution*. Huruf *N*, *Y*, dan *R* yang awalnya memiliki bentuk kapital diubah menjadi huruf kecil (*lower case*).

Tabel 3.4 Hasil Proses *Case Folding*

No	Masukan	Keluaran
1	*Self improvement!.. Mentally, physically, and financially. #NewYearsResolution*	*self improvement!.. mentally, physically, and financially. #newyearsresolution*
2	*social media purging > #NewYearsResolution*	*social media purging > #newyearsresolution*
3	*to actually work out and not be lazy about it #NewYearsResolution*	*to actually work out and not be lazy about it #newyearsresolution*

3.3.1.2. Proses Tokenizing

Proses *tokenizing* merupakan proses pemotongan teks berdasar kata penyusunnya. Pada tahap ini, teks akan dipotong atau dipisah berdasarkan letak spasi antarteks. Kata-kata yang telah berdiri secara independen sebagai sebuah teks disebut *token*.

Selain itu, proses ini juga akan menghilangkan karakter-karakter selain huruf, sehingga akan didapat kata yang murni hanya memiliki huruf di dalamnya dan dapat dibaca sebagai satu kesatuan. Proses ini bertujuan untuk membagi teks menjadi fitur. Contoh hasil proses *tokenizing* ditujukkan Tabel 3.5.

Tabel 3.5 Hasil Proses *Tokenizing*

No	Masukan	Keluaran
1	*self improvement!.. mentally, physically, and financially. #newyearsresolution*	*self* *improvement* *mentally* *physically* *and* *financially* *newyearsresolution*
2	*social media purging > #newyearsresolution*	*social* *media* *purging* *gt* *newyearsresolution*
3	*to actually work out and not be lazy about it #newyearsresolution*	*to* *actually* *work* *out* *and* *not* *be* *lazy* *about* *it* *newyearsresolution*

Hasil proses *case folding* pada penjelasan subbab 3.3.1.1, akan digunakan sebagai masukan pada proses ini. Pada Tabel 3.5 kolom masukan nomor 2, teks awal yang bertuliskan *social media purging > #newyearsresolution* dipisah berdasarkan letak spasi

sehingga berdiri secara independen sebagai sebuah kata. Kata yang dihasilkan adalah *social*, *media*, *purging*, *>*, dan *newyearsresolution*. Seperti yang dijelaskan sebelumnya, selain memisahkan kata berdasarkan letak spasi antarteks, proses ini juga akan menghilangkan karakter-karakter tidak penting, yakni selain huruf. Sehingga di akhir proses ini, kata yang mengandung karakter selain huruf seperti *>* dan *#newyearsresolution* akan berubah menjadi *gt* dan *newyearsresolution*.

3.3.1.3. Proses *Filtering*

Tabel 3.6 Hasil Proses *Filtering*

No	Masukan	Keluaran
1	*self* *improvement* *mentally* *physically* *and* *financially* *newyearsresolution*	*self* *improvement* *mentally* *physically* *financially*
2	*social* *media* *purging* *gt* *newyearsresolution*	*social* *media* *purging*
3	*to* *actually* *work* *out* *and* *not* *be* *lazy* *about* *it* *newyearsresolution*	*work* *lazy*

Proses *filtering* merupakan proses selanjutnya pada tahap praproses teks. Proses ini dilakukan karena teks hasil proses *tokenizing* masih mengandung kata-kata yang tidak bermakna,di mana dapat menjadi hambatan dalam menyimpulkan isi teks. Misalnya, kata penghubung seperti '*and*' dan '*but*'. Selain itu, kata yang memiliki jumlah karakter kurang dari tiga juka akan dihilangkan. Untuk itu pada proses *filtering* akan diplih kata-kata yang memiliki makna saja, sehingga kata-kata tersebut saja yang digunakan sebagai fitur pada proses selanjutnya. Contoh proses *filtering* ditunjukkan Tabel 3.6.

Ditunjukkan pada Tabel 3.6 kolom masukan nomor 3, dari sekian banyak kata masukan yang dihasilkan dari proses sebelumnya, yakni proses *tokenizing*, hanya tersisa dua kata yang dianggap penting dalam sebuah dokumen tersebut. Kata *newyearsresolution* juga termasuk kata tidak penting karena seluruh dokumen mengadung kata ini, di mana didapat dari kata kunci *hashtag*. Sebuah kata yang terlalu sering muncul pada setiap dokumen dapat menyebabkan pembobotan kurang menjadi valid.

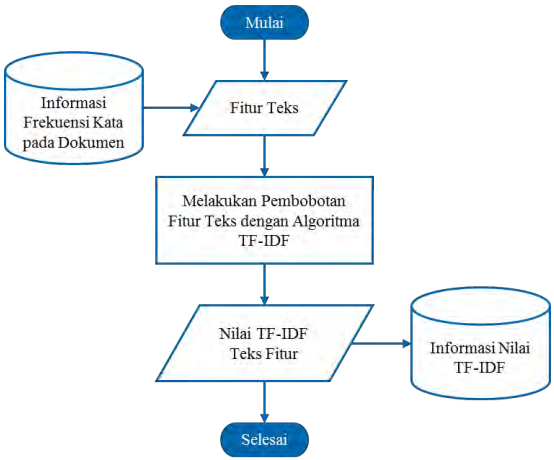
3.3.1.4. Proses *Analyzing*

Tabel 3.7 Hasil Proses *Analyzing*

No	Kata	Frekuensi Kata pada Dokumen
1	*self*	1
2	*improvement*	1
3	*mentally*	1
4	*physically*	1
5	*financially*	1
6	*social*	1
7	*media*	1
8	*purging*	1
9	*work*	1
10	*lazy*	1

Proses *analyzing* bertujuan untuk menganalisa seberapa sering sebuah kata digunakan untuk topik tertentu. Selain itu, juga untuk menghitung seberapa sering kata tersebut digunakan dalam sebuah dokumen. Informasi ini akan digunakan sebagai bahan untuk merepresentasi data teks *tweet* pada tahap selanjutnya. Contoh proses *analyzing* ditunjukkan Tabel 3.7. Pada tabel jelas diperlihatkan berapa kali sebuah kata muncul pada sebuah dokumen. Informasi ini selanjutnya akan disimpan ke dalam basis data, yang kemudian digunakan untuk menghitung keterkaitan sebuah kata atau fitur dengan dokumen yang mengandung kata tersebut, serta dokumen lainnya. Proses pembobotan masing-masing kata atau fitur akan dibahas pada subbab 3.3.2.

3.3.2. Representasi Fitur Teks



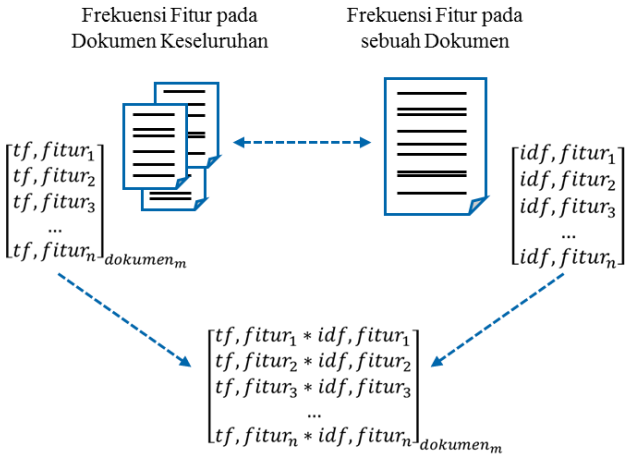
Gambar 3.4 Diagram Alir Representasi Fitur Teks

Tahapan ini adalah persiapan sebelum memasuki tahap utama, yaitu tahap pelatihan dengan Algoritma *Backpropagation*. Proses pengolahan fitur dalam membangun sistem klasifikasi teks merupakan suatu bentuk *text mining* atau pengolahan data teks. Pada tahap inilah dokumen *tweet* menjadi sumber data akan mengalami perubahan bentuk dan *format*. Setiap dokumen *tweet*

direpresentasikan atau digambarkan sebagai sebuah vektor yang berisi kata-kata dengan nilai hubungan antarkata dalam dokumen dan antardokumen, serta nilai bobot kepentingannya masing-masing. Pembobotan menggunakan prinsip dan rumus Algoritma TF-IDF, yang akan dijelaskan pada subbab 3.3.2.1. Diagram alir proses representasi fitur teks ditunjukkan pada Gambar 3.4.

Hasil tahap praproses yang berupa informasi frekuensi masing-masing fitur pada setiap di dokumen merupakan masukan utama dalam tahap ini. Informasi tersebut selanjutnya digunakan untuk mengetahui seberapa sering kata tersebut muncul pada keseluruhan dokumen. Nilai-nilai frekuensi inilah yang akan digunakan dalam perhitungan pada subbab proses selanjutnya.

3.3.2.1. Pembobotan Fitur Teks



Gambar 3.5 Pembobotan Fitur Kata Teks *Tweet*

Proses representasi teks dengan pembobotan TF-IDF sangat mempengaruhi hasil pada tahap *training*, karena teks yang sifatnya masih kualitatif akan melalui perhitungan untuk melakukan klasifikasi. Dengan pertimbangan inilah dibentuk vektor representasi teks dengan Algoritma TF-IDF untuk

mendapatkan nilai kuantitatif tersebut. Nilai TF-IDF inilah yang nantinya akan menjadi faktor utama dalam keberhasilan proses klasifikasi teks. Gambaran penghitungan pembobotan ditunjukkan pada Gambar 3.5.

Tabel 3.8 Perhitungan Frekuensi Fitur Teks pada Dokumen

No	Fitur	Dokumen ke-				
		1	2	3	4	5
1	*beijing*	0	1	0	0	3
2	*dish*	0	1	0	0	1
3	*duck*	3	2	2	0	1
4	*rabbit*	0	0	1	1	0
5	*recipe*	0	0	1	1	1
6	*roast*	0	0	0	0	0

Seperti yang ditunjukkan pada Gambar 3.5, sebuah keseluruhan data masukan teks terdiri dari dua komponen, yakni fitur dan dokumen. Dalam hal ini, fitur merupakan kata yang telah diperoleh dari tahap praproses teks sebelum tahap representasi fitur teks dilakukan. Sedangkan, dokumen merupakan *record* sebuah *tweet*, jadi satu baris *tweet* dapat dihitung sama dengan satu dokumen.

Tabel 3.9 Perhitungan Normalisasi Nilai Frekuensi Fitur

No	Fitur	Dokumen ke-					IDF
		1	2	3	4	5	
1	*beijing*	0,00	0,50	0,00	0,00	1,00	0,39
2	*dish*	0,00	0,50	0,00	0,00	0,33	0,39
3	*duck*	1,00	1,00	1,00	0,00	0,33	0,09
4	*rabbit*	0,00	0,00	0,50	1,00	0,00	0,39
5	*recipe*	0,00	0,00	0,50	1,00	0,33	0,22
6	*roast*	0,00	0,00	0,00	0,00	0,00	0,00

Tabel 3.8 menunjukkan seberapa sering masing-masing fitur teks muncul pada setiap dokumen. Pada tabel tersebut dijelaskan bahwa fitur teks *beijing* muncul sebanyak lima kali pada

dokumen ke-5, sedangkan pada dokumen ke-2 hanya muncul sebanyak satu kali. Seluruh informasi terkait frekuensi teks telah disimpan dalam basis data, berdasarkan penjelasan sebelumnya.

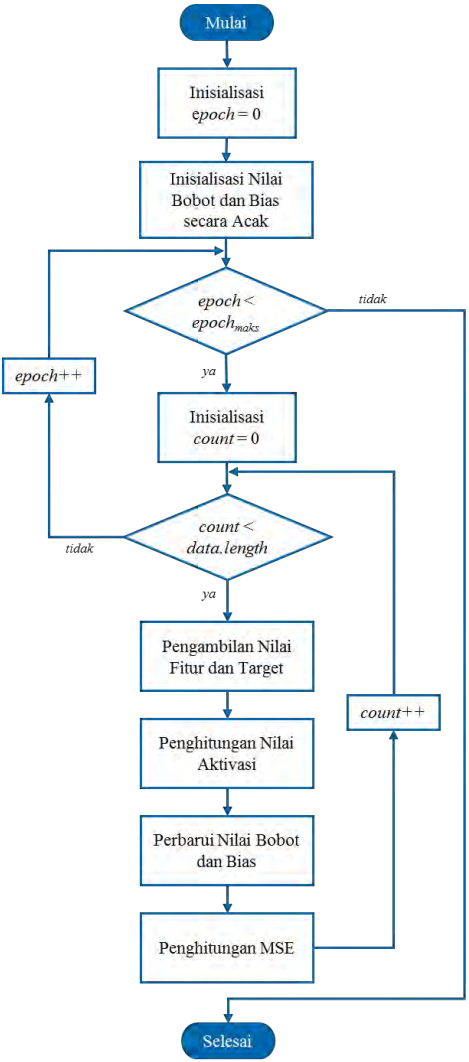
Sedangkan, pada Tabel 3.9 menunjukkan nilai hasil normalisasi terhadap nilai pada Tabel 3.8. Pada akhir tabel ditemukan nilai IDF, di mana merupakan hasil perhitungan dengan Persamaan (2.2) pada subbab 2.3.

Tabel 3.10 Hasil Pembobotan TF-IDF Fitur Teks

No	Fitur	Dokumen ke-				
		1	2	3	4	5
1	*beijing*	0,00	0,19	0,00	0,00	0,39
2	*dish*	0,00	0,19	0,00	0,00	0,13
3	*duck*	0,09	0,09	0,09	0,00	0,03
4	*rabbit*	0,00	0,00	0,19	0,39	0,00
5	*recipe*	0,00	0,00	0,11	0,22	0,07
6	*roast*	0,00	0,00	0,00	0,00	0,00

Tabel 3.10 menunjukkan nilai TF-IDF untuk masing-masing fitur pada setiap dokumen. Nilai fitur pada setiap dokumen berbeda. Fitur *recipe* pada dokumen ke-4 memiliki nilai 0,22; sedangkan pada dokumen ke-5 memiliki nilai sebesar 0,07. Nilai TF-IDF ini didapatkan dari hasil perkalian nilai TF pada Tabel 3.8 dengan nilai IDF pada Tabel 3.9. Nilai inilah yang nantinya menjadi nilai awal fitur ketika masuk ke dalam jaringan untuk melakukan tahap *training*. Tahap Pelatihan (*Training*)

Tahap *training* merupakan tahapan ini pada Tugas Akhir ini, di mana sistem akan belajar untuk mengenali pola pada data untuk membuat model jaringan yang sesuai. Tahap *training* memiliki tujuan untuk meminimalkan *error* pada nilai *output* yang dihasilkan oleh jaringan. Pada tahap inilah Algoritma *Backpropagation* mulai bekerja untuk mempelajari data. Untuk diagram alir Algoritma *Backpropagation* dapat dilihat pada Gambar 3.6.



Gambar 3.6 Diagram Alir Sistem Tahap *Training*

Yang pertama kali dilakukan sistem saat menjalankan Algoritma *Backpropagation* adalah melakukan inisialisasi nilai

epoch menjadi nol. *Epoch* merupakan banyak iterasi yang diperlukan untu melakukan *training* pada data. Selain itu, perlu juga ditentukan nilai maksimal *epoch* untuk menentukan berapa kali iterasi dilakukan dan kapan *training* harus berhenti. Selanjutnya seperti pada *pseudocode* yang ditunjukkan pada Gambar 3.7, selanjutnya sistem akan masuk ke dalam perulangan di mana batas akhirnya merupakan nilai maksimum *epoch*.

