Multi-Label Emotion Classification using Naïve Bayes (study case: Reddit Comments)

Proyek Pemrosesan Bahasa Alami

Oleh :

11S18001	Efrica C. Situmeang
11S18007	Aldi Irvan Siagian
11S18013	Grace Noelia Simorangkir
11S18040	Anjel Riska Pardede
11S18041	Grace Widya Simanjuntak
11S18048	Romual Naibaho

11S4037 - Pemrosesan Bahasa Alami

Fakultas Informatika dan Teknik Elektro

Institut Teknologi Del

2021

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1
PENDAHULUAN

Bab ini menjelaskan mengenai latar belakang topik penelitian, tujuan penelitian, manfaat penelitian, ruang lingkup penelitian.

1.1 Latar Belakang

Emosi adalah pola reaksi yang kompleks, yang melibatkan pengalaman, perilaku, dan fisiologis, di mana seorang individu mencoba untuk menangani masalah atau peristiwa yang signifikan secara pribadi. Dalam kehidupan sehari-hari, penyampaian emosi dapat disampaikan secara non-verbal menggunakan ekspresi wajah maupun verbal berupa tingkah laku. Emosi bersifat subjektif dan temporer yang muncul atau dipicu oleh stimulus seperti perlakuan dari orang sekitar atau lingkungannya.

Deteksi emosi adalah masalah umum yang berpotensi pada peningkatan seperti interaksi manusia atau komputer yang lebih baik. Setiap orang tentu pernah mengalami berbagai emosi. Analisis emosi adalah tindakan untuk menentukan sikap terhadap target atau topik. Sikap dapat berupa polaritas (positif atau negatif) atau keadaan emosional seperti senang, marah, atau sedih. Deteksi emosi pada teks dapat dianggap sebagai masalah klasifikasi pada konsep pemrosesan bahasa alami (NLP) dan Machine learning. Menurut hasil survey yang telah dilakukan pada tahun 2014 - 2018 terdapat 6 dari 10 penelitian yang melakukan penelitian tentang deteksi emosi pada teks (Al-Saaqa et al., 2018). Deteksi emosi pada teks telah menarik minat yang cukup besar karena penerapannya ke berbagai domain, termasuk klasifikasi teks, klasifikasi adegan dan video, dan bioinformatika terutama untuk kepentingan analisis emosi.

Pada penelitian ini akan menggunakan Reddit Comments (komentar dari media sosial Reddit) yang terdiri dari 27 kategori label emosi. Reddit merupakan situs web hiburan dan juga berita tempat user dapat berkontribusi dalam bentuk postingan pranala atau teks. Reddit menjadi salah satu media sosial yang populer di benua Amerika dan Eropa. Para pengguna Reddit biasanya akan membuat postingan yang mengusung tema seperti politik, ekonomi, budaya, agama, organisasi, atau tema lainnya dan akan menjadi topik perbincangan banyak orang. Semakin hari, ulasan pada Reddit yang disampaikan oleh pengguna Reddit semakin bertambah dan semakin beragam seperti komentar, kritik, opini yang positif, negatif atau netral. Namun tidak semua pengguna dapat memahami apa bentuk emosi yang ada pada postingan maupun ulasan tersebut. Karena bentuk emosi sangat sulit diidentifikasi jika hanya melihat tulisan saja. Oleh sebab itu penelitian ini dilakukan untuk mengklasifikasi

an emosi dari berbagai komentar atau ulasan yang ada sehingga pengguna dapat mengetahui bagaimana bentuk emosi yang ada dalam ulasan tersebut.

Deteksi emosi dilakukan dengan pendekatan klasifikasi emosi. Klasifikasi adalah proses dengan menggabungkan atau mengelompokkan dua atau lebih data yang memiliki kesamaan pada suatu kriterianya. Terdapat beberapa metode yang dapat digunakan untuk melakukan klasifikasi pada teks, diantaranya Support Vector Machine, Naïve Bayes, K-Nearest Neighbor, Multinomial Naïve Bayes, Multi Task Regression, Algoritma Incremental Regression, dan lain-lain. Sedangkan untuk metode fitur ekstraksi juga terdapat beberapa yang sering digunakan, diantaranya TF-IDF, N-Gram, Word Embedding, dan lain-lain. Namun pada penelitian ini beberapa pendekatan atau metode yang akan digunakan dalam melakukan fitur ekstraksi yaitu TF-IDF yang kemudian diklasifikasikan dengan metode Naïve Bayes. Berdasarkan penelitian sebelumnya fitur TF-IDF memberikan hasil yang lebih baik sekitar 3% - 4% jika dibandingkan dengan fitur N-Gram (Ahuja et al., 2019). TF-IDF merupakan teknik yang paling sederhana

sehingga mudah untuk melakukan pengecekan corpus yang besar dan juga dapat menghitung banyak istilah yang ada dalam sebuah dokumen. Algoritma pengklasifikasian Naive Bayes adalah salah satu pengklasifikasian statistik, pengklasifikasian ini dapat memprediksi probabilitas anggota kelas data yang akan masuk ke kelas tertentu, menurut perhitungan probabilitas. Klasifikasi ini menunjukkan akurasi dan kecepatan yang tinggi bila diterapkan kedalam database yang besar. Metode ini sering digunakan untuk memecahkan masalah di bidang Machine Learning karena metode ini memiliki tingkat akurasi yang tinggi dengan perhitungan sederhana.

1.2 Rumusan Masalah

Berdasarkan latar belakang masalah yang telah disampaikan sebelumnya, dapat ditarik rumusan masalah yang akan diselesaikan sebagai berikut.

1. Bagaimana mengklasifikasikan emosi berdasarkan data komentar menggunakan algoritma Naive Bayes.

1.3 Tujuan

Tujuan penelitian dalam Tugas Pemrosesan Bahasa Alami ini adalah sebagai berikut:

1. Melakukan implementasi Naive Bayes dalam mengklasifikasikan emosi berdasarkan data komentar.

1.4 Manfaat Penelitian

Manfaat penelitian dalam Tugas Pemrosesan Bahasa Alami ini adalah sebagai berikut:

1. Penelitian ini diharapkan mampu melakukan klasifikasi emosi berdasarkan kalimat yang diberikan dengan memberi label pada setiap kelas emosi sesuai dengan dataset yang diberikan.