Algoritma 1 Algoritma Backpropagation

```
 1:  function backpropagation()
 2:   inisialisasi nilai max_epoch
 3:   inisialisasi nilai bobot dan bias secara acak
 4:   epoch ← 0
 5:   mse ← 1
 6:   while epoch < max_epoch
 7:     for i ← 0 to datatrain.length do
 8:       input ← vektor fitur datatrain ke-i
 9:       target ← vektor target datatrain ke-i
10:       hitung nilai aktivasi pada output layer
11:       perbarui nilai bobot dan bias
12:     end for
13:     hitung mse
14:     epoch ← epoch + 1
15:   end while
16:   return model
17:  end function
```

Gambar 3.7 *Pseudocode* Algoritma *Back Propagation*

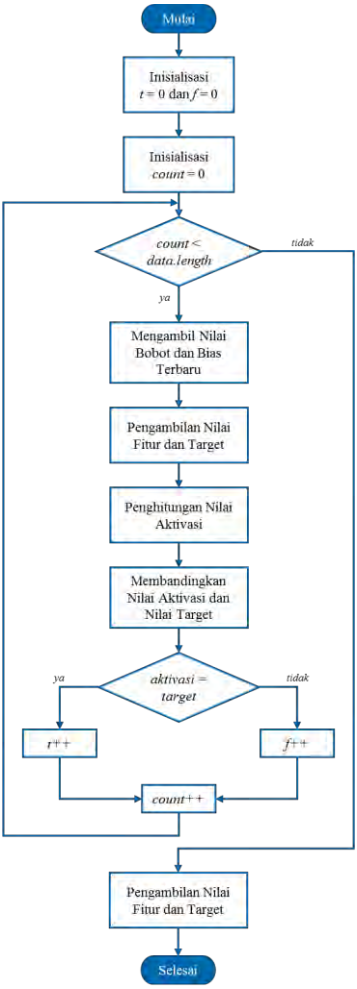
Setiap satu perulangan proses *training*, beberapa proses akan dilakukan oleh sistem. Namun, sebelum itu, sistem akan masuk ke perulangan lain yaitu sebanyak dokumen yang dimiliki oleh data *training*. Di antaranya, yang pertama adalah memisahkan fitur masukan dengan nilai target dan menyimpannya ke dalam variabel berbeda, seperti yang ditunjukkan baris ke-8 dan baris ke-9 pada Gambar 3.7. Selanjutnya, *backpropagation* akan menghitung nilai aktivasi pada *output layer*. Dan, menentukan seberapa besar *error* dengan membandingkan nilai aktivasi tersebut dengan nilai target yang disimpan sebelumnya. Setelah mendapatkan nilai *error* tersebut, *backpropagation* akan memperbarui bobot dan bias dengan mengurangi nilai awal dengan nilai sebesar persentase *error* yang telah dihitung. Proses ini terus

berulang hingga seluruh dokumen melakukan *training*. Satu proses hingga seluruh dokumen melalui *training* ini dihitung sebagai satu *epoch*. Selanjutnya, sistem akan mengulangi seluruh proses yang dilakukan hingga nilai *epoch* mencapai maksimum.

Secara keseluruhan, Algoritma *Backpropagation* dibagi menjadi dua fase, yakni Fase Propagasi dan Fase Perbarui Bobot. Secara mendetail perhitungan keseluruhan Algoritma *Backpropagation* telah dijelaskan dan dapat dilihat pada subbab 2.5.1. Yang termasuk pada fase propagasi adalah proses yang dilakukan sejak algoritma atau metode ini pertama kali dijalankan hingga mendapat nilai *error* terhadap perbandingan nilai *output* dan nilai target. Fase perbarui bobot merupakan proses di mana *training* sebuah vektor dokumen telah dilakukan dan bobot harus diperbarui berdasarkan besar *error*. Di mana bobot sangat penting untuk diperbarui karena akan digunakan pada *training* selanjutnya dengan dokumen masukan yang sama hingga *epoch* mencapai nilai maksimum.

3.3.3. Tahap Pengujian (*Testing*) dan Evaluasi

Secara umum langkah-langkah pada tahap *testing* tidak jauh berbeda dengan tahap *training*. Pada tahap ini juga akan mengambil nilai vektor fitur dan target dari data *testing*, sama seperti tahap *training*. Kemudian, menghitung nilai aktivasi dari masing-masing vektor, yang mana juga sama dengan tahap *training*. Yang membedakan dengan tahap *training* adalah pada tahap *testing* tidak dilakukan inisialisasi bobot di awal proses. Nilai bobot diambil dari nilai bobot terbaru pada akhir tahap *training*. Hal inilah yang membawa model jaringan yang sudah "pintar" untuk melakukan prediksi pada data *testing*. Selain itu, pada tahap *testing* hanya dilakukan satu kali iterasi atau *epoch*. Jadi, dengan nilai bobot terbaru dan model jaringan yang sama, sistem akan menghitung nilai aktivasi masing-masing dokumen. Detail diagram alir tahap *testing* ditunjukkan pada Gambar 3.8.



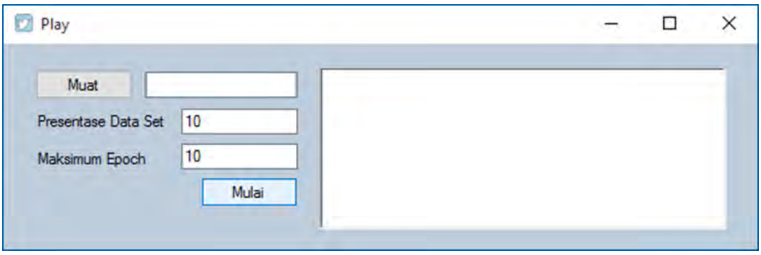
Gambar 3.8 Diagram Alir Sistem Tahap *Testing*

Setelah nilai aktivasi dihasilkan pada proses sebelumnya, yang selanjutnya dilakukan adalah membandingkan nilai aktivasi dengan nilai target yang telah disimpan pada awal tahap. Pada proses ini dilakukan pengecekkan benar dan salah. Jika nilai

aktivasi sama dengan nilai target, maka prediksi diartikan benar. Sedangkan jika nilai aktivasi tidak sama dengan nilai target, maka dianggap salah. Seluruh nilai benar dan salah dari seluruh prediksi data *testing* akan disimpan hingga akhir tahap *testing*. Agar supaya di akhir tahap *testing* dapat dihitung persentase akurasi sistem dalam melakukan prediksi terhadap klasifikasi kelas.

3.4. Perancangan Antarmuka Pengguna

Pada subbab ini akan dibahas mengenai perancangan antarmuka perangkat lunak yang bertujuan untuk dapat mempermudah interaksi antara perangkat lunak dengan pengguna. Rancangan antarmuka pengguna ditunjukkan oleh Gambar 3.9.



Gambar 3.9 Rancangan Antarmuka Pengguna

Antarmuka pengguna dibuat sesederhana mungkin agar pengguna tidak kesulitan dalam menggunakan sistem. Pada Gambar 3.9 terdapat *button* dan beberapa baris teks yang dapat diisi oleh pengguna. Tombol Muat digunakan untuk meng-*import* data *tweet* awal yang disimpan dalam bentuk berkas berekstensi *.xls*. Baris Presentase Data Set adalah untuk menentukan perbandingan data *training* dan data *testing* yang akan digunakan untuk klasifikasi teks. Sedangkan Maksimum *Epoch* adalah baris untuk menentukan berapa maksimum iterasi yang akan digunakan dalam tahap *training* dengan Algoritma *Backpropagation*. Kemudian, tombol Mulai adalah tombol untuk menjalankan tahap *training* serta tahap *testing* secara berurutan. Kotak kosong yang berukuran hampir setelah tampilan merupakan *space* untuk menampilkan hasil evaluasi kinerja klasifikasi sistem.

[Halaman ini sengaja dikosongkan]

BAB IV
IMPLEMENTASI

Bab ini membahas implementasi dari perancangan sistem sesuai dengan perancangan yang telah dibuat. Bahasa pemrograman yang digunakan untuk implementasi sistem adalah bahasa pemrograman C# dengan Kerangka Kerja .NET.

4.1. Lingkungan Implementasi

Lingkungan implementasi sistem yang digunakan untuk mengembangkan Tugas Akhir memiliki spesifikasi perangkat keras dan perangkat lunak seperti yang ditunjukkan pada Tabel 4.1 lebi detail dan jelas.

Tabel 4.1 Lingkungan Implementasi Sistem

Perangkat	Spesifikasi
Perangkat keras	Prosesor: AMD E-450 APU with Radeon™ HD Graphics 1.65 GHz Memori: 4.00 GB
Perangkat lunak	Sistem Operasi: Microsoft Windows 10 Pro Perangkat Pengembang: Visual Studio Ultimate 2013, SQL Server 2014 Management Studio Perangkat Pembantu: Microsoft Excel 2013, Microsoft PowerPoint 2013, Microsoft Word 2013, Sublime Text 3

4.2. Implementasi Data

Implementasi data dilakukan berdasarkan perancangan data yang sudah dijelaskan pada bab analisis dan perancangan.

4.2.1. Implementasi Basis Data

Implementasi basis data dilakukan agar sistem mampu menyimpan informasi secara permanen hingga data dihapus dari basis data. Terdapat dua buah tabel sebagai hasil implementasi basis data. Tabel pertama adalah Tabel Feature, dapat dilihat pada Gambar 4.1. Tabel Feature berisi seluruh informasi fitur, mulai dari nomor urut dokumen, teks fitur, nilai TF, nilai normalisasi TF, nilai IDF, hingga nilai pembobotan TFIDF.

Column Name	Data Type	Allow Nulls
IDREC	int	☐
IDDOC	int	☐
IDWRD	int	☑
WORDT	varchar(30)	☐
TFVAL	int	☐
TFNOR	float	☑
IDFVL	float	☑
TFIDF	float	☑

Gambar 4.1 Implementasi Tabel Feature pada Basis Data

Gambar 4.2 menunjukkan contoh fitur teks yang masuk dalam basis data. Proses memasukkan fitur ke dalam basis data dilakukan dengan memanggil *query* `INSERT INTO` untuk masing-masing fitur dari kode sumber. *Query* akan dipanggil terus-menerus hingga seluruh dokumen telah dibaca dan menghasilkan fitur setelah melalui tahap praproses teks. Selama proses memasukkan fitur masih berlangsung, secara otomatis nilai perhitungan TF-IDF akan bernilai *null*. Pembobotan TF-IDF akan dilakukan oleh basis data dengan memanggil *stored procedure*, untuk itu perlu seluruh data masuk secara menyeluruh terlebih dahulu sebelum melakukan pembobotan. Karena untuk menentukan pembobotan perlu diketahui total seluruh fitur untuk seluruh dokumen. Gambar 4.2 menunjukkan seluruh fitur telah masuk dan secara otomatis pembobtan TF-IDF telah dilakukan.

Untuk keterangan nama atribut pada tabel dan kegunaannya ditunjukkan pada Tabel 4.2.

	IDREC	IDDOC	IDWRD	WORDT	TFVAL	TFNOR	IDFVL	TFIDF
1	1	0	0	stay	1	1	1.82908185433774	1.82908185433774
2	2	0	1	saaaame	1	1	3.2914798522367	3.2914798522367
3	3	1	2	acting	1	1	2.89353984356466	2.89353984356466
4	4	1	3	mixed	1	1	3.59250984790068	3.59250984790068
5	5	2	4	eat	1	1	1.5796726231955	1.5796726231955
6	6	2	5	healthier	1	1	2.0484418035504	2.0484418035504
7	7	3	6	produce	1	1	2.89353984356466	2.89353984356466
8	8	3	7	druggie	1	1	3.59250984790068	3.59250984790068
9	9	3	8	anthem	1	1	3.59250984790068	3.59250984790068
10	10	4	9	dream	1	1	2.59250984790068	2.59250984790068

Gambar 4.2 *Record* Fitur dalam Tabel Feature

Tabel kedua yang dibangun adalah Tabel Class. Tabel ini bertujuan untuk menyimpan informasi kelas dari suatu dokumen. Informasi ini bertujuan untuk mempermudah penentuan nilai target pada tahap *training*. Informasi yang tersimpan pada tabel ini di antaranya, nomor urut dokumen, urut kelas, serta nama kelas. Untuk desain tabel ditunjukkan pada Gambar 4.3.

Column Name	Data Type	Allow Nulls
IDREC	int	☐
IDDOC	int	☐
IDCLS	int	☐
CLASS	varchar(30)	☐

Gambar 4.3 Implementasi Tabel Class dalam Basis Data

Ketika sistem melakukan tahap praproses, sistem mengambil informasi nomor dokumen serta nomor kelas yang dikirimkan ke dalam basis data. Jadi, dalam satu tahap praproses teks, sistem juga informasi kelas klasifikasi. Dapat dilihat Gambar 4.4 bahwa terdapat atribut kolom yang sama. Hal ini bertujuan agar fitur pada dokumen tertentu dapat mengenali kelas prediksinya.

	IDREC	IDDOC	IDCLS	CLASS
1	1	0	4	Personal Growth
2	2	1	4	Personal Growth
3	3	2	3	Health & Fitness
4	4	3	1	Career & Education
5	5	4	4	Personal Growth
6	6	5	1	Career & Education
7	7	6	3	Health & Fitness
8	8	7	4	Personal Growth
9	9	8	4	Personal Growth
10	10	9	4	Personal Growth

Gambar 4.4 *Record* Kelas dalam Tabel Class

Kedua tabel tersebut digunakan terlebih saat sistem mulai memuat data *tweet* resolusi hingga proses awal tahap *training*. Setelah itu, interaksi antara sistem dan basis data tidak terjadi lagi, karena seluruh informasi di atas akan disimpan dalam variabel yang telah tersedia dalam kode sumber.

Tabel 4.2 Keterangan Atribut Implementasi Basis Data

No	Nama Atribut	Keterangan
1	IDREC	Menentukan *record ID* yang di-*generate* otomatis dari basis data.
2	IDDOC	Menyimpan urutan dokumen yang masuk, dimulai dari indeks nol.
3	IDWRD	Menyimpan urutan fitur yang masuk, dimulai dari indeks nol.
4	WORDT	Menyimpan teks fitur.
5	TFVAL	Menyimpan nilai TF tiap fitur.
6	TFNOR	Menyimpan hasil normalisasi nilai TF.
7	IDFVL	Menyimpan nilai IDF tiap fitur
8	TFIDF	Menyimpan hasil kali nilai pada kolom `TFNOR` dan `IDFVL`.
9	IDCLS	Menyimpan urutan kelas yang masuk.
10	CLASS	Menyimpan nama kelas dari data masukan.

4.3. Implementasi Proses

Implementasi proses dilakukan berdasarkan perancangan proses yang sudah dijelaskan pada bab analisis dan perancangan. Dalam satu *project*, beberapa berkas digunakan dalam pembuatan Tugas Akhir ini sebagai implementasi kode sumber. Untuk lebih jelas mengenai berkas yang digunakan dan kegunaannya, dapat dilihat pada Tabel 4.3.

Tabel 4.3 Keterangan Berkas Implementasi Proses

No	Nama Berkas	Keterangan
1	Play. Designer.cs	Berisi kode sumber untuk membangun antarmuka pengguna. Di antaranya, menginisialisasi *button*, *textboxt*, serta jendela aplikasi.
2	Play. cs	Berisi kode sumber mengenai fungsi-fungsi yang masih berkaitan dengan antarmuka pengguna. Mengeksekusi fungsi jika *button* pada antarmuka pengguna ditekan. Terdiri dari fungsi `muat_Click()`, `tampil_Click()`, `train_Click()`, dan `reset_Click()`.
3	Program.cs	Berisi kode sumber fungsi-fungsi yang dibutuhkan selama sistem berjalan. Menitikberatkan pada fungsi di luar tahap *training*, *testing*, dan evaluasi kinerja sistem. Terdiri dari fungsi `tokenizeTweet()`, `insertClass()`, `insertTF()`, `calculateTFIDF()`, `separateMatrix()`, dan `randomizeOrder()`.

Tabel 4.3 Keterangan Berkas Implementasi Proses (Lanj.)

No	Nama Berkas	Keterangan
4	NeuralNetwork.cs	Berisi kode sumber mengenai keseluruhan fungsi dan elemen yang dibutuhkan untuk tahap *training*, *testing*, hingga evaluasi kinerja sistem. Terdiri dari fungsi `InitializeWeights()`, `SetWeights()`, `GetWeights()`, `ComputeOutputs()`, `UpdateWeights()`, `Train()`, `MeanSquaredError()`, dan `Accuracy()`.

4.3.1. Implementasi Tahap Praproses Teks

Data masukan pada tahap praproses teks merupakan sebuah berkas berekstensi *.xls* yang berisi kumpulan dokumen teks *tweet* yang telah disiapkan sebelumnya. Berkas tersebut akan dimuat sehingga sistem dapat membacanya kemudian disimpan ke basis data sebagai fitur. Proses memuat berkas berlangsung ketika fungsi `muat_Click()` dijalankan. Tujuan utama fungsi ini adalah mengirim dokumen teks dari sebuah berkas kemudian menyimpan ke dalam variabel dalam kode sumber. Untuk kode sumber `muat_Click()` dapat dilihat pada Kode Sumber 4.1.

Penjelasan masing-masing baris untuk Kode Sumber 4.1 adalah sebagai berikut.

- Baris ke-1 merupakan deklarasi fungsi `muat_Click()`. Fungsi ini otomatis dijalankan ketika *button* Muat pada tampilan antarmuka ditekan, yang akan dijelaskan pada subbab 4.4.
- Baris ke-2 dan baris ke-25 menunjukkan bahwa seluruh kode sumber yang ada di antaranya merupakan bagian dari fungsi `muat_Click()`.
- Baris ke-3 merupakan kode untuk inisialisasi variabel bertipe data `DataTable` untuk menyimpan hasil pembacaan pada

berkas. Dan, pada baris ke-4 dilakukan *reset* data pada variabel untuk menghindari apabila variabel tersebut telah terisi.

- Baris ke-5 merupakan kode untuk inisialisasi fungsi `OpenFileDialog`. Baris ke-6, baris ke-7, dan baris ke-18 merupakan kode membuka jendela baru untuk memilih berkas dalam direktori.
- Baris ke-8 hingga baris ke-13 merupakan kode untuk menyambungkan *software* pengembang dengan berkas berekstensi *.xls*. Dengan tujuan aplikasi dapat membaca isi berkas.
- Baris ke-14 merupakan kode untuk mengambil informasi dari berkas yang telah dimuat.
- Baris ke15 hingg baris ke-17 merupakan kode untuk menyimpan seluruh informasi ke dalam sebuah variabel yang telah diinisialisasi di awal fungsi.
- Baris ke-19, baris ke-20, dan baris ke-23 akan melakukan tahap praproses teks terhadap masing-masing dokumen yang telah disimpan dalam variabel.
- Baris ke-21 memanggil fungsi `insertClass()`, yakni fungsi untuk menjalankan *query* `INSERT INTO` yang bertujuan menyimpan informasi dokumen dan kelas ke dalam basis data, agar sistem mengenali kelas dari masing-masing dokumen.
- Baris ke-22 memanggil fungsi `tokenizeTweet()`, yakni fungsi untuk mengimplementasikan tahap praproses teks. Untuk detail teknis fungsi `tokenizeTweet()` dapat dilihat pada Kode Sumber 4.2.
- Baris ke-24 merupakan kode untuk memanggil fungsi calculateTFIDF(). Fungsi tersebut merupakan fungsi untuk menjalankan *stored procedure* yang telah dibuat dalam basis data untuk melakukan pembobotan. *Query stored procedure* dimaksud termasuk dalam subbab 4.3.2 pada Kode Sumber 4.3.

```
1:  void muat Click(object sender, EventArgs e)
2:  {
3:    DataTable xlsData = new DataTable();
4:    xlsData.Reset();
5:    OpenFileDialog xlsOfd = new OpenFileDialog();
6:    if (xlsOfd.ShowDialog() == DialogResult.OK)
7:    {
8:      OleDbConnection xlsCon = new
        OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.
        0;Data Source=" + xlsOfd.FileName + ";Extended
        Properties='Excel 8.0;HDR=Yes'");
9:      OleDbCommand xlsCmd = new OleDbCommand();
10:     OleDbDataAdapter xlsAdt = new OleDbDataAdapter();
11:     xlsCmd.Connection = xlsCon;
12:     xlsCon.Open();
13:     DataTable xlsSch =
        xlsCon.GetOleDbSchemaTable(OleDbSchemaGuid.Tables
        , null);
14:     xlsCmd.CommandText = "SELECT tweet_id,
        tweet_user, tweet, class_id, class_name From [" +
        xlsSch.Rows[0]["TABLE NAME"].ToString() + "]";
15:     xlsAdt.SelectCommand = xlsCmd;
16:     xlsAdt.Fill(xlsData);
17:     xlsCon.Close();
18:   }
19:   for (int i = 0; i < xlsData.Rows.Count; i++)
20:   {
21:     insertClass(i,
        Int32.Parse(xlsData.Rows[i][3].ToString()),
        xlsData.Rows[i][4].ToString());
22:     tokenizeTweet(i, xlsData.Rows[i][2].ToString());
23:   }
24:   Database.calculateTFIDF();
25: }
```

Kode Sumber 4.1 Implementasi Fungsi `muat_Click()`

Fungsi `tokenizeTweet()` pada Kode Sumber 4.2 merupakan implementasi dari tahap praproses teks. Pada fungsi ini, teks pada masing-masing dokumen akan melalui empat proses, yakni *case folding*, *tokenizing*, *filtering*, dan *analyzing*. Teks utuh akan berubah menjadi potongan teks yang lebih kecil. Hasil yang didapat dari fungsi ini adalah fitur teks yang akan digunakan pada tahap *training*, serta nilai frekuensi masing-masing fitur yang akan disimpan ke dalam basis data sebagai bahan untuk pembobotan.

```
 1:  void tokenizeTweet(int iddoc, string tweet)
 2:  {
 3:    StringBuilder builder = new StringBuilder();
 4:    Dictionary<string, int> book = new
       Dictionary<string, int>();
 5:    String[] texts = Regex.Split(tweet, " ");
 6:    var found = new Dictionary<string, bool>();
 7:    string complete = null;
 8:    foreach (string text in texts)
 9:    {
10:      string check = text.ToLower();
11:      check = (check.Contains("@") ||
         check.Contains("#") || check.Contains("rt") ||
         check.Contains("new") || check.Contains("year")
         || check.Contains("resolution") ||
         check.Contains("http")) ? "" : check;
12:      complete += " " + check;
13:    }
14:    String[] words =
       (complete.Trim()).Split(delimiters,
       StringSplitOptions.RemoveEmptyEntries);
15:    foreach (string word in words)
16:    {
17:      string check = word.ToLower();
18:      check = Regex.Replace(check, "[^a-zA-Z]+", "");
19:      if (check.Length >= 3 &&
         !stops.ContainsKey(word))
20:      {
21:        check = (!stops.ContainsKey(check) &&
          !found.ContainsKey(check)) ? check : "";
22:        if (book.ContainsKey(check)) book[check]++;
23:        else book[check] = 1;
24:        if (!flag.ContainsKey(check) && check != "")
25:          flag[check] = count++;
26:      }
27:    }
28:    var sorted = (from entry in book orderby
       entry.Value descending select
       entry).ToDictionary(pair => pair.Key, pair =>
       pair.Value);
29:    foreach (KeyValuePair<string, int> pair in sorted)
30:      if (pair.Key != "")
31:        insertTF(iddoc, pair.Key.ToString(),
          pair.Value, flag[pair.Key]);
32:  }
```

Kode Sumber 4.2 Implementasi Fungsi `tokenizeTweet()`

Penjelasan untuk masing-masing baris pada Kode Sumber 4.2 adalah sebagai berikut.

- Baris ke-1 merupakan deklarasi fungsi `tokenizeTweet()`.
- Baris ke-2 dan baris ke-32 menunjukkan seluruh kode sumber yang merupakan bagian dari fungsi `tokenizeTweet()`.
- Baris ke-3 hingga baris ke-7 merupakan kode inisialisasi untuk variabel yang diperlukan dalam sistem.
- Baris ke-8 hingga baris ke-13 merupakan kode implementasi proses *case folding* dan *filtering*, hanya untuk menghilangkan elemen-elemen *tweet* seperti *mention*, *hashtag*, dan *link*.
- Baris ke-14 adalah kode untuk memisahkan teks.
- Baris ke-15 hingga baris ke-27 merupakan tahapan sistem menghitung berapa kali suatu kata pada sebuah dokumen. Serta menghilangkan kata yang termasuk dalam *stopword*.
- Baris ke-29 hingga baris ke-31 merupakan kode untuk memasukkan nilai frekuensi masing-masing kata ke dalam basis data. Setiap fitur akan dikirim ke dalam basis data setelah memanggil fungsi `insertTF()` pada baris ke-31. Contoh hasil fungsi dalam basis data ditunjukkan Gambar 4.5.

	IDREC	IDDOC	IDWRD	WORDT	TFVAL	TFNOR	IDFVL	TFIDF
1	1	0	0	stay	1	NULL	NULL	NULL
2	2	0	1	saaaame	1	NULL	NULL	NULL
3	3	1	2	acting	1	NULL	NULL	NULL
4	4	1	3	mixed	1	NULL	NULL	NULL
5	5	2	4	eat	1	NULL	NULL	NULL
6	6	2	5	healthier	1	NULL	NULL	NULL
7	7	3	6	produce	1	NULL	NULL	NULL
8	8	3	7	druggie	1	NULL	NULL	NULL
9	9	3	8	anthem	1	NULL	NULL	NULL
10	10	4	9	dream	1	NULL	NULL	NULL

Gambar 4.5 Hasil Masukan Fungsi `insertTF()`

Nilai dalam basis data, seperti yang ditunjukkan pada Gambar 4.5, merupakan informasi awal yang dibutuhkan pada

tahap selanjutnya, yakni tahap representasi fitur teks. Pada tahap representasi teks, sistem akan melakukan pembobotan dengan Algoritma TF-IDF.

4.3.2. Implementasi Tahap Representasi Fitur Teks

Pembobotan TF-IDF fitur dilakukan dengan menjalankan *query stored procedure* yang dibuat dalam basis data. *Stored procedure* akan menghitung frekuensi total masing-masing fitur dalam sebuah tabel. Kemudian, akan dilakukan perhitungan terhadap nilai TF untuk memperoleh nilai normalisasinya, dan disimpan pada kolom `TFVAL`. Selanjutnya, pada *stored procedure* akan dihitung nilai IDF yang disimpan pada kolom `IDFVL`. Kode *stored procedure* akan ditunjukkan pada Kode Sumber 4.3.

```
 1:  CREATE PROCEDURE [dbo].[SET TFIDFVALUE]
 2:  AS
 3:  BEGIN
 4:    SET NOCOUNT ON;
 5:    UPDATE src
 6:    SET src.TFNOR = src.TFVAL * 1.00 / dst.TFSUM
 7:    FROM FEATURE src
 8:    INNER JOIN (
 9:    SELECT IDDOC, MAX(TFVAL) TFSUM FROM FEATURE GROUP
       BY  IDDOC
10:    ) dst ON src.IDDOC = dst.IDDOC
11:    UPDATE FEATURE
12:    SET IDFVL = LOG(((SELECT MAX(IDDOC) FROM FEATURE)
       * 1.0 / val.TFCNT), 10)
13:    FROM (
14:    SELECT src.WORDT, COUNT(*) AS TFCNT
15:    FROM FEATURE AS dst
16:    Join FEATURE src
17:    On src.WORDT = dst.WORDT AND src.IDDOC = dst.IDDOC
18:    GROUP BY src.WORDT
19:    ) val
20:    WHERE FEATURE.WORDT = val.WORDT
21:    UPDATE FEATURE
22:    SET TFIDF = TFNOR * 1.00 * IDFVL
23:  END
```

Kode Sumber 4.3 *Stored Procedure* `SET_TFIDFVALUE`

	IDREC	IDDOC	IDWRD	WORDT	TFVAL	TFNOR	IDFVL	TFIDF
1	1	0	0	stay	1	1	1.82908185433774	1.82908185433774
2	2	0	1	saaaame	1	1	3.2914798522367	3.2914798522367
3	3	1	2	acting	1	1	2.89353984356466	2.89353984356466
4	4	1	3	mixed	1	1	3.59250984790068	3.59250984790068
5	5	2	4	eat	1	1	1.5796726231955	1.5796726231955
6	6	2	5	healthier	1	1	2.0484418035504	2.0484418035504
7	7	3	6	produce	1	1	2.89353984356466	2.89353984356466
8	8	3	7	druggie	1	1	3.59250984790068	3.59250984790068
9	9	3	8	anthem	1	1	3.59250984790068	3.59250984790068
10	10	4	9	dream	1	1	2.59250984790068	2.59250984790068

Gambar 4.6 Hasil *Stored Procedure* `SET_TFIDFVALUE`

Setelah *stored procedure* dijalankan, maka kolom `TFNOR`, `IDFVL`, dan `TFIDF` akan terisi nilai. Kolom `TFNOR` merupakan hasil normalisasi nilai TF setiap fitur, sedangkan `IDFVL` merupakan nilai yang mewakili fitur pada dokumen. Untuk kolom `TFIDF` adalah nilai TF-IDF masing-masing fitur, hasil perkalian pada `TFNOR` dan `IDFVL`. Hasil *stored procedure* dapat dilihat pada Gambar 4.6.

4.3.3. Implementasi Tahap Metode *Hold-out*

Data pada kolom `TFIDF` masing-masing fitur selanjutnya diambil dan disimpan dalam variabel. Data inilah yang selanjutnya akan diolah dalam tahap *training*. Sebelum memasuki tahap utama sistem, yakni tahap *training*, sistem akan memisahkan data total di yang telah diambil di awal menjadi dua bagian, yaitu data *training* dan data *testing*. Metode *Hold-out* akan diimplementasikan pada fungsi `separateMatrix()`, di mana sistem akan mengacak urutan awal data kemudian membagi data menjadi dua bagian sesuai jumlah persentase yang diinginkan.

Penjelasan untuk masing-masing baris pada Kode Sumber 4.2 adalah sebagai berikut.

- Baris ke-1 merupakan deklarasi fungsi `separateMatrix()`.
- Baris ke-11 hingga baris ke-12 merupakan kode untuk menyimpan nilai sekuens atau urutan awal data.
- Baris ke-13 hingga baris ke-19 merupakan tahapan sistem akan mengacak sekuens atau urutan dokumen data masukan.