1.5 Ruang Lingkup Penelitian

Ruang Lingkup dari Tugas Pemrosesan Bahasa Alami ini adalah sebagai berikut:

1. Data yang diolah dikumpulkan dari *website* reddit.
2. Penelitian ini melakukan normalisasi hanya dalam bentuk huruf.
3. Penelitian ini menerima *inputan* dalam bentuk kalimat yang mempunyai kata singkatan bahasa Inggris.
4. Penelitian ini akan melakukan normalisasi kata singkatan yang di-*input* oleh *user*.

BAB 2
TINJAUAN PUSTAKA

Pada bab ini diuraikan dasar teori yang didapatkan dari pustaka yang relevan dengan kajian Penelitian.

2.1 Emosi

Emosi manusia adalah suatu keadaan jiwa kompleks yang terdiri dari tiga komponen berbeda yaitu pengalaman subjektif, respon psikologis dan respon ekspresif. Untuk lebih memahami emosi, peneliti melakukan identifikasi dan klasifikasi pada beberapa jenis emosi. Pada 1972, psikologis Paul Ekman menyatakan bahwa terdapat enam emosi dasar yang terdapat pada budaya manusia secara universal. Ekman melakukan penelitian di seluruh pelosok dunia dan menemukan enam emosi yang sama melalui ekspresi wajah. Enam emosi dasar adalah bahagia, sedih, terkejut, marah, jijik dan takut. Ekman menemukan bahwa enam ekspresi ini bersifat universal kepada seluruh manusia di dunia.

2.2 Natural Language Processing

Natural Language Processing (NLP) atau pemrosesan bahasa alami merupakan bidang dari ilmu *computer science.* NLP adalah kajian tentang struktur yang berkaitan dengan memberi komputer untuk memahami teks dan kata-kata yang diucapkan dengan cara yang sama seperti yang dapat dilakukan manusia. NLP menggabungkan linguistik komputasi pemodelan bahasa manusia berbasis aturan dengan model statistik, pembelajaran mesin, dan pembelajaran mendalam. NLP menggerakkan program komputer yang menerjemahkan teks dari satu bahasa ke bahasa lain, menanggapi perintah lisan, dan meringkas teks dalam jumlah besar dengan cepat bahkan secara real time.

2.3 Data Preprocessing

Data preprocessing adalah suatu proses yang digunakan untuk menghilangkan kata-kata atau teks yang tidak diperlukan dalam komputasi. Inti dari data *preprocessing* adalah mengubah data yang tidak terstruktur sesuai dengan kebutuhan dan proses *mining* yang dilakukan. Tahapan tahapan yang dilakukan dalam preprocessing ini ada 5 yaitu *text cleaning, case folding*, *tokenization, stopword removal, stemming.*

1. *Text cleaning*

   Proses *text cleaning* (pembersihan teks) merupakan suatu proses dalam membersihkan data yang dimiliki dari suatu komponen yang tidak diperlukan dalam proses dan tahapan selanjutnya, seperti tanda baca, data tweet yang sama (retweet), link, dan sebagainya. Sehingga, jika telah dilakukan proses penghapusan link, tanda baca (punctuation), dan tweet yang berulang (retweet), maka akan semakin memudahkan peneliti dalam proses pembersihan data selanjutnya dan juga menambah keakuratan dalam melakukan proses pengklasifikasian sentimen pada tahapan berikutnya.

2. *Case Folding*

   Case folding merupakan mengubah keseluruhan huruf menjadi huruf kecil (*lower case*). Hal ini dilakukan karena tidak semua teks konsisten dalam penggunaan huruf.

3. *Tokenizing*

   Tokenizing adalah suatu tahap untuk memenggal setiap kata yang ada dalam kalimat. Proses ini dilakukan untuk menjadikan sebuah kalimat tersebut menjadi lebih bermakna. Dan suatu kata, tanda baca, angka, simbol, dan entitas penting lain dapat disebut sebagai token. Untuk proses ini diperlukan untuk mengenali setiap kata maupun huruf yang ada pada korpus normalisasi yang diperlukan untuk proses spelling normalization.

4. *Stopword removal*

   Stopwords Removal merupakan suatu proses yang bertujuan untuk mengambil kata-kata penting dari tokenisasi. Stopwords merupakan kata-kata yang tidak mengandung sentimen atau bukan merupakan ciri dari sebuah sentimen. Misalnya ‘di’, ‘ke’, ‘dari’, ‘oleh’, dan sebagainya. Stoplist berisi kumpulan kata yang kurang relevan namun sering muncul pada sebuah dokumen atau tweet. Stoplist berisi sekumpulan stopwords.

5. *Stemming*

   Proses stemming merupakan tahapan yang mengembalikan kata-kata ke bentuk dasarnya dengan cara menghapus semua awalan (prefix), imbuhan (affix) dan akhiran (suffix). Tahapan stemming digunakan agar memperkecil banyaknya indeks yang beda dari suatu dokumen dan mengelompokkan kata yang mempunyai arti yang sama untuk bentuk kata yang beda karena adanya imbuhan. Dan proses stemming diperlukan untuk kebutuhan pada proses selanjutnya yaitu proses penentuan sentimen dengan menggunakan kamus lexicon yang menggunakan kata-kata dasar.

2.4 TF-IDF

Metode TF-IDF merupakan metode untuk menghitung bobot setiap kata yang paling umum digunakan pada *information retrieval*. Dimana *information retrieval* adalah bagian dari *computer science* yang berhubungan dengan pengambilan informasi dari dokumen-dokumen yang didasarkan pada isi dan konteks dari dokumen-dokumen itu sendiri (Melita, et al., 2018). Metode *Term Frequency Inverse Document Frequency (TFIDF)* adalah cara pemberian bobot hubungan suatu kata (*term*) terhadap dokumen. TFIDF ini adalah sebuah ukuran statistik yang digunakan untuk mengevaluasi seberapa penting sebuah kata di dalam sebuah dokumen atau dalam sekelompok kata. Untuk dokumen tunggal tiap kalimat dianggap sebagai dokumen. Frekuensi kemunculan kata di dalam dokumen yang diberikan menunjukkan seberapa penting kata itu di

dalam dokumen tersebut. Frekuensi dokumen yang mengandung kata tersebut menunjukkan seberapa umum kata tersebut. Bobot kata semakin besar jika sering muncul dalam suatu dokumen dan semakin kecil jika muncul dalam banyak dokumen. Pada TF-IDF digunakan rumus untuk menghitung bobot (W) masing-masing dokumen yang berikan:

$$Wdt = tfdt * Idft$$

dimana:

Wdt : bobot dokumen ked terhadap kata ket

tfdt : banyaknya kata yang dicari pada sebuah dokumen

Idft : Inversed Document Frequency (log (N/df))

N : total dokumen

df : banyak dokumen yang mengandung kata yang dicari.