- Baris ke-20 hingga baris ke-28 merupakan kode untuk menyimpan data *train* yang telah dipisah ke variabel baru.
- Baris ke-29 hingga baris ke-36 merupakan kode untuk menyimpan data *train* yang telah dipisah ke variabel baru.

```
 1:  void separateMatrix(double[][] data_set, double
     test, out double[][] train_set, out double[][]
     test set)
 2:  {
 3:    Random rnd = new Random(0);
 4:    int row = data_set.Length;
 5:    int col = data_set[0].Length;
 6:    int test_row = (int)(row * test);
 7:    int train_row = row - test_row;
 8:    train_set = new double[train_row][];
 9:    test_set = new double[test_row][];
10:    int[] sequence = new int[row];
11:    for (int i = 0; i < sequence.Length; ++i)
12:      sequence[i] = i;
13:    for (int i = 0; i < sequence.Length; ++i)
14:    {
15:      int r = rnd.Next(i, sequence.Length);
16:      int tmp = sequence[r];
17:      sequence[r] = sequence[i];
18:      sequence[i] = tmp;
19:    }
20:    int s = 0;
21:    int d = 0;
22:    for (; s < train_row; ++s)
23:    {
24:      train_set[d] = new double[col];
25:      int idx = sequence[s];
26:      Array.Copy(data_set[idx], train_set[d], col);
27:      ++d;
28:    }
29:    d = 0;
30:    for (; s < row; ++s)
31:    {
32:      test_set[d] = new double[col];
33:      int idx = sequence[s];
34:      Array.Copy(data_set[idx], test_set[d], col);
35:      ++d;
36:    }
37:  }
```

Kode Sumber 4.4 Implementasi Fungsi `separateMatrix()`

4.3.4. Implementasi Tahap Pelatihan

Implementasi tahap *training* dimulai setelah dataset utama dibagi menjadi data *training* dan data *testing*. Karena ini merupakan tahap *training*, maka data yang digunakan adalah data *training*. Sedangkan untuk data *testing* akan dikesampingkan terlebih dahulu. Tahap *training* akan dijalankan ketika *button* Train pada antarmuka pengguna ditekan. *Button* ini akan menjalankan fungsi `train_Click()`, yang kemudian akan memanggil fungsi `Train()` untuk Algoritma *Backpropagation* seperti yang ditunjukkan Kode Sumber 4.5.

```
 1:  void Train(double[][] data, int max, double
     learn_rate, double momentum, out double mse)
 2:  {
 3:    int epoch = 0;
 4:    mse = 0;
 5:    double[] values = new double[node_inp];
 6:    double[] target = new double[node_out];
 7:    while (epoch < max)
 8:    {
 9:      for (int i = 0; i < data.Length; ++i)
10:      {
11:        Array.Copy(data[i], values, node_inp);
12:        Array.Copy(data[i], node_inp, target, 0,
           node_out);
13:        ComputeOutputs(values);
14:        UpdateWeights(target, learn_rate, momentum);
15:      }
16:      mse = MeanSquaredError(data);
17:      ++epoch;
18:    }
19:  }
```

Kode Sumber 4.5 Implementasi Fungsi `Train()`

Penjelasan untuk masing-masing baris pada Kode Sumber 4.2 adalah sebagai berikut.

- Baris ke-1 merupakan deklarasi fungsi `Train()`.
- Baris ke-3 hingga baris ke-6 merupakan kode inisialisasi untuk variabel yang diperlukan dalam sistem. Di antaranya adalah inisialisasi *epoch*, *mse*, *values* untuk menyimpan nilai

masukan pada *input layer* dan *targets* untuk menyimpan nilai target pada *output layer*.

- Baris ke-7 hingga baris ke-18 merupakan kode untuk melakukan proses *training* sebanyak nilai maksimum *epoch*.
- Baris ke-9 hingga baris ke-15 merupakan tahapan sistem *training* untuk masing-masing vektor dokumen yang masuk.
- Baris ke-11 dan baris ke-12 secara berturut-turut menyimpan nilai masukan dan nilai target vektor ke sebuah variabel baru.
- Fungsi `ComputeOutputs()`pada baris ke-13 merupakan fungsi untuk menghitung nilai aktivasi pada *output layer*. Sedangkan fungsi `UpdateWeights()` merupakan fungsi memperbaiki nilai bobot setelah mengetahu nilai *error* hasil `ComputeOutputs()`. Secara berturut-turut implementasi fungsi ditunjukkan Kode Sumber 4.6 dan Kode Sumber 4.
- Baris ke-16 merupakan kode untuk menghitung nilai MSE dengan yang ditunjukkan pada Kode Sumber 4.8.

```
 1:  double[] ComputeOutputs(double[] values)
 2:  {
 3:    double[] sum_hid = new double[node_hid];
 4:    double[] sum_out = new double[node_out];
 5:    for (int i = 0; i < values.Length; ++i)
 6:      this.inputs[i] = values[i];
 7:    for (int j = 0; j < node_hid; ++j)
 8:      for (int i = 0; i < node_inp; ++i)
 9:        sum_hid[j] += this.inputs[i] *
           this.weight_ih[i][j];
10:    for (int i = 0; i < node_hid; ++i)
11:      sum_hid[i] += this.bias_hid[i];
12:    for (int i = 0; i < node_hid; ++i)
13:      this.out_hid[i] = HyperTanFunction(sum_hid[i]);
14:    for (int j = 0; j < node_out; ++j)
15:      for (int i = 0; i < node_hid; ++i)
16:        sum_out[j] += out_hid[i] * weight_ho[i][j];
17:    for (int i = 0; i < node_out; ++i)
18:      sum_out[i] += bias_out[i];
19:    double[] result = new double[node_out];
20:    Array.Copy(this.outputs, result, result.Length);
21:    return result;
22:  }
```

Kode Sumber 4.6 Implementasi Fungsi `ComputeOutputs()`

Penjelasan untuk masing-masing baris pada Kode Sumber 4.6 adalah sebagai berikut.

- Baris ke-1 merupakan deklarasi fungsi `ComputeOutputs()`.
- Baris ke-3 hingga baris ke-4 merupakan kode inisialisasi untuk variabel yang diperlukan dalam sistem, yakni untuk menyimpan nilai *sum* pada *hidden layer* dan *output layer*.
- Baris ke-5 hingga ke-6 merupakan proses memindahkan nilai masukan ke variabel.
- Baris ke-7 hingga barus ke-9 merupakan penjumlahan total hasil kali nilai masukan dengan bobot *neuron* pada *hidden layer*.
- Baris ke-10 hingga baris ke-11 merupakan hal penjumlahan bobot *neuron* pada *hidden layer*.
- Baris ke-12 hingga baris ke-13 merupakan proses menghitung nilai aktivasi pada *hidden layer*.
- Baris ke-14 hingga baris ke-16 merupakan penjumlahan total hasil kali nilai keluaran pada *hidden layer* dengan bobot *neuron* pada *output layer*.
- Baris ke-17 hingga baris ke-18 merupakan hal penjumlahan bobot *neuron* pada *output layer*.
- Baris ke-19 merupakan proses menghitung nilai aktivasi pada *output layer*.

```
 1:  void UpdateWeights(double[] target, double
     learn_rate, double momentum)
 2:  {
 3:    for (int i = 0; i < grad_out.Length; ++i)
 4:    {
 5:      double derivative = (1 - outputs[i]) *
         outputs[i];
 6:      grad_out[i] = derivative * (target[i] -
         outputs[i]);
 7:    }
 8:    for (int i = 0; i < grad_hid.Length; ++i)
 9:    {
10:      double derivative = (1 - out_hid[i]) * (1 +
         out_hid[i]);
```

Kode Sumber 4.7 Implementasi Fungsi `UpdateWeights()`

```
11:       for (int j = 0; j < node_out; ++j)
12:       {
13:         double x = grad_out[j] * weight_ho[i][j];
14:         sum += x;
15:       }
16:       grad_hid[i] = derivative * sum;
17:     }
18:     for (int i = 0; i < weight_ih.Length; ++i)
19:     {
20:       for (int j = 0; j < weight_ih[0].Length; ++j)
21:       {
22:         double delta = learn_rate * grad_hid[j] *
            inputs[i];
23:         weight_ih[i][j] += delta;
24:         weight_ih[i][j] += momentum * delta_ih[i][j];
25:         delta_ih[i][j] = delta;
26:       }
27:     }
28:     for (int i = 0; i < bias_hid.Length; ++i)
29:     {
30:       double delta = learn_rate * grad_hid[i] * 1.0;
31:       bias_hid[i] += delta;
32:       bias_hid[i] += momentum * delta_hid[i];
33:       delta_hid[i] = delta;
34:     }
35:     for (int i = 0; i < weight_ho.Length; ++i)
36:     {
37:       for (int j = 0; j < weight_ho[0].Length; ++j)
38:       {
39:         double delta = learn_rate * grad_out[j] *
            out_hid[i];
40:         weight_ho[i][j] += delta;
41:         weight_ho[i][j] += momentum * delta_ho[i][j];
42:         delta_ho[i][j] = delta;
43:       }
44:     }
45:     for (int i = 0; i < bias_out.Length; ++i)
46:     {
47:       double delta = learn_rate * grad_out[i] * 1.0;
48:       bias_out[i] += delta;
49:       bias_out[i] += momentum * delta_out[i];
50:       delta_out[i] = delta;
51:     }
52:   }
```

Kode Sumber 4.7 Implementasi Fungsi `UpdateWeights()` (lanj.)

```
 1:  double MeanSquaredError(double[][] data)
 2:  {
 3:    double sumSquaredError = 0.0;
 4:    double[] values = new double[node_inp];
 5:    double[] targets = new double[node_out];
 6:    for (int i = 0; i < data.Length; ++i)
 7:    {
 8:      Array.Copy(data[i], values, node_inp);
 9:      Array.Copy(data[i], node_inp, targets, 0,
         node_out);
10:      double[] actives = this.ComputeOutputs(values);
11:      for (int j = 0; j < node_out; ++j)
12:      {
13:        double err = targets[j] - actives[j];
14:        sumSquaredError += err * err;
15:      }
16:    }
17:    return sumSquaredError / data.Length;
18:  }
```

Kode Sumber 4.8 Implementasi Fungsi `MeanSquaredError()`

Penjelasan untuk masing-masing baris pada Kode Sumber 4 adalah sebagai berikut.

- Baris ke-1 merupakan deklarasi fungsi `UpdateWeights()`.
- Baris ke-3 hingga baris ke-7 merupakan kode untuk menghitung nilai *gradient descent* masing-masing *neuron* pada *output layer*.
- Baris ke-8 hingga baris ke-18 merupakan kode untuk menghitung nilai *gradient descent* masing-masing *neuron* pada *hidden layer*.
- Baris ke-19 hingga baris ke-28 merupakan kode untuk memperbaiki nilai bobot antara *neuron* pada *input layer* dan *neuron* pada *hidden layer*.
- Baris ke-29 hingga baris ke-35 merupakan kode untuk memperbaiki nilai bias *neuron* pada *hidden layer*.
- Baris ke-36 hingga baris ke-45 merupakan kode untuk memperbaiki nilai bobot antara *neuron* pada *hidden layer* dan *neuron* pada *output layer*.

- Baris ke-46 hingga baris ke-52 merupakan kode untuk memperbaiki nilai bias *neuron* pada *output layer*.

Penjelasan untuk masing-masing baris pada Kode Sumber 4.8 adalah sebagai berikut.

- Baris ke-1 merupakan deklarasi fungsi `MeanSquareError()`.
- Baris ke-2 dan baris ke-18 menunjukkan bahwa seluruh kode sumber yang ada di antaranya merupakan bagian dari fungsi `MeanSquaredError()`.
- Baris ke-3 hingga baris ke-5 merupakan kode inisialisasi untuk variabel yang diperlukan dalam sistem.
- Baris ke-8 dan baris ke-9 secara berturut-turut menyimpan nilai masukan dan nilai target vektor ke sebuah variabel baru untuk diproses pada proses selanjutnya.
- Fungsi `ComputeOutputs()`pada baris ke-10 merupakan fungsi untuk menghitung nilai aktivasi pada *output layer* yang nantinya dibandingkan dengan nilai target.
- Baris ke-11 hingga baris ke-15 merupakan kode untuk menghitung selisih perbedaan antara nilai aktivasi dan nilai target. Kemudian, hasil kuadrat selisih akan dijumlahkan hingga seluruh dokumen dihitung.
- Baris ke-17 merupakan perhitungan akhir untuk mendapatkan nilai MSE dari satu kali tahap *training* Algoritma *Backpropagation*.

4.3.5. Implementasi Tahap Pengujian dan Evaluasi

Pada Tugas Akhir ini, tahap *testing* dan tahap evaluasi akan diimplementasikan secara bersamaan. Karena, untuk menentukan nilai benar dan salah, sistem perlu melakukan perhitungan nilai aktivasi yang kemudian baru bisa dibandingkan dengan nilai target. Penjelasan untuk masing-masing baris pada Kode Sumber 4.9 adalah sebagai berikut.

- Baris ke-1 merupakan deklarasi fungsi `Accuracy()`.

- Baris ke-2 dan baris ke-20 menunjukkan bahwa kode sumber yang ada merupakan bagian dari fungsi `Accuracy()`.
- Baris ke-3 hingga baris ke-7 merupakan kode inisialisasi untuk variabel yang diperlukan dalam sistem.
- Baris ke-10 dan baris ke-11 secara berturut-turut menyimpan nilai masukan dan nilai target vektor ke sebuah variabel baru untuk diproses pada proses selanjutnya.
- Fungsi `ComputeOutputs()` pada baris ke-12 merupakan fungsi untuk menghitung nilai aktivasi pada *output layer* yang nantinya dibandingkan dengan nilai target.
- Baris ke-13 hingga baris ke-18 merupakan kode untuk membandingkan apakah nilai aktivasi yang dihasilkan sama dengan nilai target. Apabila sama, maka nilai benar bertambah satu. Dan apabila tidak sama, maka sebaliknya.
- Baris ke-17 merupakan perhitungan akhir untuk mendapatkan nilai akurasi untuk evaluasi kinerja sistem.

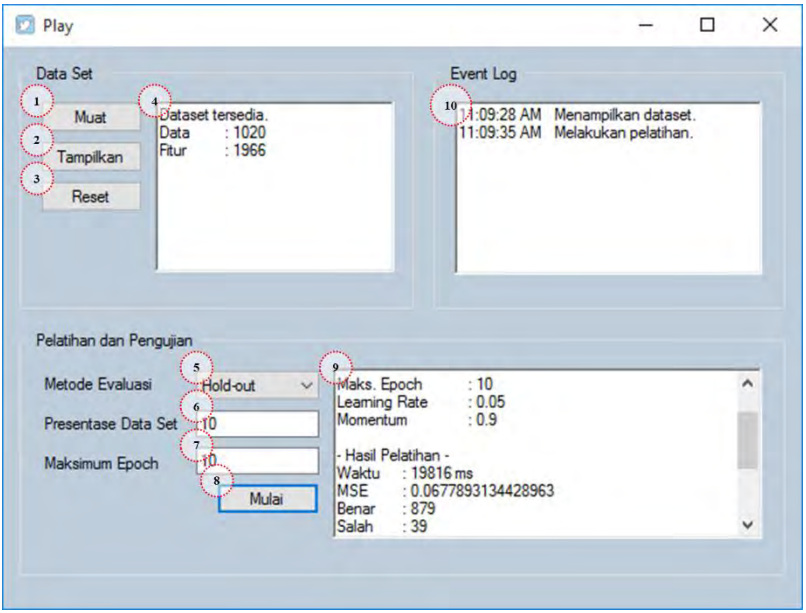
```
 1:  double Accuracy(double[][] data, out int correct,
     out int wrong)
 2:  {
 3:    correct = 0;
 4:    wrong = 0;
 5:    double[] values = new double[node_inp];
 6:    double[] target = new double[node_out];
 7:    double[] result;
 8:    for (int i = 0; i < data.Length; ++i)
 9:    {
10:      Array.Copy(data[i], values, node_inp);
11:      Array.Copy(data[i], node_inp, target, 0,
         node_out);
12:      result = this.ComputeOutputs(values);
13:      int ind = MaxIndex(result);
14:      if (Equals(target[ind], 1.0))
15:        ++correct;
16:      else
17:        ++wrong;
18:    }
19:    return (correct * 1.0) / (correct + wrong);
20:  }
```

Kode Sumber 4.9 Implementasi Fungsi `Accuracy()`

4.4. Implementasi Antarmuka Pengguna

Implementasi tampilan antarmuka pengguna pada perangkat lunak berbasis *desktop* dan berjalan pada sistem operasi *Windows*. Berikut ini dijelaskan mengenai implementasi tampilan antarmuka pengguna yang terdapat pada perangkat lunak. Implementasi antarmuka pengguna dapat dilihat pada Gambar 4.7.

Tampilan antarmuka pengguna pada Tugas Akhir ini dibagi menjadi tiga bagian. Bagian pertama adalah bagian Data Set. Pada bagian ini terfokus pada data masukan. Pada bagian kedua, yaitu Pelatihan dan Pengujian difokuskan pada tahap pelatihan, pengujian, hingga menghasilkan evaluasi kinerja sistem. Sedangkan untuk bagian *Event Log* merupakan bagian yang bertujuan mencatat kegiatan yang dilakukan pengguna selama aplikasi berjalan. Contoh konten pada masing-masing bagian telah jelas ditunjukkan pada Gambar 4.7.



Gambar 4.7 Implementasi Antarmuka Pengguna

Fungsi masing-masing elemen pada implementasi antarmuka akan dijelaskan lebih detail pada Tabel 4.4. Karena masing-masing elemen memiliki fungsi yang spesifik, sehingga jika pengguna elemen salah maka dapat menyebabkan *error* aplikasi.

Tabel 4.4 Keterangan Implementasi Antarmuka Pengguna

No	Nama Elemen	Keterangan
1	Tombol Muat	Untuk memuat berkas baru yang berisi dokumen teks *tweet*, dengan menjalankan fungsi `muat_Click()`.
2	Tombol Tampil	Untuk menampilkan banyak dokumen dan fitur yang telah tersimpan dalam basis data, dengan menjalankan fungsi `tampil_Click()`.
3	Tombol Reset	Untuk mereset ulang isi data dalam basis data, dengan menjalankan fungsi `reset_Click()`.
4	Kotak Data Set	Untuk menampilkan banyak dokumen dan fitur ketika tombol Tampil ditekan.
5	Teks Metode	Untuk menampilkan metode evaluasi yang digunakan pada sistem.
6	Teks Presentase	Untuk menentukan presentase perbandingan data *training* dan data *testing* dari data keseluruhan.
7	Teks *Epoch*	Untuk menentukan maksimum *epoch* yang digunakan dalam tahap *training*.
8	Tombol Mulai	Untuk memulai tahap *training* dengan informasi data yang sudah lengkap, dengan menjalankan fungsi `train_Click()`.
9	Kotak Pelatihan dan Pengujian	Untuk menampilkan hasil tahap *training* dan tahap *testing* sistem. Di antaranya yang ditampilkan adalah akurasi, *running time*, dan nilai MSE.

BAB V
UJI COBA DAN EVALUASI

Bab ini membahas uji coba dan evaluasi terhadap perangkat lunak yang telah dikembangkan dari implementasi Algoritma *Backpropagation* pada klasifikasi *tweet* resolusi.

5.1. Lingkungan Uji Coba

Lingkungan uji coba yang digunakan dalam pembuatan Tugas Akhir ini meliputi perangkat lunak dan perangkat keras yang digunakan untuk melakukan uji coba implementasi Algoritma *Backpropagation* pada klasifikasi *tweet* resolusi. Lingkungan uji coba meliputi perangkat keras dan perangkat lunak yang dijelaskan pada Tabel 5.1.

Tabel 5.1 Lingkungan Uji Coba Sistem

Perangkat	Spesifikasi
Perangkat keras	Prosesor: AMD E-450 APU with Radeon™ HD Graphics 1.65 GHz Memori: 4.00 GB
Perangkat lunak	Sistem Operasi: Windows 10 Pro Perangkat Pengembang: Visual Studio Ultimate 2013, SQL Server 2014 Management Studio

5.2. Data Uji Coba

Data uji coba yang digunakan adalah kumpulan data teks *tweet* yang disimpan pada berkas berekstensi *.xls*. Pada pengujian ini, akan digunakan dua jenis berkas. Berkas pertama memiliki 1056 dokumen teks *tweet*, sedangkan berkas kedua adalah memiliki 3914 dokumen teks *tweet*. Masing-masing berkas tediri dari enam kelas kategori. Di antaranya adalah *Career &*

Education, *Finance*, *Health & Fitness*, *Personal Growth*, *Recreation & Leisure*, dan *Relationship*. Perbedaan kedua berkas tidak hanya terletak pada jumlah dokumen, tetapi juga jumlah porsi dokumen pada masing-masing kelas. Berkas pertama memiliki ketidakseimbangan pada jumlah dokumen untuk masing-masing kelas, sedangkan dokumen memiliki jumlah dokumen yang sama untuk masing-masing kelas. Untuk lebih jelas, dapat dilihat pada Tabel 5.2. Dan, untuk contoh tampilan berkas yang berisi dokumen teks *tweet* ditunjukkan pada halaman LAMPIRAN.

Tabel 5.2 Informasi Komposisi Berkas 1 dan 2

No	Nama Berkas	Total Dok	Jumlah Dokumen untuk Kelas ke					
			1	2	3	4	5	6
1	Berkas 1	1056	176	176	176	176	176	176
2	Berkas 2	3914	215	176	840	1781	467	435

Tabel 5.3 Informasi Jumlah Fitur Berkas 1 dan 2

No	Nama Berkas	Total Dokumen	Jumlah Fitur Kata	
			Sebelum	Sesudah
1	Berkas 1	1056	3149	1990
2	Berkas 2	3914	7770	4551

Tabel 5.2 menunjukkan jumlah dokumen yang dimiliki masing-masing berkas. Namun, hal ini belum dapat menentukan jumlah fitur hasil tahap praproses teks. Setelah dilakukan tahap praproses teks, diketahui bahwa berkas ke-1 dengan jumlah dokumen sebanyak 1056 menghasilkan fitur sebanyak 1990 kata, sedangkan berkas ke-2 menghasilkan sebanyak 4551 fitur kata. Sedangkan, Tabel 5.3 menunjukkan perbedaan jumlah fitur kata sebelum dan sesudah melalui tahap praproses teks. Di mana menunjukkan bahwa tahap ini berpengaruh besar dalam penentuan fitur untuk data set awal sebelum memasuki tahap utama sistem, yakni tahap *training*. Selain itu, juga menujukkan bahwa jika data dilipatgandakan sebanyak empat kali, belum tentu fitur yang dihasilkan pun empat kali dari jumlah yang lebih kecil. Hal ini

bergantung pada jenis kata yang digunakan, terutama penggunaan yang sering dalam suatu dokumen teks.

5.3. Skenario Uji Coba

Pada subbab ini akan dijelaskan mengenai skenario uji coba yang telah dilakukan. Skenario uji coba bertujuan untuk mencari faktor yang mempengaruhi tingkat akurasi pada sistem yang dibangun. Terdapat beberapa skenario uji coba yang telah dilakukan, diantaranya yaitu:

1. Hasil akurasi berdasarkan variasi nilai *learning rate*.
   Tujuan skenario ini adalah untuk mengetahui pengaruh nilai *learning rate* terhadap nilai akurasi.
2. Hasil akurasi berdasarkan variasi jumlah *neuron* pada *hidden layer*.
   Tujuan skenario ini adalah untuk mengetahui pengaruh jumlah *neuron* pada *hidden layer* terhadap nilai akurasi.
3. Hasil akurasi berdasarkan variasi maksimum nilai *epoch*.
   Tujuan skenario ini adalah untuk mengetahui pengaruh nilai *epoch* terhadap nilai akurasi.
4. Hasil akurasi berdasarkan variasi persentase data pada metode *hold-out*.
   Tujuan skenario ini adalah untuk mengetahui pengaruh perbandingan data *training* dan data *testing* terhadap nilai akurasi.
5. Hasil akurasi menggunakan kombinasi *neuron*, *epoch*, dan *learning rate* dari skenario uji coba ke-1, ke-2, dan ke-3.
   Tujuan skenario ini adalah untuk mengetahui seberapa besar perubahan akurasi terhadap perubahan nilai *learning rate* menjadi 0,10 dan 0,05; serta pembagian data set dengan persentase 10% dan 20%.

Selanjutnya skenario uji coba akan dibahas pada subbab-subbab selanjutnya berbarengan dengan hasil analisa yang didapat pada uji coba tersebut.

5.4. Hasil Pengujian dan Analisa Berdasarkan Variasi Nilai *Learning Rate*

Analisa dan pengujian pertama yang dilakukan adalah berdasarkan variasi nilai *learning rate* yang diaplikasikan pada sistem klasifikasi *tweet* resolusi. Berdasarkan tampilan antarmuka pengguna, masukan nilai yang dibutuhkan di antaranya persentase pembagian data *training* dan data *testing*, jumlah *neuron* pada *hidden layer*, serta maksimum *epoch*. Untuk ketiga nilai ini akan ditentukan secara *default*, yakni 10%, 1%, dan 10 kali untuk seluruh uji coba skenario ini.

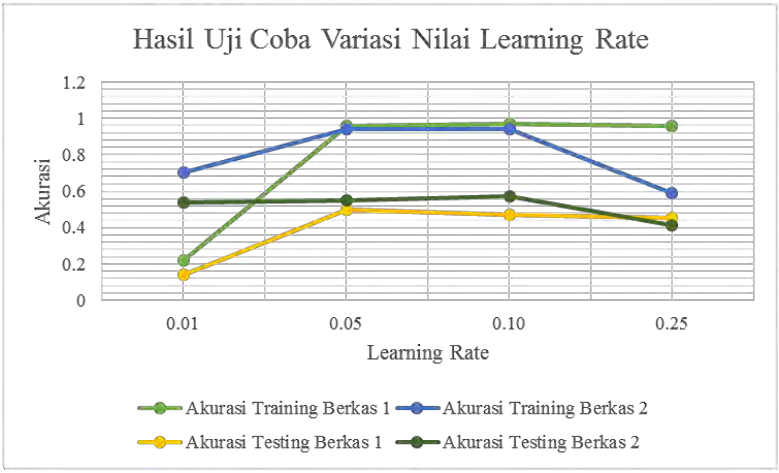
Tabel 5.4 Hasil Uji Coba Variasi Nilai *Learning Rate*

No	Data	Fitur	*Learning Rate*	MSE	Akurasi (%)		Lama *Train* (*s*)
					Train	*Test*	
1	1056	1990	0,01	0,82	0,22	0,14	78
2			0,05	0,06	0,96	0,50	80
3			**0,10**	**0,02**	**0,97**	**0,47**	**77**
4			0,25	0,03	0,95	0,45	97
5	3914	4551	0,01	0,35	0,70	0,54	3328
6			0,05	0,07	0,94	0,55	3497
7			**0,10**	**0,06**	**0,94**	**0,57**	**2959**
8			0,25	0,62	0,59	0,41	3073

Uji coba dengan variasi nilai *learning rate* akan diaplikasikan pada dua jenis berkas yang telah dijelaskan pada awal bab ini. Dan, untuk variasi nilai *learning rate* adalah 0,01; 0,05; 0,10 dan 0,25. Ketiga nilai tersebut akan diuji cobakan pada dua berkas data.

Dapat dilihat pada Tabel 5.4 hasil pengujian variasi nilai *learning rate*. Untuk berkas data ke-1, nilai akurasi baik untuk data *training* dan data *testing* berbanding lurus dengan nilai *learning rate*. Semakin besar nilai *learning rate*, akurasi juga meningkat. Yang awalnya dengan nilai *learning rate* 0,01 sistem klasifikasi hanya memiliki akurasi 0,22 pada data *training*, dengan *learning rate* 0,05 akurasi mampu meningkat hingga empat kali lipatnya,

yakni mencapai 0,96. Hal ini juga terjadi pada nilai *learning rate* 0,10; nilai akurasi juga meningkat menjadi 0,97. Tidak sesignifikan perubahan antara *learning rate* 0,01 ke 0,05; tapi nilai akurasi telah cukup mengindikasi bahwa meningkatnya nilai *learning rate* meningkatkan nilai akurasi juga. Namun, jika dilihat Gambar 5.1, peningkatan akurasi berhenti pada nilai *learning rate* 0,10, dan kemudian menurun pada 0,25.



Gambar 5.1 Grafik Uji Coba Variasi Nilai *Learning Rate*

Pada berkas ke-2, nilai akurasi meningkat, benar. Dengan *learning rate* 0,01 akurasi hanya mencapai angka 0,70; sedangkan dengan *learning rate* 0,05 akurasi menjadi lebih tinggi yaitu 0,94. Sama halnya dengan yang terjadi pada berkas ke-1, akurasi terus meningkat dan berhenti pada nilai *learning rate* 0,10. Namun, akurasi menurun cukup drastis ketika nilai *learning rate* diubah menjadi 0,25.

Pada skenario ini, didapatkan dua nilai *learning rate* dengan nilai akurasi tertinggi yakni 0,05 dan 0,10. Namun, jika menilik faktor lain pada nilai MSE, uji coba dengan nilai *learning rate* sebesar 0,10 memiliki nilai *error* yang lebih kecil. Serta waktu *training* yang dibutuhkan juga lebih sedikit. Maka dari itu, nilai

learning rate terbaik yang diambil untuk sistem klasifikasi tugas akhir ini adalah 0,10.

5.5. Hasil Pengujian dan Analisa Berdasarkan Variasi Jumlah *Neuron* pada *Hidden Layer*

Selanjutnya adalah analisa dan pengujian berdasarkan variasi jumlah *neuron* pada *hidden layer* yang diaplikasikan pada sistem klasifikasi *tweet* resolusi. Berdasarkan tampilan antarmuka pengguna, masukan nilai yang dibutuhkan di antaranya persentase pembagian data *training* dan data *testing*, serta maksimum *epoch*. Untuk ketiga nilai ini akan ditentukan secara *default*, yakni 10% dan 10 kali untuk seluruh uji coba skenario ini. Dan untuk nilai *learning rate*, secara *default* menggunakan hasil terbaik dari skenario uji coba sebelumnya, yakni 0,10.