2.5 Klasifikasi

Klasifikasi adalah masalah pemodelan prediktif yang melibatkan keluaran label kelas yang diberikan beberapa masukan. Tugas klasifikasi umumnya adalah memprediksi satu label atau target kelas. Namun klasifikasi juga mungkin melibatkan prediksi terhadap dua atau lebih label kelas. Dua atau lebih label kelas dalam kasus klasifikasi disebut sebagai Multi-Label Classification. Adapun penjelasan terkait Multi-Label Classification disampaikan dalam subbab berikut ini.

2.5.1 Multi-Label Classification

Multi-Label Classification adalah varian dari masalah klasifikasi dimana beberapa label atau target kelas dapat ditetapkan untuk setiap *instance*. Dalam multi-label classification, nol atau lebih label diperlukan sebagai output untuk setiap sampel input, dan output diperlukan secara bersamaan. Multi-Label Classification memerlukan algoritma pembelajaran mesin untuk mendukung prediksi beberapa kelas atau label. Dalam Multi-Label Classification, *training set* terdiri dari instance yang masing-masing terkait dengan satu set label, dan tugasnya adalah memprediksi set label dari *instance* yang tidak terlihat melalui analisis *instance training* dengan set label yang diketahui.

Multi-Label Classification dapat diterapkan pada kasus-kasus tertentu seperti penentuan *genre* dari sebuah film, penentuan jenis penyakit berdasarkan gejala tertentu, penentuan emosi berdasarkan text, dan lain sebagainya. Dalam penelitian ini Multi-Label Classification digunakan untuk mengklasifikasikan emosi dalam sebuah kalimat atau komentar dari sebuah media sosial.

2.5.2 Naive Bayes Classification

Naïve Bayes adalah pendekatan *statistic fundamental* yang menggunakan probabilitas dan *cost* (Yasin, 2005). *Bayes rule* adalah suatu aturan dalam menentukan nilai probabilitas dari suatu kondisi tertentu menjadi sebuah kondisi yang didefinisikan. Kaitan antara Naïve Bayes dengan metode klasifikasi adalah bahwa hipotesis dalam teorema Bayes merupakan label kelas yang menjadi target mapping dalam melakukan klasifikasi, dan bukti klasifikasi merupakan fitur yang menjadi masukan dalam model klasifikasi. Selain dalam klasifikasi, metode ini dapat juga digunakan dalam mengekstraksi kata kunci pada dokumen, yang mampu memberikan hasil yang baik berupa fitur yang digunakan pada training set, di antaranya TF-IDF, dan posisi kata kunci pada dokumen terkait. Metode ini telah digunakan di berbagai penelitian dan sangat populer di bidangnya.

Di beberapa penelitian, dilakukan perbandingan antara metode-metode klasifikasi lainnya seperti pada penelitian yang dilakukan Syahmia Gusriani, yang membandingkan metode K-NN, Naïve Bayes, dan Decision Tree, untuk kasus Analisis Sentimen dan menghasilkan sebuah kesimpulan yaitu Naïve Bayes memiliki tingkat akurasi yang paling tinggi diantara dua metode lainnya. Tiga metode ini digunakan Syahmia karena metode-metode ini merupakan metode yang paling populer saat ini. Selain itu, pada penelitian yang dilakukan Xhemali, Daniela, Chris J. Hinde, dan Roger G. Stone yaitu meneliti perbandingan tingkat akurasi Naïve Bayes, Decision Tree, Neural Networks dalam klasifikasi data training web pages, dan menghasilkan kesimpulan bahwa Naïve Bayes Classifier merupakan metode yang mampu memberikan tingkat akurasi yang lebih baik dibanding metode classifier lainnya.

Persamaan teorema Bayes:

$$P(A|B) = \frac{P(B|A).P(A)}{P(B)}$$

Keterangan:

B : Data class yang belum diketahui

A : Hipotesis data class spesifik

P(A|B) : Probabilitas hipotesis A berdasar kondisi B (posterior probability)

P(A) : Probabilitas hipótesis A (prior probability)

P(B|A) : Probabilitas B berdasarkan kondisi pada hipotesis A

P(B) : Probabilitas B

Berikut merupakan model dari Teorema Naïve Bayes:

$$P(c|X1, \ldots, Xn) = P(C) \prod_{i=1}^{n} P(Xi|C)$$

$$P(X) = P(c)P(c) \ldots P(c)P\,(c)$$

Persamaan tersebut merupakan model dari Teorema Naïve Bayes, yang akan digunakan dalam proses klasifikasi dengan data kontinu dengan rumus Densitas Gauss:

$$P = (Xi = xi \mid Yi = yi) = \frac{1}{\sqrt{2\pi\sigma ij}} e^{-\frac{(xi-uij)^2}{2\sigma^2 ij}}$$

Keterangan:

P : Peluang

Xi : Aribut ke i

xi : Nilai atribut ke i

Y : Kelas yang di cari

yj : Sub kelas Y yang dicari

u : Mean (rata-rata dari seluruh atribut)

o : Deviasi standar (varian dari seluruh atribut)

Rumus untuk menghitung Mean:

$$\mu = \frac{1}{n} \sum_{i=1}^{n} xi$$

Rumus untuk menghitung Deviasi Standar

$$\sigma = \left[\frac{1}{n-1} \sum_{i=1}^{n} (xi - \mu)^2 \right]^{0.5}$$

BAB 3
ANALISIS DAN DESAIN

Pada bab ini dijelaskan mengenai analisis dan desain penelitian yang menjadi acuan pada tahapan implementasi selama pelaksanaan Proyek Akhir PBA.

3.1 Analisis

Bagian ini akan menjelaskan proses pengolahan data sampai dengan proses pengklasifikasian emosi. Proses ini dilakukan secara manual menggunakan metode yang sebelumnya sudah ditentukan. Proses pengerjaan terdiri dari analisis data, pengolahan data (*cleaning data, stopword removal, case folding, tokenizing* dan *stemming*), metode *feature selection* (TF-IDF) dan terakhir proses klasifikasi menggunakan algoritma *Naïve Bayes*.

3.1.1 Analisis Data

Proyek mata kuliah PBA ini menggunakan dataset yang bersumber dari tautan berikut: https://huggingface.co/datasets/go_emotions. Dataset GoEmotions ini berisikan kumpulan data beranotasi manual terbesar yaitu terdiri dari 58.000 Reddit English comments, yang dikategorikan kedalam 27 label kategori emosi dan sebuah label netral. Data mentah disertakan serta versi dataset yang lebih kecil dan disederhanakan dengan pemisahan train/val/test yang telah ditentukan sebelumnya. Dataset GoEmotions ditujukan untuk keperluan pendalaman multi-class, multi-label emotion classification.

Dalam data mentah, label terdaftar sebagai kolomnya sendiri dengan entri biner 0/1 daripada daftar id seperti pada data yang disederhanakan.Selanjutnya pada proyek ini data akan dibagi menjadi data test dan data train.

3.1.2 Preprocessing Data

Data preprocessing merupakan proses pengolahan data, sehingga data yang hendak di klasifikasi merupakan data yang baik. Pada penelitian ini data yang digunakan untuk preprocessing adalah data dari reddit, untuk data tahapan preprocessing yang dilakukan adalah *cleaning data, stopword removal, case folding, tokenizing* dan *stemming*. Berikut ini merupakan tahapan yang dilakukan pada tahap preprocessing data.

1. Cleaning Data

   Cleaning adalah proses yang diperlukan karena dapat menghasilkan data yang bersih dan mampu mengurangi noise secara efektif. Karakter-karakter yang dihapus pada tahapan cleaning merupakan karakter karakter yang tidak penting, sehingga harus dibuang karena dapat mengganggu kinerja sistem dalam melakukan ekstraksi kata kunci.

   Teknik Cleaning Data yang digunakan dalam penelitian ini adalah sebagai berikut.

   a. Punctuation Removal

      Punctuation Removal proses dimana sistem akan menghilangkan menghilangkan tanda baca yang ada dalam kumpulan data. Tanda baca adalah karakter yang tidak memiliki pengaruh pada pengklasifikasian teks. Tanda baca yang umumnya akan dihapus dari kumpulan kalimat: (!), (.), (,), ($), (*), (%), (@), dan lain sebagainya.

      Contoh penerapan Punctuation Removal dapat dilihat pada Tabel 3.1 berikut ini.

Tabel 3.1. Punctuation Removal

Before Punctuation Removal	After Punctuation Removal
>sexuality shouldn’t be a grouping category	sexuality shouldn’t be a grouping category
You do right, if you don’t care then fuck ‘em!	You do right if you dont care then fuck em
Man i love reddit.	Man i love reddit
[NAME] was nowhere near them, he was by the Fact.	NAME was nowhere near them he was by the Fact

b. Case Folding

Case Folding merupakan proses yang dilakukan untuk menyamaratakan seluruh karakter pada teks menjadi huruf kecil. Pada tahap *case folding* semua huruf kapital, atau huruf besar akan diubah menjadi huruf kecil. Proses ini bertujuan untuk menyeragamkan data agar mudah di proses.

Contoh penerapan Case Folding dapat dilihat pada Tabel 3.2 berikut ini.

Tabel 3.2. Case Folding

Before Case Folding	After Case Folding
sexuality shouldn’t be a grouping category	sexuality shouldn’t be a grouping category
You do right if you dont care then fuck em	you do right if you dont care then fuck em
Man i love reddit	man i love reddit
NAME was nowhere near them he was by the Fact	name was nowhere near them he was by the fact

2. Tokenization

Tokenization merupakan proses untuk membagi teks yang dapat berupa kalimat, paragraf, atau dokumen, menjadi token-token atau bagian-bagian tertentu. Kemudian paragraph yang dipisahkan berdasarkan kalimat penyusunnya, dengan melihat beberapa delimiter yaitu tanda titik (.), tanda koma (,), dan tanda tanya (?). Proses ini dilakukan untuk menjadikan sebuah kalimat tersebut menjadi lebih bermakna. Pada penelitian ini, proses *tokenization* dilakukan untuk memenggal kalimat menjadi token-token berdasarkan *space*.

Contoh penerapan Case Folding dapat dilihat pada Tabel 3.3 berikut ini.

Tabel 3.3. Tokenization

Before Tokenization	After Tokenization
sexuality shouldn't be a grouping category	['sexuality', 'shouldn't', 'be', 'a', 'grouping', 'category']
you do right if you dont care then fuck em	['you', 'do', 'right', 'if', 'you', 'dont', 'care', 'then', 'fuck', 'em']
man i love reddit	['man', 'i', 'love', 'reddit']
name was nowhere near them he was by the fact	['name', 'was', 'nowhere', 'near', 'them', 'he', 'was', 'by', 'the', 'fact']

3. Stopword Removal

Pada tahap ini, semua kata-kata yang dianggap kurang memiliki makna yang penting dalam sebuah kalimat akan dihilangkan. Stop word merupakan kata-kata yang tidak penting yang tidak memiliki potensi sebagai kata kunci, seperti ialah, adalah, yaitu, ini, itu, dan sebagainya. Pustaka NLTK terdiri dari daftar kata-kata yang dianggap sebagai

stopwords untuk bahasa Inggris. Beberapa di antaranya adalah: [i, me, my, myself, we, our, ours, ourselves, you, you're, you've, you'll, you'd, your, yours, yourself, yourselves, he, most, other, some, such, no, nor, not, only, own, same, so, then, too, very, s, t, can, will, just, don, don't, should, should've, now, d, ll, m, o, re, ve, y, ain, aren't, could, couldn't, didn't, didn't].

Contoh penerapan Stopword Removal dapat dilihat pada Tabel 3.4 berikut ini.

Tabel 3.4. Stopword Removal

Before Stop Removal	After Stop Removal
[‘sexuality’, ‘shouldn’t’, ‘be’, ‘a’, ‘grouping’, ‘category’]	[‘sexuality’, ‘’, ‘grouping’, ‘category’]
[‘you‘, ‘do‘, ‘right‘, ‘if‘, ‘you‘, ‘dont‘, ‘care‘, ‘then‘, ‘fuck‘, ‘em‘]	[‘right‘, ‘dont‘, ‘care‘, ‘fuck‘, ‘em‘]
[‘man‘, ‘i‘, ‘love‘, ‘reddit‘]	[‘man‘, ‘love‘, ‘reddit‘]
[‘name‘, ‘was‘, ‘nowhere‘, ‘near‘, ‘them‘, ‘he‘, ‘was‘, ‘by‘, ‘the‘, ‘fact‘]	[‘name‘, ‘nowhere‘, ‘near‘, ‘fact‘]

4. Stemming

Proses stemming merupakan tahapan yang mengembalikan kata-kata ke bentuk dasarnya dengan cara menghapus semua awalan (prefix), imbuhan (affix) dan akhiran (suffix). Tahapan stemming digunakan agar memperkecil banyaknya indeks yang beda dari suatu dokumen dan mengelompokkan kata yang mempunyai arti yang sama untuk bentuk kata yang beda karena adanya imbuhan.