Tabel 5.5 Keterangan Jumlah *Neuron* pada *Hidden Layer*

No	Data	Fitur	Persentase	Jumlah *Neuron* pada *Hidden Layer*
1	1056	1990	1‰	2
2			1%	20
3			10%	199
5	3914	4551	1‰	5
6			1%	46
7			10%	455

Uji coba awal pada tugas akhir ini adalah sebesar 2/3 dari *neuron* pada *hidden layer*, yang merupakan ukuran yang sering digunakan pada model jaringan saraf tiruan. Dengan pertimbangan jumlah *neuron* pada *hidden layer* terlalu besar dan berpengaruh sangat banyak pada lama waktu *training*, maka akan dilakukan uji coba untuk menemukan jumlah *neuron* pada *hidden layer* yang sesuai dengan data pada tugas akhir ini. Untuk itu dilakukan uji coba dengan variasi jumlah *neruron* pada *hidden layer* akan diaplikasikan pada dua jenis berkas juga tentunya. Dan, untuk variasinya adalah 1‰, 1%, dan 10% dari jumlah *neuron* pada *input*

layer. Sehingga, jumlah *neuron* pada *hidden layer* bergantung pada jumlah *neuron* pada *hidden layer*.

Uji coba dengan variasi jumlah *neruron* pada *hidden layer* akan diaplikasikan pada dua jenis berkas juga tentunya. Dan, untuk variasinya adalah 1‰, 1%, dan 10% dari jumlah *neuron* pada *input layer*. Sehingga, jumlah *neuron* pada *hidden layer* bergantung pada jumlah *neuron* pada *hidden layer*.

Tabel 5.6 Hasil Uji Coba Variasi *Neuron Hidden Layer*

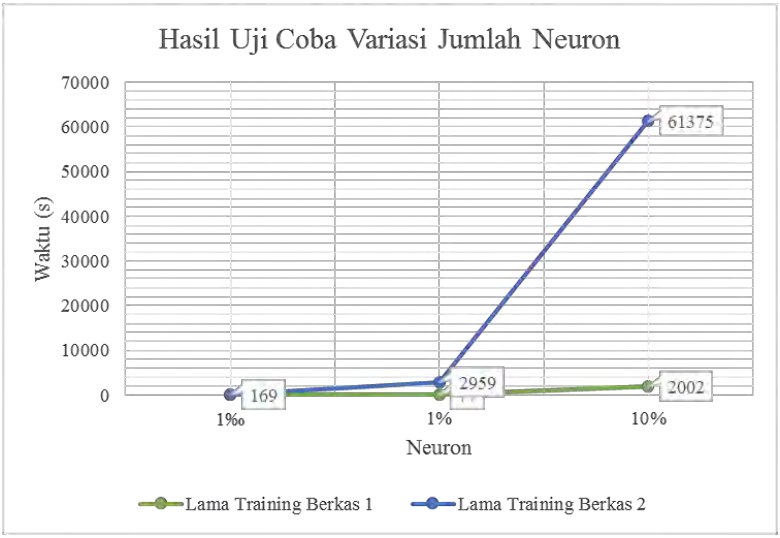
No	Data	Fitur	*Neuron*	MSE	Akurasi (%)		Lama *Train* (*s*)
					Train	*Test*	
1	1056	1990	1‰	0,58	0,54	0,34	4
2			**1%**	**0,02**	**0,97**	**0,47**	**77**
3			10%	0,01	0,95	0,50	2002
4	3914	4551	1‰	0,08	0,93	0,57	169
5			**1%**	**0,06**	**0,94**	**0,57**	**2959**
6			10%	0,06	0,94	0,57	61375

Dapat dilihat pada Tabel 5.6 hasil pengujian variasi *neuron* pada *hidden layer*. Untuk berkas data ke-2, nilai akurasi baik untuk data *training* dan data *testing* tidak berubah secara signifikan seiring bertambahnya jumlah *neuron*. Yang awalnya dengan 5 *neuron* sistem klasifikasi hanya memiliki akurasi 0,93 pada data *training*, dengan 46 *neuron* akurasi juga berada di sekitar angka yang sama, yakni 0,94. Sama halnya dengan aplikasi *neuron* sebanyak 455. Yang berbeda dari ketiga uji coba untuk berkas ke-2 adalah lama waktu *training* yang dibutuhkan. Semakin banyak *neuron* pada *hidden layer* yang digunakan, semakin lama juga waktu yang dibutuhkan, grafik dapat dilihat pada Gambar 5.2.

Pada berkas ke-1, nilai akurasi meningkat, dari 0,54 menjadi 0,97; yakni hingga dua kali lipatnya, ketika jumlah *neuron* ditambah. Berbeda dengan berkas data ke-2 yang tidak memiliki pengaruh terhadap akurasi. Namun, jika dilihat hasil dengan 199 *neuron*, terbukti bahwa jumlah *neuron* tidak mempengaruhi akurasi, melainkan lama *training*. Dapat diasumsikan bahwa 2

neuron merupakan angka yang terlalu kecil sehingga tidak mendukung nilai akurasi.

Dari hasil uji coba skenario dua, disimpulkan bahwa jumlah *neuron* pada *hidden layer* sebanyak 1% dari jumlah *neuron* pada *input layer* menjadi opsi yang terbaik karena sudah menghasilkan nilai akurasi yang tinggi, waktu *training* yang wajar, serta bukan angka yang terlalu sedikit untuk menentukan jumlah *neuron*.



Gambar 5.2 Grafik Uji Coba Variasi Jumlah *Neuron*

5.6. Hasil Pengujian dan Analisa Berdasarkan Variasi Maksimum Nilai *Epoch*

Skenario ketiga dan terakhir adalah analisa dan pengujian berdasarkan variasi maksimum nilai *epoch* yang diaplikasikan pada sistem klasifikasi *tweet* resolusi. Berdasarkan tampilan antarmuka pengguna, masukan nilai yang dibutuhkan di antaranya persentase pembagian data *training* dan data *testing*. Untuk nilai ini akan ditentukan secara *default*, yakni 10% untuk seluruh uji coba skenario ini. Untuk nilai *learning rate*, secara *default* menggunakan hasil terbaik dari skenario uji coba sebelumnya,

yakni 0,10. Dan, untuk jumlah *neuron* pada *hidden layer* akan diaplikasikan sebanyak 1% dari jumlah *neuron* pada *input layer* agar dapat menghasilkan kombinasi terbaik di akhir skenario uji coba Tugas Akhir ini.

Uji coba dengan variasi maksimum nilai *epoch* akan diaplikasikan pada dua jenis berkas seperti dua skenario uji coba sebelumnya. Dan, untuk variasinya maksimum nilai *epoch* adalah 1, 5, 10, 50, dan 100 *epoch*.

Tabel 5.7 Hasil Uji Coba Variasi Maksimum *Epoch*

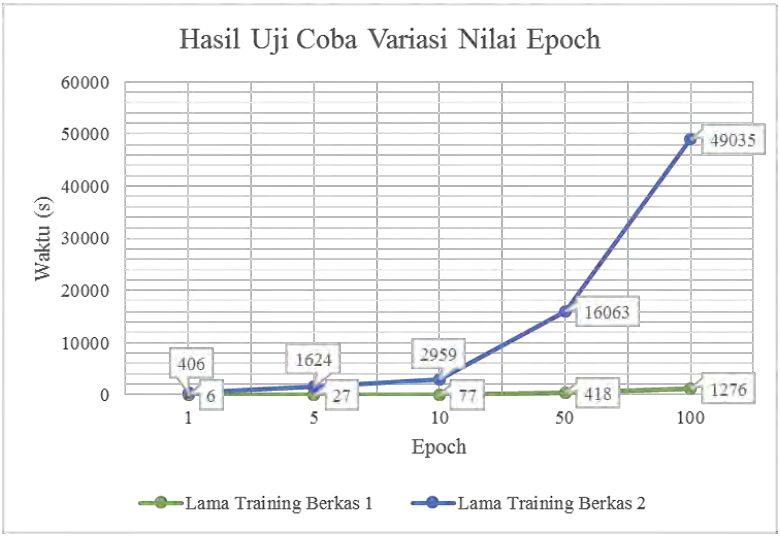
No	Data	Fitur	*Epoch*	MSE	Akurasi (%)		Lama *Train* (*s*)
					Train	*Test*	
1	1056	1990	**10**	**0,02**	**0,97**	**0,47**	**77**
2			50	0,01	0,97	0,46	418
3			100	0,01	0,97	0,42	1276
4	3914	4551	**10**	**0,06**	**0,94**	**0,57**	**2959**
5			50	0,06	0,94	0,57	16063
6			100	0,06	0,94	0,55	49035

Dapat dilihat pada Tabel 5.7 hasil pengujian variasi maksimum nilai *epoch*. Yang terjadi pada berkas data ke-1 dan berkas ke-2 terhadap variasi maksimum nilai *epoch* adalah tidak banyak berpengaruh. Jika dilihat dari perubahan *epoch* menjadi 10, akan nampak perubahan yang cukup signifikan. Namun, perubahan *epoch* di atas 10 hingga 100 tidak menunjukkan perubahan. Ini membuktikan bahwa *epoch* tidak mempengaruhi akurasi, semakin besar *epoch*, akurasi bisa jadi tetap.

Namun, jika nilai *epoch* terlalu kecil untuk diaplikasikan, tentunya akan mempengaruhi akurasi. Hal ini membuktikan bahwa sistem membutuhkan kesempatan sedikit lebih lama untuk mempelajari data.

Di sisi lain, data yang berubah seiring berubahnya maksimum *epoch* adalah lama waktu *training*. Semakin banyak *epoch* yang diaplikasikan, semakin lama waktu yang dibutuhkan. Dengan *epoch* sebanyak 10 pada berkas ke-1, *training* membutuhkan

waktu sekitar 80 detik. Begitu nilai *epoch* ditambah menjadi 50, lama *training* juga bertambah lima kali lipatnya menjadi 405 detik. Hal ini juga berlaku pada berkas data ke-2. Grafik perubahan lama waku *training* ditunjukkan pada Gambar 5.3.



Gambar 5.3 Grafik Uji Coba Variasi Nilai Maksimum *Epoch*

Hasil skenario uji coba ini, disimpulkan 10 menjadi maksimum *epoch* yang paling karena mampu memberikan akurasi yang tinggi. Selain itu juga mempertimbangkan lama waktu *training* yang singkat, serta cukup rasional.

5.7. Hasil Pengujian dan Analisa Berdasarkan Variasi Persentase Data Set

Skenario keempat adalah melakukan pengujian terhadap perbedaan persentase pada data set. Ketika nilai persentase dimasukkan, ini artinya untuk menentukan seberapa banyak data untuk data *testing* dari data set utama. Pada skenario ini, akan dilakukan tiga uji coba pada dua jenis berkas, yaitu 80%, 50%, 20%, dan 10% data *testing*. Hal ini bertujuan apakah persentase

perbandingan data *training* dan data *testing* mempengaruhi akurasi, serta mengetahui makminal nilai perbandingan untuk menentukan kedua data. Pada Tabel 5.8 ditunjukkan hasil pembagian data set ke dalam data *training* dan data *testing* sesuai dengan nilai persentase yang telah ditentukan. Sedangkan, Tabel 5.9 menunjukkan hasil akurasi dari skenario yang telah dilakukan seperti skenario pada subbab-subbab sebelumnya.

Tabel 5.8 Hasil Pembagian Data Set Variasi Persentase

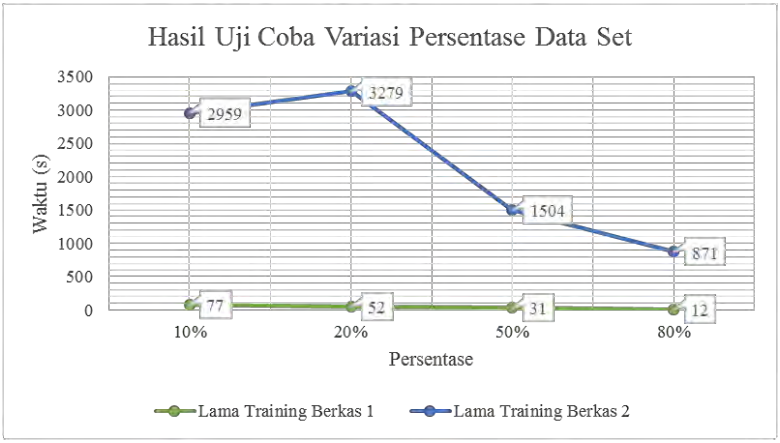
No	Data	Fitur	Persentase	Jumlah Data	
				Training	*Testing*
1	1056	1990	10%	951	105
2			20%	845	211
3			50%	528	528
4			80%	212	844
5	3914	4551	10%	3523	391
6			20%	3132	782
7			50%	1957	1957
8			80%	783	3131

Tabel 5.9 Hasil Uji Coba Perbandingan Persentase Data Set

No	Data	Fitur	Persen	MSE	Akurasi (%)		Lama *Train* (*s*)
					Train	*Test*	
1	1056	1990	10%	0,02	0,97	0,47	77
2			20%	0,01	0,98	0,51	52
3			50%	0,02	0,97	0,48	31
4			80%	0,35	0,70	0,32	12
5	3914	4551	10%	0,06	0,94	0,57	2959
6			20%	0,06	0,94	0,58	3279
7			50%	0,04	0,94	0,55	1504
8			80%	0,03	0,92	0,50	871

Ditunjukkan pada Tabel 5.9, berkas 1 dengan 1990 fitur memiliki nilai akurasi yang tinggi pada uji coba dengan data

testing diambil 20% dari data set. Di mana pada uji coba baris ke-2 menunjukkan akurasi data *training* sebesar 0,98. Sedangkan pada berkas ke-2 dengan 4551 fitur, terdapat tiga uji coba yang memiliki hasil sama, yakni dengan persentase 10%, 20%, dan 50%. Nilai akurasi *training* mencapai 0,94. Namun jika dilihat pada akurasi data *testin*, yaitu 0,58, maka persentase sebesar 20% untuk berkas ke-2 dianggap yang paling baik. Namun, jika dilihat pada lama waktu *training*, persentase 10% lebih unggul karena lebih rendah. Sedangkan, nilai *error* terkecil dimiliki oleh persentase 50%.



Gambar 5.4 Grafik Uji Coba Variasi Persentase Data Set

Tidak hanya mempengaruhi besar nilai akurasi, pembagian persentase data *training* dan data *testing* juga mempengaruhi lama waktu *training*. Pada Tabel 5.9 bahwa semakin besar persentase yang diberikan pada data *training*, maka lama waktu *training* akan membutuhkan waktu semakin lama. Grafik hasil uji coba dapat dilihat pada Gambar 5.4, di mana menunjukkan nilai penurunan dari uji coba pertama hingga uji coba keempat. Persentase pada grafik menujukkan persentase data *testing*. Semakin besar persentase, semakin sedikit waktu yang diperlukan karena jumlah data *training* juga semakin sedikit.

5.8. Hasil Pengujian dan Analisa Kombinasi Variabel Klasifikasi

Tabel 5.10 Hasil Uji Coba Kombinasi Variabel 1056 Data

No	Variabel	MSE	Akurasi (%)		Lama *Training* (*s*)
			Train	*Test*	
1	*Learning Rate* : 0,05 *Neuron* : 1% Persentase : 10%	0,07	0,96	0,50	80
2	*Learning Rate* : 0,10 *Neuron* : 1% Persentase : 10%	0,02	0,97	0,47	87
3	*Learning Rate* : 0,05 *Neuron* : 1% Persentase : 20%	**0,01**	0,95	**0,54**	**52**
4	*Learning Rate* : 0,10 *Neuron* : 1% Persentase : 20%	**0,01**	**0,98**	0,52	**52**

Empat uji coba yang telah dilakukan pada sebelumnya menghasilkan variabel dengan nilai-nilai ideal yang dapat diaplikasikan pada sistem klasifikasi *tweet* resolusi tugas akhir ini. Maka dari itu, skenario berikutnya dilakukan untuk mengetahui kombinasi variabel yang menghasilkan nilai keluaran terbaik, nilai *error*, akurasi data *training* dan data *testing*, serta lama waktu yang dibutuhkan untuk proses *training*. Untuk variabel kombinasi akan digunakan di antaranya adalah nilai *learning rate*, jumlah *neuron*, maksimum *epoch*, serta pembagian data set menjadi data *training* dan data *testing*. Jumlah *neuron* yang digunakan adalah *default* pada uji coba subbab ini, yakni 1%, yang didapat dari uji coba subbab 5.5. Sedangkan, nilai *learning rate* yang digunakan adalah dua nilai, yaitu 0,05 dan 0,10, karena pada subbab 5.4 kedua nilai tersebut memiliki nilai keluaran yang sangat mirip. Untuk persentase pembagian data set juga diaplikasikan dua nilai, yakni 10% dan 20%. Seperti dapat dilihat pada subbab 5.7, kedua nilai

memiliki keluaran yang sangat mirip dan bisa dikatan sebagai dua terbaik.

Tabel 5.10 hasil uji coba ketiga variabel pada 1056 data. Nilai *error* dan lama waktu *training* terendah dihasilkan oleh uji coba dengan persentase sebesar 20% dengan kombinasi nilai *learning rate* 0,05 dan 0,10. Akurasi *training* terbesar dimiliki uji coba dengan *learning rate* 0,10; sedangkan untuk akurasi *testing* adalah 0,05. Kombinasi paling ideal belum bisa ditentukan, karena analisa belum dilakukan pada berkas ke-2 dengan 3914 data.

Tabel 5.11 Hasil Uji Coba Kombinasi Variabel 3914 Data

No	Variabel		MSE	Akurasi (%)		Lama *Training* (*s*)
				Train	*Test*	
1	*Learning Rate* *Neuron* Persentase	: 0,05 : 1% : 10%	0,07	**0,94**	0,55	3497
2	*Learning Rate* *Neuron* Persentase	: 0,10 : 1% : 10%	**0,06**	**0,94**	0,57	**2959**
3	*Learning Rate* *Neuron* Persentase	: 0,05 : 1% : 20%	**0,06**	**0,94**	**0,58**	4461
4	*Learning Rate* *Neuron* Persentase	: 0,10 : 1% : 20%	**0,06**	**0,94**	**0,58**	3280

Jika dilihat pada Tabel 5.11 tidak banyak perubahan nilai keluaran hasil kombinasi variabel. Karena pada berkas ke-1 dengan 1056 data memiliki persentase paling ideal 20%, maka pada berkas ke-2 akan dipilih 20% juga sebagai persentase paling ideal. Pada uji coba dengan persentase 20%, hasil nilai *error*, akurasi data *training* dan data *testing*, adalah sama. Yang berbeda adalah pada lama waktu *training*. Dan, nilai terkecil dimiliki oleh data dengan nilai *learning rate* 0,10.

5.9. Analisa Hasil Klasifikasi Data

Berdasarkan hasil uji coba pada subbab-subbab sebelumnya, maka didapatkan nilai ideal untuk variabel-variabel yang dibutuhkan dalam menjalankan sistem klasifikasi teks *tweet* pada tugas akhir ini. Variabel-variabel tersebut di antaranya adalah nilai *learning rate*, jumlah *neuron* pada *hidden layer*, maksimum *epoch*, serta pembagian data *training* dan data *testing* dengan metode *hold-out*. Uji coba kali ini dilakukan dengan variabel yang paling ideal untuk mengetahui hasil klasifikasi data, baik pada tahap *training* maupun tahap *testing*, yakni 0,10 untuk *learning rate*, 1% jumlah *neuron*, 10 *epoch*, dan 20% untuk data *testing*.

Tabel 5.12 Hasil Pengujian dengan Variabel Paling Ideal

No	Data	Fitur	MSE	Akurasi (%)		Lama *Training* (*s*)
				Train	*Test*	
1	1056	1990	0,01	0,98	0,52	52
2	3914	4551	0,06	0,94	0,58	3280

Dari hasil uji coba, dapat dilihat pada Tabel 5.12 akurasi masih terhitung cukup rendah, hanya berkisar sekitar 50%. Untuk akurasi tahap *training* dihasilkan nilai yang tinggi karena prediksi dilakukan menggunakan model jaringan yang dihasilkan oleh data itu sendiri. Sedangkan, nilai akurasi tahap *testing* terbilang cukup rendah karena tidak semua fitur yang memiliki nilai pada data *testing* mendapat bobot jaringan yang sesuai saat tahap *training*. Di mana pada data *training* fitur yang dimaksud tidak memiliki nilai pembobotan awal yang tinggi selayaknya seperti pada data *testing*.

Tabel 5.13 menunjukkan hasil klasifikasi data pada berkas ke-1 dengan 1056 data. Beris pertama Tabel 5.13 menjelaskan bahwa 132 data dengan target kelas ke-1 diprediksikan dengan tepat ke kelas ke-1 setelah tahap *training*. Masih pada baris ke-2, terdapat satu data yang terjadi misklasifikasi. Data tersebut memiliki target kelas ke-2, yakni *Finance*, namun setelah melalui tahap *training* data diprediksikan sebagai data kelas ke-1, yakni kelas *Career &*

Education. Misklasifikasi terbesar terjadi pada kelas ke-5 *Recreation & Leisure* yang diprediksikan sebagai *Career & Growth*, sebanyak 18 data yang salah. Dan, untuk hasil klasifikasi data tahap *testing* dapat dilihat pada Tabel 5.14.

Tabel 5.13 Tabel Kebenaran Tahap *Training* Berkas ke-1 dengan 1056 Data

Kelas		Prediksi						Total
		1	2	3	4	5	6	
Target	1	138	0	0	0	0	0	138
	2	1	138	0	0	0	0	139
	3	0	0	139	0	0	0	139
	4	0	1	0	131	0	0	132
	5	**18**	0	0	0	136	0	154
	6	0	0	0	1	0	142	143
Total		157	139	139	132	136	142	**845**

Tabel 5.14 Tabel Kebenaran Tahap *Testing* Berkas ke-1 dengan 1056 Data

Kelas		Prediksi						Total
		1	2	3	4	5	6	
Target	1	19	0	3	5	4	1	32
	2	1	25	1	1	0	1	29
	3	3	0	18	2	4	2	29
	4	2	5	4	11	3	6	31
	5	**11**	3	3	**11**	19	2	49
	6	4	0	4	**11**	4	16	39
Total		40	33	33	41	34	28	**209**

Selanjutnya akan ditampilkan hasil klasifikasi data untuk berkas ke-2 dengan 3914 data dalam bentuk tabel kebenaran. Untuk mengetahui detail misklasifikasi uji coba berkas ke-2, dapat dilihat pada Tabel 5.15 untuk tahap *training* dan Tabel 5.16 untuk tahap *testing*. Jika diperhatikan pada tabel kebenaran pada Tabel 5.15, data dengan kelas ke-5, yakni *Recreation & Leisure*, merupakan kelas dengan jumlah misklasifikasi data paling tinggi.

Di mana hal ini terjadi sama persis dengan uji coba pada 1056 data. Kesalahan klasifikasi pada tahap *training* berpengaruh besar terhadap tahap *testing* dalam memprediksi, karena model jaringan itu sendiri tidak pintar dalam mengenali pola kelas *Personal Growth*, sesuai yang ditunjukkan pada Tabel 5.16 yang bertanda merah.

Tabel 5.15 Tabel Kebenaran Tahap *Training* Berkas ke-2 dengan 3914 Data

Kelas		Prediksi						Total
		1	2	3	4	5	6	
Target	1	149	0	1	0	0	2	152
	2	0	131	1	3	0	0	135
	3	1	0	636	10	5	0	652
	4	**106**	2	7	1375	17	20	1527
	5	1	0	0	7	327	3	338
	6	0	0	0	5	0	323	328
Total		257	133	645	**1400**	349	348	**3132**

Tabel 5.16 Tabel Kebenaran Tahap *Testing* Berkas ke-2 dengan 3914 Data

Kelas		Prediksi						Total
		1	2	3	4	5	6	
Target	1	17	1	7	9	5	0	39
	2	2	27	2	5	0	0	36
	3	6	2	104	21	12	6	151
	4	**33**	7	**52**	254	**44**	**43**	433
	5	8	1	2	17	29	6	63
	6	2	1	6	15	12	24	60
Total		68	39	173	**321**	102	79	**782**

Jika diperhatikan pada Tabel 5.16, kelas ke-4 merupakan kelas dengan hasil prediksi terbanyak. Hal ini mungkin terjadi karena jumlah data pada kelas ke-4 merupakan yang paling banyak dan paling dominan dari kelas-kelas yang lain. Sehingga, ketika sistem melakukan prediksi terdapat kecondongan terhadap kelas

ke-4. Hal ini menunjukkan bahwa adanya dominasi data pada kelas tertentu sangat berpengaruh terhadap proses prediksi, yang dapat menyebabkan prediksi lebih condong ke kelas dominan. Selain itu, ketika kelas ke-4 diprediksikan, sistem juga kebingungan dengan model yang disebabkan banyaknya data kelas ke-4. Untuk mendapatkan analisa terhadap banyaknya misklasifikasi, maka ditampilkan persebaran fitur pada masing-masing kelas, seperti yang ditunjukkan pada Tabel 5.17.

Tabel 5.17 Persebaran Fitur Masing-Masing Kelas 3914 Data

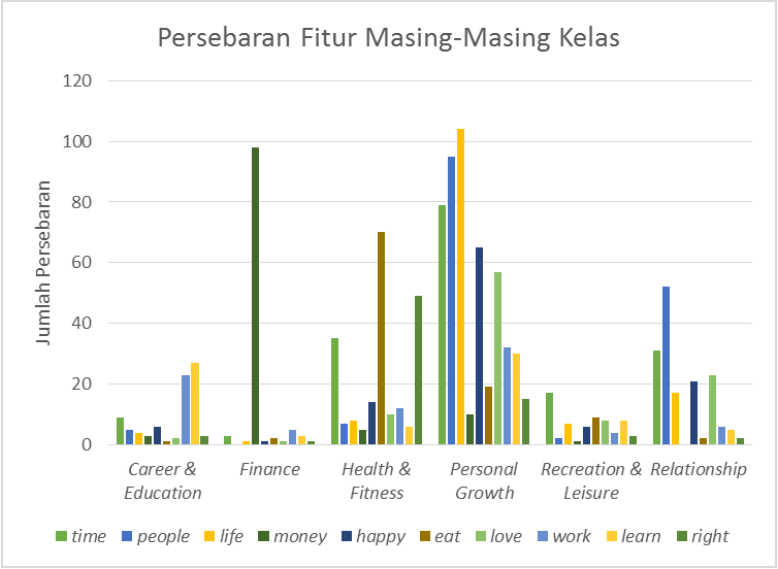
No	Fitur	Kelas						Total
		1	2	3	4	5	6	
1	*time*	9	3	35	79	17	31	174
2	*people*	5	0	7	95	2	52	161
3	*life*	4	1	8	104	7	17	141
4	*money*	3	98	5	10	1	0	117
5	*happy*	6	1	14	65	6	21	113
6	*eat*	1	2	70	19	9	2	103
7	*love*	2	1	10	57	8	23	101
8	*work*	23	5	12	32	4	6	82
9	*learn*	27	3	6	30	8	5	79
10	*right*	3	1	49	15	3	2	73
11	*friends*	0	0	4	17	8	43	72
12	*things*	1	1	0	60	2	3	67
13	*gym*	0	1	64	1	0	0	66
14	*think*	2	0	12	39	8	3	64
15	*getting*	5	0	21	22	8	4	60
16	*smoking*	0	0	59	1	0	0	60
17	*thanks*	1	51	3	2	2	0	59
18	*stay*	4	1	9	39	0	5	58

Ditunjukkan pada Gambar 5.5 merupakan grafik persebaran lima fitur terbanyak yang didapatkan dari Tabel 5.17. Dapat dilihat pada grafik, fitur *money* merupakan yang paling dominan pada kelas *Finance*, di mana sesuai dengan pemetaan secara manual.

Ketika kata ‘uang’ disebut maka yang terlintas pasti mengenai bidang keuangan. Sama halnya dengan *money*, fitur *eat* merupakan fitur paling mendominasi pada kelas *Health & Fitness*. Untuk fitur *work*, memiliki persebaran yang cukup tinggi pada kelas *Career & Education*, namun juga pada kelas *Personal Growth*. Fitur *learn* yang memiliki makna mempelajari dimiliki sangat tinggi pada kelas *Personal Growth*, di mana sesuai. Namun fitur *learn* juga sesuai jika masuk kelas *Career & Education*. Hal ini yang menjadi penyebab adanya misklasifikasi pada sistem.

Gambar 5.5 Grafik Persebaran Fitur Kelas 1056 Data

Kelemahan menggunakan data *tweet* adalah karena memiliki karakter yang sangat terbatas. Sehingga dengan kata seminimal mungkin, informasi harus tersampaikan. Dan juga, dalam media sosial pengguna tentunya tidak menggunakan kata baku. Sehingga sulit untuk diinterpretasikan menjadi fitur yang benar-benar merepresentasikan masing-masing kelas.