Contoh penerapan Stemming dapat dilihat pada Tabel 3.5 berikut ini.

Tabel 3.5. Stemming

Before Stemming	After Stemming
[‘sexuality’, ‘’, ‘grouping’, ‘category’]	[‘sexual’, ‘’, ‘group’, ‘categori’]
[‘right‘, ‘dont‘, ‘care‘, ‘fuck‘, ‘em‘]	[‘right‘, ‘dont‘, ‘care‘, ‘fuck‘, ‘em‘]
[‘man‘, ‘love‘, ‘reddit‘]	[‘man‘, ‘love‘, ‘reddit‘]
[‘name‘, ‘nowhere‘, ‘near‘, ‘fact‘]	[‘name‘, ‘nowher‘, ‘near‘, ‘fact‘]

5. Lemmatization

Lemmatization adalah proses yang bertujuan untuk melakukan normalisasi pada teks/kata dengan berdasarkan pada bentuk dasar yang merupakan bentuk lemma-nya. Normalisasi disini adalah dalam artian mengidentifikasikan dan menghapus prefiks serta sufiks dari sebuah kata. Lemma adalah bentuk dasar dari sebuah kata yang memiliki arti tertentu berdasar pada kamus.

Contoh penerapan lemmatization dapat dilihat pada Tabel 3.6 berikut ini.

Tabel 3.6. Lemmatization

Before Stemming	After Stemming
[‘sexual’, ‘’, ‘group’, ‘categori’]	[‘sexual’, ‘’, ‘group’, ‘categori’]
[‘right‘, ‘dont‘, ‘care‘, ‘fuck‘, ‘em‘]	[‘right‘, ‘dont‘, ‘care‘, ‘fuck‘, ‘em‘]
[‘man‘, ‘love‘, ‘reddit‘]	[‘man‘, ‘love‘, ‘reddit‘]
[‘name‘, ‘nowher‘, ‘near‘, ‘fact‘]	[‘name‘, ‘nowher‘, ‘near‘, ‘fact‘]

3.1.3 TF-IDF

Metode TF-IDF merupakan metode untuk menghitung bobot setiap kata yang paling umum digunakan pada information retrieval (Ridwan, et al., 2019). Metode TF-IDF juga menggabungkan dua konsep penghitungan bobot, yaitu frekuensi kata dalam dokumen tertentu dan inverse frekuensi dokumen yang mengandung kata tersebut. TF-IDF ini juga adalah sebuah ukuran statistik yang digunakan untuk mengevaluasi seberapa penting sebuah kata didalam sebuah dokumen atau dalam sekelompok kata. Frekuensi kata dalam dokumen tertentu menunjukkan pentingnya kata tersebut dalam dokumen. Berapa kali sebuah dokumen berisi kata ini menunjukkan seberapa umum kata itu. Oleh karena itu, jika frekuensi kata lebih tinggi dalam dokumen, dan frekuensi seluruh dokumen yang berisi kata lebih rendah dalam dokumen, bobot hubungan antara kata dan dokumen akan tinggi.

3.1.4 Analisis Algoritma Naive Bayes

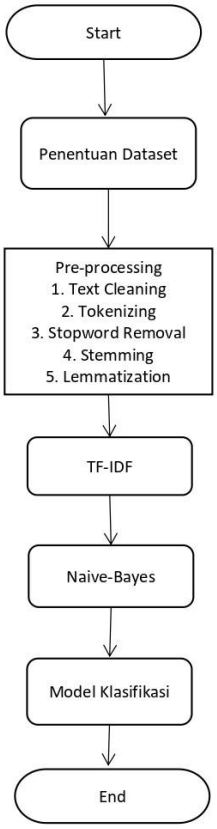
Pada metode pengklasifikasian mengenai *emotion* yang peneliti lakukan yaitu menggunakan metode *Naïve Bayes Classifier*. Pengklasifikasian ini dilakukan setelah melalui proses dan tahapan yang telah dilakukan sebelumnya.

3.2 Desain

Adapun desain yang akan dibuat adalah desain implementasi klasifikasi emosi yang akan dilakukan menggunakan algoritma Naïve Bayes. Desain yang dijelaskan meliputi proses klasifikasi menggunakan metode Naïve Bayes dan menggunakan metode ekstraksi TF-IDF.

3.2.1 Desain Pengerjaan Proyek

Desain percobaan pada bagian ini merupakan pengklasifikasian dengan metode Naïve Bayes. Klasifikasi dilakukan dengan menggunakan dataset yang telah diberikan yaitu dataset go-emotions. Berikut adalah flowchart selama pengerjaan proyek ini.



Gambar 1. Flowchart Pengerjaan proyek

Berikut penjelasan *flowchart* dari desain percobaan di atas:

1. Dataset

   Untuk dataset yang digunakan pada percoobaan ini menggunakan dataset yang sudah disediakan yaitu “go emotions”.

2. Pre-Processing Data

   Setelah memiliki dataset maka masuk ke tahap preprocessing yang dimana bertujuan untuk menghapus tanda baca, menyamakan huruf dan juga membuat kata menjadi kata dasar. Adapun tahapan preprocessing yaitu *cleaning data, stopword removal, case folding, tokenizing* dan *stemming.*

3. TF-IDF

   Pada penerapan TF-IDF model yang dibangun menggunakan library yang tersedia dan memberikan bobot pada setiap kata pada data.

4. Klasifikasi menggunakan Naïve Bayes

   Selanjutnya melakukan tahap klasifikasi menggunakan model Naïve Bayes. Untuk label emosi terdiri dari 28 label yaitu *admiration, amusement, anger, annoyance, approval, caring, confusion, curiosity, desire, disappointment, disapproval, disgust, embarrassment, excitement, fear, gratitude, grief, joy, love, nervousness, optimsm, pride, realization, relief, remorse, sadness, surprise, neutral.*

5. Hasil klasifikasi dari data uji.

   Setelah dilakukan seluruh tahapan, maka akan diperoleh hasil klasifikasi berdasarkan data uji.

BAB 4
IMPLEMENTASI DAN PEMBAHASAN

Pada bagian ini dijelaskan mengenai implementasi metode dan model yang dilakukan. Implementasi dilakukan sesuai dengan hasil analisis pada bab 3.

4.1 Implementasi Pre-processing

Pre-processing pada pemrosesan bahasa alami digunakan untuk mempersiapkan dokumen berupa text yang tidak terstruktur menjadi data yang telah siap untuk diolah. Implementasi *preprocessing* meliputi fungsi *cleaning, case folding, stopword removal,* dan *tokenization*. Data *preprocessing* merupakan teknik yang digunakan untuk mengkonversi data menjadi dataset yang siap untuk diolah. Bentuk *code* dari proses *preprocessing* adalah seperti pada *Script* berikut,

4.1.1 Data Cleaning

Implementasi yang dilakukan dalam Cleaning Data yang terdiri dari Punctuational Removal dan Case Folding.

4.1.1.1 Punctuation Removal

Tahapan yang dilakukan dalam proses Punctuation Removal adalah dengan menggunakan *library string* yang ada dalam bahasa pemrograman python. Pustaka *string* ini berisi beberapa daftar tanda baca yang telah ditentukan sebelumnya seperti '!”#$%&'()*+,-./:;?@[\]^_`{|}~'.

Setelah pustaka berhasil diimpor, dibuat sebuah fungsi yang menerima parameter berupa *string* (kalimat). Fungsi ini kemudian menghapus tanda yang ada dalam kalimat tersebut dengan cara memeriksa setiap karakter pada kalimat tersebut. Jika ditemukan ada karakter yang sesuai dengan tanda baca yang telah ditentukan, maka karakter tersebut akan dihapus.

Hasil dari pemanggilan fungsi ini adalah sebuah *string* (kalimat) yang telah bersih dari tanda baca. Kode program yang digunakan untuk menjalankan proses ini terlihat pada *source code* berikut ini.

```
#Cleaning Data
import string
string.punctuation

#Punctuation Removal

def remove_punctuation(text):
    punctuationfree="".join([i for i in text if i not in
string.punctuation])
    return punctuationfree

#storing the punctuation free text
feats_df['text']= feats_df['text'].apply(lambda
x:remove_punctuation(x))
feats_df.head()
```

4.1.1.2 Case Folding

Tahapan yang dilakukan dalam *Case Folding* adalah mengubah semua huruf dalam kalimat menjadi huruf kecil. Fungsi yang dilakukan dalam proses ini adalah menggunakan fungsi *lower()* dari python. Fungsi *lower()* merupakan fungsi yang mengubah setiap huruf menjadi huruf kecil. Kode program yang digunakan untuk menjalankan proses ini terlihat pada *source code* berikut ini.

```
# Mengubah ke huruf kecil
feats_df['text']= feats_df['text'].apply(lambda x: x.lower())
feats_df.head()
```

4.1.2 Tokenization

Pada tahap ini kalimat dipisahkan menjadi kata-kata penyusunnya atau disebut juga menjadi token-tokennya. Tahapan ini dilakukan dengan menggunakan fungsi *word_tokenize()* yang sudah tersedia di library *nltk* python. Fungsi ini menerima parameter berupa *string* (kalimat) dan *output* nya adalah kumpulan token-token dalam bentuk *array*. Kode program yang digunakan untuk menjalankan proses ini terlihat pada *source code* berikut ini.

```
import nltk
nltk.download('punkt')

def tokenization(text):
    tokens = nltk.word_tokenize(text)
    return tokens

#tokenization pada dataset
feats_df['text'] = feats_df['text'].apply(lambda x:
tokenization(x))
feats_df.head()
```

4.1.3 Stopword Removal

Stopwords removal melakukan penyimpanan kata-kata penting dalam sebuah dokumen dan menghilangkan kata-kata yang tidak penting untuk mengurangi indeks dan waktu pemrosesan. Proses ini dilakukan dengan memanfaatkan *library nltk* dan corpus *stopwords* yang telah tersedia di dalamnya. Corpus ini berisi kata-kata yang penting atau memiliki pengaruh dalam pemodelan.

Dalam tahap ini dibuat sebuah fungsi yang menerima parameter input berupa *array* yang terdiri dari elemen-elemen berupa kata. Parameter tersebut kemudian akan dilakukan perulangan sebanyak elemen yang ada di dalamnya. Perulangan bertujuan untuk memeriksa apakah kata dalam array tersebut terdapat dalam corpus yang ada. Jika sebuah kata tidak ada dalam corpus

maka kata tersebut akan dibuang. Kode program yang digunakan untuk menjalankan proses ini terlihat pada *source code* berikut ini.

```
#importing nlp library
import nltk

#Stop words present in the library
nltk.download('stopwords')
stopwords = nltk.corpus.stopwords.words('english')
stopwords[0:10]
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves',
'you', "you're"]

#defining the function to remove stopwords from tokenized text
def remove_stopwords(text):
    output= [i for i in text if i not in stopwords]
    return output

#applying the function
feats_df['text'] = feats_df['text'].apply(lambda
x:remove_stopwords(x))
feats_df.head()
```

4.1.4 Stemming

Pada tahapan ini suatu kata diubah menjadi bentuk kata dasarnya. Tahapan ini dilakukan dengan memanfaatkan fungsi *PorterStemmer* dari *nltk.stem.porter* yang ada di python. Sebelumnya akan dibuat sebuah fungsi yang akan menerima parameter input berupa *array* yang terdiri dari elemen-elemen berupa kata. Parameter tersebut kemudian akan dilakukan perulangan sebanyak elemen yang ada di dalamnya. Perulangan bertujuan untuk mengubah beberapa kata yang memiliki makna sama menjadi satu kata yang sama. Kode program yang digunakan untuk menjalankan proses ini terlihat pada *source code* berikut ini.

```
#importing the Stemming function from nltk library
from nltk.stem.porter import PorterStemmer
#defining the object for stemming
porter_stemmer = PorterStemmer()

#defining a function for stemming
```

```
def stemming(text):
  stem_text = [porter_stemmer.stem(word) for word in text]
  return stem_text

feats_df['text'] = feats_df['text'].apply(lambda x:
stemming(x))
```