Gambar 5.6 Grafik Persebaran Fitur Kelas 3914 Data

LAMPIRAN

tweet_user	*tweet*	*class_id*	*class_name*
jackhar	*Must have Tech partners in 2015 @FieldViewSolu #DCIM @AnordCritical #switchgear @Seven10Software #cloudstorage #NewYearsResolution*	1	*Career & Education*
jax1213	*I need to just work on my Spanish so I can finally move here.... ‰ÏÓ•ü• _Ù÷_#NewYearsResolution _ÙÎ«*	1	*Career & Education*
_Mosby	*RT @GolTV: Bae: I want a man with a career... I'm on it! #NewYearsResolution http://t.co/yeHkT7ywzJ*	1	*Career & Education*
x_noel_kreiss_x	*My graduation photo.... I should probably take one soon... #newyearsresolution http://t.co/FgH5xHym4X*	1	*Career & Education*

Trey0781	*Should I go to community college next year or go to a 4 year university... Hmm #NewYearsResolution*	1	*Career & Education*
itsmeCampy	*2014- the year of ideas....2015 - the year of execution! #2015 #NewYearsResolution*	1	*Career & Education*
roybusinesschat	*#NewYearsResolution - that college graduates of the Class of 2015 enter into #entrepreneurship instead of a #job*	1	*Career & Education*
__KRISTOFFER__	*May I find more work onstage and on-film in 2015! #NewYearsResolution*	1	*Career & Education*
millerpod	*In honor of my #NewYearsResolution to learn Spanish #rosettastone #PCSB ESOL http://t.co/3zmp1Ol1yN*	1	*Career & Education*
DetwilerLeroux	*#motivation. Set plan. Skill set to do it. #NewYearsResolution Taking my @ThirtyOne business to the next level in 2015!*	1	*Career & Education*
leahjuster	*My #NewYearsResolution should be to make payments on my student loan that are equal to my @ULTA_Beauty orders. #platinum*	2	*Finance*

SoundtrackSci	*One #NewYearsResolution of mine is to work on my spending. Free alcohol might help one line item ;) Via @BostInno http://t.co/MSWJAXbnsX*	2	*Finance*
treachery49	*Money #NewYearsResolution*	2	*Finance*
HunnitBlkRah300	*Time to make more money less excuse #NewYearsResolution*	2	*Finance*
roybusinesschat	*#NewYearsResolution - that #neworleans experiences economic growth and worth valued at 1 billion!*	2	*Finance*
katasidy	*Great new years resolution. Learn to #budget. http://t.co/F8v3306pxF #NewYearsResolution #financialadvice #finance*	2	*Finance*
kEELONELY	*I swear all my money goes to food .. This shxt gotta stop #NewYearsResolution*	2	*Finance*
seantyoung	*#NewYearsResolution save money damnit*	2	*Finance*

RamonaRichards	*I don't wait till NYDay for resolutions. I start when the last gift is opened. 1) In 2015, I need to save more money. #NewYearsResolution*	2	*Finance*
Jonesin_13	*RT @ETizghost: I dont need new goals my broke ass still trying to accomplish the ones I set 3 years ago.. Hows that for a #NewYearsResoluti‰Û_*	2	*Finance*
RaeShae90	*I got the hair. I just need to work on the body to pull off Lara Croft next Halloween #NewYearsResolution http://t.co/Ihqh5Z7prx*	3	*Health & Fitness*
JASMINECLEMENTE	*Note to self: "Stop being lazy & get back into hot yoga." I have a monthly pass, so no excuses! #hotyoga #newyearsresolution #healthiswealth*	3	*Health & Fitness*
MizzTifferz	*I have to go do my 6 mile run. #stayingFIT #toneUP #StallionBOOTY #NewYearsResolution starts now.*	3	*Health & Fitness*

N0NST0P	*‰ÛÏ@PartiPants27: My #NewYearsResolution is to make "dem gainz" By any means necessary‰Û• bang bang muscle gainz*	3	*Health & Fitness*
Red_Inspired	*#bucketlist eat at chipotle never had it #1problem #NewYearsResolution is to go vegetarian. Panara it is, never had that either #pretentious*	3	*Health & Fitness*
loveechelsea	*RT @Lindseylelledee: Can't wait to be super clichÌ© & jump back on the fitness wagon come January! #NewYearsResolution*	3	*Health & Fitness*
Stefanipns	*http://t.co/lCV97vwVn6 : motivate yourself. #workoutorwalkout #GymTime #healthyliving #NewYearsResolution ‰Ð¼ http://t.co/GlxW6TWj38*	3	*Health & Fitness*
LouisaSmith14	*RT @BelleIsle_Rack: Just got a ton of new workout clothes in.. I'm glad because I will need some after the holidays! #NewYearsResolution ht‰Û_*	3	*Health & Fitness*

sarah052794	*RT @margaretyo: Already thinking of #NewYearsResolution for #2015. Lose weight, save money, and most importantly, get rid of bad friends.*	3	*Health & Fitness*
AllisonDar	*5 down, 25 miles to go #iwillwalk500miles #health #fitness #newyearsresolution #almostthere http://t.co/yl3zcSoWkI*	3	*Health & Fitness*
_K_E_V	*Self improvement!.. Mentally, physically, and financially. #NewYearsResolution*	4	*Personal Growth*
kasbury23	*All I want for Christmas is to be Carrie Underwood. #NewYearsResolution*	4	*Personal Growth*
Trudacious	*RT @barryjohnharper: To eat so much this Christmas by new year I can have my own boob avi #NewYearsResolution*	4	*Personal Growth*
filmya247	*RT @ashley_pg19: I'm sick of waiting on life to happen. I want to make my life worth living. _Ù• â_ÙÎÈ #life #loveyourlife #christmastime #NewYear‰Û_*	4	*Personal Growth*

KatelynHadder	*social media purging > #NewYearsResolution*	4	*Personal Growth*
Ambry_DeLeon	*RT @K1NG_JAMES_: I just wanna be as happy as the ppl that be on my chemistry book pretend to be #NewYearsResolution*	4	*Personal Growth*
hscuevas	*& too obsessed with social media.... habit I need to break #NewYearsResolution*	4	*Personal Growth*
BusinessManFlow	*#NewYearsResolution become the fun person i use to be.*	4	*Personal Growth*
KirbyLaguerre	*to actually work out and not be lazy about it #NewYearsResolution*	4	*Personal Growth*
VanessaKEccles	*Thankful that spring always comes after winter. #writerslife #NewYearsResolution http://t.co/3N4D9vZ8A8*	4	*Personal Growth*
Danna__Day	*#NewYearsResolution: I will play more #tennis in 2015. _Ù÷_*	5	*Recreation & Leisure*
zhcnsile	*1. Catch up on the 20 series I've totally missed out on. #NewYearsResolution*	5	*Recreation & Leisure*

keylamarques	*#vacation! Ahhhh I can finally relax. Work on my #NewYearsResolution ;)*	5	*Recreation & Leisure*
CindyyyRella25	*2015 lots of traveling ‰Ï•ü• #NewYearsResolution*	5	*Recreation & Leisure*
BabyAviators	*Drinking only #sriracha for 2015 #NewYearsResolution*	5	*Recreation & Leisure*
Hayesisprf	*#NewYearsResolution is for @AaronCarpenter to flipping FOLLOW ME!!!!! #FollowMeAaron*	5	*Recreation & Leisure*
mustache_party	*Planning on uploading my entire music catalog on @Spotify and @iTunesMusic in early 2015. #NewYearsResolution #EDM #Dance #Music*	5	*Recreation & Leisure*
ALLANBD	*#NewYearsResolution Read more, write more.*	5	*Recreation & Leisure*
adogg281	*my #NewYearsResolution is to watch more anime and sports. also going outside and workout. plus going to Ohio State.*	5	*Recreation & Leisure*
AmeliaMRogocka	*My New Year's resolution is to use twitter more. #Iwantobehip #NewYearsResolution*	5	*Recreation & Leisure*

Teddy_Retro	*they gotta listen tho "@K1NG_JAMES_ #NewYearsResolution teach everyone I come in contact with a positive lesson that helps them out in life"*	6	*Relationship*
classidypeace	*In 2015 I want to save all year to give a significant donation to a few charities I feel passionately about #NewYearsResolution*	6	*Relationship*
JustonBrommel	*I Am here to save the world. Damn it feels good. Daily Miracles. Join the #evolution #NewYearsResolution #resolutions What are you here 4?*	6	*Relationship*
adrianna_love	*RT @xNova_Cane: @adrianna_love oh #NewYearsResolution ? It should be to build a real friendship like you promised we'd have*	6	*Relationship*
xNova_Cane	*@adrianna_love oh #NewYearsResolution ? It should be to build a real friendship like you promised we'd have*	6	*Relationship*
Salegrino	*#NewYearsResolution fight Jack everytime I see him*	6	*Relationship*

SweetAssTiffene	*Not Fucking with ANYTHING!!! Friends, Family, Job..NOTHING_Ù_Ç Where I'm not APPRECIATED_ÙÕø_ÙÕø_ÙÕø #YearEndResolution #Change #NewYearsResolution*	6	*Relationship*
gemaruiz01	*#NewYearsResolution 1. To become best friends with CAMERON Alexander Dallas _Ù÷_‰□_•ü□ @camerondallas _Ù÷_‰□_•ü□*	6	*Relationship*
KMLapham	*A #NewYearsResolution should probably be something more substantial than "tweet more", huh? Asking for a friend.*	6	*Relationship*
Airnie21	*It's starting to look like the only way to get hella likes on FB is to have kids. Oh well, #NewYearsResolution*	6	*Relationship*

BAB VI
KESIMPULAN DAN SARAN

Bab ini berisi tentang kesimpulan yang diperoleh selama pengerjaan Tugas Akhir ini. Selain itu, juga terdapat beberapa saran terhadap Tugas Akhir ini yang diharapkan bisa membuat Tugas Akhir ini menjadi lebih baik lagi.

6.1. Kesimpulan

Kesimpulan yang diperoleh berdasarkan uji coba dan evaluasi yang telah dilakukan pada Tugas Akhir antara lain:

1. Nilai *learning rate* mempengaruhi nilai akurasi sistem klasifikasi. Pada Tugas Akhir ini perubahan akurasi melambat ketika nilai *learning rate* yang digunakan adalah sama dengan atau lebih besar dari 0,10.
2. Pada dasarnya jumlah *neuron* pada *hidden layer* tidak berpengaruh terhadap nilai akurasi sistem. Lebih banyak berpengaruh pada lama waktu *training*. Semakin banyak *neuron* maka akan semakin lama waktu yang dibutuhkan. Karena pasangan antar *neuron* menjadi berkali lipat lebih banyak. Namun, jika jumlah *neuron* pada *hidden layer* terlalu sedikit, dapat mempengaruhi akurasi sistem klasifikasi. Pada kasus Tugas Akhir ini persentase jumlah *neuron* terbaik yang digunakan adalah 1%.
3. Maksimum *epoch* tidak banyak berdampak pada nilai akurasi, lebih ke lama waktu *training*. Semakin besar maksimum *epoch*, semakin lama waktu yang dibutuhkan. Namun, jika maksimum *epoch* yang dimasukkan terlalu kecil, hal ini dapat mempengaruhi nilai akurasi.
4. Kesamarataan jumlah dokumen pada masing-masing kelas juga merupakan faktor penting yang dapat mempengaruhi hasil klasifikasi. Dengan jumlah dokumen yang sama pada masing-masing kelas, akan membantu sistem mempelajari dan melakukan prediksi dengan lebih adil. Kelas dengan jumlah dokumen yang lebih banyak dibanding kelas yang

lebih sedikit dapat menyebabkan kecondongan sistem pada kelas tertentu. Sehingga, akan mempengaruhi nilai akhir akurasi.

5. Algoritma *Backpropagation* yang diaplikasikan pada tugas akhir ini menghasilkan akurasi *testing* tertinggi sebesar 58% dengan kombinasi variabel adalah 0,10 untuk nilai *learning rate*, jumlah *neuron* pada *hidden layer* adalah 1% dari jumlah *neuron* pada *input layer*, *epoch* maksimum adalah 10, dan persentase jumlah data *testing* adalah 20%.

6.2. Saran

Terdapat beberapa saran terkait Tugas Akhir ini yang diharapkan bisa membuat Tugas Akhir ini menjadi lebih baik. Saran-saran tersebut antara lain:

1. Pengembangan metode ekstraksi fitur teks untuk mengurangi jumlah fitur yang terlalu banyak.
2. Pengembangan metode pada pembobotan. Pada kasus *tweet*, sebuah kata cenderung hanya muncul beberapa kali, tidak sebanyak dalam paragraf. Perlu adanya pembobotan yang tepat untuk masing-masing fitur.
3. Pengembangan Algoritma *Backpropagation* untuk menangani kasus apabila jumlah dokumen pada masing-masing kelas tidak sama atau tidak rata, yang menyebabkan dominasi atau kecondongan pada kelas tertentu saat proses *training*.

DAFTAR PUSTAKA

D. & Gurusamy, V., 2015. *Preprocessing Techniques for Text Mining.* Podi, s.n.

Eldira, H., K, E. . M. & M, N. R., 2010. Web Mining untuk Pencarian Dokumen Bahasa Inggris. *Politeknik Elektronika Negeri Surabaya.*

Karmayasa, O. & Mahendra, I. B., 2012. Implementasi Vector Space Model dan Beberapa Notasi Metode Term Requency Inverse Document Frequency (Tf-Idf) pada Sistem Temu Kembali Infomasi. *Jurnal Elektronik Ilmu Komputer Universitas Udayana,* 1(1).

Manroe, M., 2014. *Daftar Istilah Penting Dalam Twitter dan Artinya.* [Online] Available at: maxmanroe.com/daftar-istilah-penting-dalam-twitter-dan-artinya.html [Diakses 20 Mei 2016].

Mulyawati, N. Z. D. L., 2012. *Machine Learning.* [Online] Available at: nita_zelfia-fst09.web.unair.ac.id/artikel_detail-44883-Umum-Machine%20Learning.html [Diakses 1 April 2016].

Prasetiyo, I., 2015. *Jenis-Jenis Neural Network.* [Online] Available at: tukarpengetahuan.com/2015/06/jenis-jenis-neural-network.html [Diakses 16 Maret 2016].

Saadah, M. N., Atmagi, R. W., Rahayu, D. S. & Arifin, A. Z., 2012. Sistem Temu Kembali Dokumen Teks dengan Pembobotan Tf-Idf Dan LCS. *Jurusan Teknik Informatika, Fakultas Teknologi Informasi , Institut Teknologi Sepuluh Nopember.*

Sano, D., 2013. *Metode-metode dalam Data Mining - Seri Data Mining for Business Intelligence.* [Online] Available at: beritati.blogspot.co.id/2013/08/metode-metode-dalam-data-mining-seri.html [Diakses 2016 April 15].

Setiawan, S. I. A., 2011. *Penerapan Jaringan Saraf Tiruan Metode Backpropagation Menggunakan VB 6,* Tangerang: SofTech.

Tenuto, J., 2015. *From Getting in Shape to Finally Meeting One Direction: A Roundup of 2015 New Year's Resolutions.* [Online] Available at: crowdflower.com/blog/new-years-resolutions-2015 [Diakses 16 Maret 2016].

Usmaida, A., 2007. Web Mining untuk Pencarian Berdasarkan Kata Kunci dengan Teknik Clustering. *Tugas Akhir Jurusan Teknologi Informasi Politeknik Elektronika Negeri Surabaya.*

Wikipedia, 2006. *Twitter.* [Online] Available at: id.wikipedia.org/wiki/Twitter [Diakses 11 Maret 2016].

Wikipedia, 2013. *Backpropagation.* [Online] Available at: en.wikipedia.org/wiki/Backpropagation [Diakses 20 Maret 2016].

Wikipedia, 2013. *Jaringan saraf tiruan.* [Online] Available at: id.wikipedia.org/wiki/Jaringan_saraf_tiruan [Diakses 20 Maret 2016].

BIODATA PENULIS

Penulis lahir di Banyuwangi, 18 Januari 1995. Penulis telah menempuh pendidikan dasar di SDN 2 Ketapang, kemudian untuk pendidikan menengah pertama di SMPN 1 Banyuwangi dan di jenjang menengah atas di SMAN 1 Glagah. Penulis memiliki ketertarikan pada bidang komputer dan teknologi sehingga penulis memutuskan untuk mengambil pendidikan sarjana S1 di Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember Surabaya. Selama kuliah, penulis aktif berorganisasi sebagai Sekretaris BEM FTIf 2014/2015, staf Departemen Pengembangan Sumber Daya Mahasiswa Himpunan Mahasiswa Teknik Computer-Informatika (HMTC) ITS 2013/2014 dan BEM FTIf 2013/2014.

Penulis dalam menyelesaikan pendidikan S1 mengambil rumpun mata kuliah (RMK) Komputasi Cerdas dan Visi serta memiliki ketertarikan di bidang Basis Data, Pemrograman *Web* dan *Mobile*, serta *Data Mining*. Penulis dapat dihubungi melalui surel: linggarjuwita@gmail.com.