4.1.5 Lemmatization

Setelah proses *stemming* akan terbentuk sebuah kata yang tidak ada artinya atau tidak sesuai dengan kata dasar. Maka itu proses *lemmatization* dilakukan untuk mengubah kata-kata yang tidak memiliki arti menjadi memiliki arti (misalnya *crazi* menjadi *crazy* (tidak waras)). Tahapan ini dilakukan dengan memanfaatkan fungsi *WordNetLemmatizer* dari *library nltk.stem* yang terdapat dalam python. Sebuah fungsi dibentuk untuk menerima parameter input berupa *array* yang terdiri dari elemen-elemen berupa kata. Parameter tersebut kemudian akan dilakukan perulangan sebanyak elemen yang ada di dalamnya. Perulangan bertujuan untuk mengubah beberapa kata yang tidak memiliki makna menjadi kata yang memiliki makna (kata dasar). Kode program yang digunakan untuk menjalankan proses ini terlihat pada *source code* berikut ini.

```
##Lemmatization
from nltk.stem import WordNetLemmatizer
nltk.download('wordnet')
#defining the object for Lemmatization
wordnet_lemmatizer = WordNetLemmatizer()

#defining the function for lemmatization
def lemmatizer(text):
  lemm_text = [wordnet_lemmatizer.lemmatize(word) for word in
text]
  return lemm_text

feats_df['text'] = feats_df['text'].apply(lambda
x:lemmatizer(x))
feats_df.head()
```

4.2 Implementasi TF-IDF

Tahap implementasi TF-IDF dilaksanakan dengan memanfaatkan fungsi *TfidfVectorizer* yang diambil dari *library python* yaitu *sklearn.feature_extraction.text*. Tahap ini dimulai dengan membagi dataset menjadi 2 yaitu *X* dan *y*, dimana *X* adalah kumpulan kalimat (komentar Reddit) dan *y* adalah ke-28 label yang digunakan. Label yang digunakan kemudian diubah dalam bentuk array numerik agar dapat diolah oleh model klasifikasi. Selanjutnya fungsi *TfidfVectorizer* dipanggil dengan *max_features* adalah 3000. Artinya *features* yang akan diekstraksi adalah maksimal sebanyak 3000 *features*. Kode program yang digunakan untuk menjalankan proses ini terlihat pada *source code* berikut ini.

```
from sklearn.feature_extraction.text import TfidfVectorizer
import numpy as np

X = feats_df["text"]
y = np.asarray(feats_df[feats_df.columns[1:]])

vetorizar = TfidfVectorizer(preprocessor=lambda x: x,
tokenizer=lambda x: x, max_features=3000, max_df=0.85)
vetorizar.fit(X)
```

4.3 Implementasi Hold Out

Splitting dataset dilakukan dengan menggunakan teknik Hold Out dengan rasio perbandingan *train* dan *test set* adalah sebesar 67:33. Tahapan ini dilakukan dengan memanfaatkan fungsi *train_test_split* yang terdapat dalam *library sklearn.model_selection*. *Train* dan *test set* diambil dari hasil TF-IDF yang telah dilakukan sebelumnya (dari variabel *X* dan *y*). Kode program yang digunakan untuk menjalankan proses ini terlihat pada *source code* berikut ini.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.33, random_state=42)
```

4.4 Implementasi Naive-bayes

Pada tahap ini dilakukan analisis pada pengimplementasian algoritma Naive Bayes yang akan menghasilkan sebuah model. Kemudian model yang dihasilkan akan dijadikan sebagai *knowledge* untuk mengklasifikasikan input data baru. Tahapan implementasi Naive Bayes ini memanfaatkan fungsi *BinaryRelevance* yang didapatkan dari *Library skmultilearn.problem_transform* dan fungsi *GaussianNB* yang didapatkan dari *Library sklearn.naive_bayes*. Proses pemodelan dilakukan dengan menggunakan *X_train* dan *y_train* yang sudah berupa vektorisasi hasil TF-IDF. Kode program yang digunakan untuk menjalankan proses ini terlihat pada *source code* berikut ini.

```
from skmultilearn.problem_transform import BinaryRelevance
from sklearn.naive_bayes import GaussianNB

classifier = BinaryRelevance(GaussianNB())

classifier.fit(X_train_tfidf, y_train)
```

BAB 5
HASIL DAN PEMBAHASAN

Pada bab ini akan dijelaskan hasil yang didapatkan selama melaksanakan penelitian ini.

5.1 Hasil Implementasi Naive Bayes

Implementasi yang dilakukan dengan Klasifikasi Naive Bayes dievaluasi dengan menggunakan *classification_report* yang berisi informasi berupa *precision*, *recall*, *f1_score,* dan *accuracy*. Hasil *classification_report* dapat dilihat pada Gambar 5.1 berikut ini.

	precision	recall	f1-score	support
0	0.10	0.90	0.18	5739
1	0.05	0.92	0.10	3067
2	0.04	0.89	0.08	2664
3	0.07	0.84	0.13	4457
4	0.09	0.79	0.16	5723
5	0.03	0.88	0.06	1892
6	0.04	0.88	0.07	2442
7	0.05	0.90	0.09	3148
8	0.02	0.82	0.04	1256
9	0.04	0.88	0.08	2790
10	0.06	0.86	0.11	3799
11	0.03	0.86	0.05	1759
12	0.01	0.68	0.03	814
13	0.03	0.85	0.06	1860
14	0.02	0.79	0.04	1031
15	0.07	0.94	0.13	3834
16	0.00	0.46	0.01	203
17	0.04	0.90	0.08	2589
18	0.05	0.93	0.09	2703
19	0.01	0.62	0.02	595
20	0.05	0.90	0.09	2905
21	0.01	0.50	0.02	446
22	0.04	0.86	0.08	2851
23	0.01	0.62	0.02	431
24	0.02	0.77	0.03	828
25	0.04	0.87	0.07	2176
26	0.03	0.88	0.06	1788
27	0.34	0.54	0.42	18410
micro avg	0.05	0.79	0.09	82200
macro avg	0.05	0.80	0.09	82200
weighted avg	0.12	0.79	0.17	82200
samples avg	0.05	0.76	0.10	82200

```
Accuracy:  2.869234631662004e-05
```

Gambar 2 Classification Report

Berdasarkan *classification_report* yang terlihat pada Gambar 5.1, didapatkan nilai *precision*, *recall*, dan *f1_score* dari masing-masing label yang ada. Sedangkan akurasi menunjukkan nilai keberhasilan model dalam memprediksi atau mengklasifikasikan data.

Precision atau Presisi adalah representasi tingkat ketepatan antara data yang diinginkan dengan hasil prediksi model. Precision membandingkan prediksi True Positive dengan semua hasil prediksi yang positif. Berdasarkan hasil *precision* yang didapatkan, nilai tertinggi berada pada label ke-27 (Label Neutral) yaitu sebanyak 0,34 poin. Hal ini mengindikasikan bahwa model mampu memprediksi label *Neutral* dengan benar lebih banyak dibandingkan label-label lainnya. Pada posisi kedua terdapat label ke-0 (Label Admiration) yaitu sebanyak 0,10 poin dan selebihnya hanya mendapatkan nilai *precision* sebanyak < 0,1 poin.

Recall adalah rasio prediksi benar positif dibandingkan dengan keseluruhan data yang benar positif. Berdasarkan hasil *classification_report*, nilai *recall* tertinggi diperoleh pada label ke-15 (label Gratitude) yaitu sebanyak 0,94 poin. Dari nilai tersebut dapat diperoleh informasi bahwa model memprediksi data dengan label Gratitude lebih banyak dibandingkan label lainnya. Hal ini mengindikasikan bahwa data dengan label Graditude lebih banyak diprediksi dengan benar dibandingkan label-label lainnya. Sedangkan label dengan hasil *recall* yang paling kecil terdapat pada label ke-16 (Label Grief). Hal ini mengindikasikan bahwa data dengan label Grief lebih sedikit diprediksi dengan benar oleh model klasifikasi.

F1-Score adalah perbandingan rata-rata presisi dan *recall* yang dibobotkan. Berdasarkan hasil dari *classification_report*, rata-rata dari nilai *f1_score* adalah sebanyak 0,09. Hal ini mengindikasikan bahwa keseimbangan antara *precision*

dan *recall* cukup rendah karena model belum mampu memprediksi dengan tepat berbagai label yang ada dalam masing-masing teks atau kalimat.

Akurasi atau *accuracy* adalah representasi ketepatan model berhasil melakukan klasifikasi. Nilai akurasi yang didapatkan dengan menggunakan algoritma Naive Bayes adalah sebesar 0,0000286 atau 0,00268% yang dimana cukup rendah dalam memprediksi data. Hasil akurasi yang rendah kemungkinan didapatkan dalam kasus *multi-label classification* dimana model masih belum mampu memprediksi dengan akurat setiap label yang seharusnya ada dalam sebuah data. Yang membuat nilai akurasi ini kecil adalah ketika proses *training* menganggap suatu data diprediksi dengan salah bahkan jika terdapat 1 label yang tidak sesuai.

Tabel 5.1. Contoh Hasil Prediksi

Teks	Label Sesungguhnya	Label Hasil Prediksi
That game hurt	sadness	annoyance, disapproval, disgust, sadness, surprise

Pada Tabel 5.1 diatas ketika suatu data (misalnya “That Game Hurt”) yang sebenarnya berlabel *sadness*, namun ketika diprediksi menjadi *annoyance, disapproval, disgust, sadness,* dan *surprise* akan dihitung sebagai prediksi yang salah oleh evaluasi metrik. Sehingga untuk mendapatkan nilai akurasi yang tinggi, sebuah model harus mampu memprediksi dengan benar dan tepat seluruh label yang sesuai pada data tertentu.

BAB 6
PENUTUP

Pada bab ini menjelaskan mengenai pembagian tugas pengerjaan proyek, kesimpulan pengerjaan proyek, serta saran yang diberikan oleh Penulis.

6.1 Pembagian Tugas

Pembagian tugas anggota kelompok dalam pengerjaan proyek Pemrosesan Bahasa Alami dituliskan pada tabel

No	Tahap	Penanggung jawab					
		Efrica Situmeang	**Aldi Siagian**	**Grace Noel**	**Anjel Pardede**	**Grace Widya**	**Romual Naibaho**
1	Pembuatan proposal	✔	✔	✔	✔	✔	✔
2	Preprocessing	✔	✔	✔	✔	✔	✔
3	TF-IDF	✔		✔			✔
4	Klasifikasi Naive Bayes		✔			✔	✔
5	Classification Report				✔		✔

6	Pembuatan laporan akhir	✔	✔	✔	✔	✔	✔

6.2 Kesimpulan

Berdasarkan hasil percobaan yang diperoleh, maka kesimpulan yang peneliti dapatkan adalah sebagai berikut:

1. Penelitian ini telah mengiymplementasikan algoritma Naive Bayes dalam mengklasifikasikan multi-label emosi pada komentar Reddit. Penelitian ini juga telah mengimplementasikan TF-IDF untuk ekstraksi fitur dari data teks.
2. Hasil akurasi yang didapatkan setelah melaksanakan penelitian ini adalah sebanyak 0.00268% dengan menggunakan TF-IDF dan Naive Bayes.

6.3 Saran

Setelah melalui tahapan pengklasifikasian *multi label emotion* menggunakan Naive Bayes maka ada beberapa saran yang diberikan untuk pengembangan hasil pengklasifikasian yaitu:

1. Hasil akurasi dari penelitian ini masih cukup rendah sehingga diharapkan untuk penelitian selanjutnya bisa membangun model yang lebih baik agar didapatkan hasil akurasi yang lebih tinggi.
2. Fitur ekstraksi yang digunakan pada penelitian ini adalah TF-IDF, diharapkan pengembangan penelitian ini bisa menggunakan fitur ekstraksi lainnya.
3. Penelitian lanjutan dari penelitian ini diharapkan bisa membandingkan hasil metode Naive Bayes dengan metode klasifikasi lainnya.

DAFTAR REFERENSI

Al-Saaqa, S., Abdel-Nabi, H., & Awajan, A. (2018). A Survey of Textual Emotion Detection. 2018 8th International Conference on Computer Science and Information Technology, CSIT 2018, 136–142. https://doi.org/10.1109/CSIT.2018.8486405

Xhemali, D., Hinde C.J., And Stone R.G., 2009." Naïve Bayes vs. Decision Trees vs. Neural Networks in the Classification of Training Web Pages", Journal of Computer Science Issues, Vol. 4, No. 1.

Yasin, U., 2005. Keyword Extraction Using Naive Bayes. pp. 1-5.

Maulana, M. 2016. Emotion Detection of Indonesian Tweets Using Naive Bayes Classification. repository its.

Melita, R., Amrizal, V., Suseno, H. B. & Dirjan, T. 2018. Penerapan Metode Term Frequency Inverse Document Frequency. *Jurnal Teknik,* Vol. 11 No. 2.

P. Ekman, Universals and cultural differences in facial expressions of emotion, vol. 19, Nebraska: University of Nebraska Press, 1972.