**PROPOSAL PROYEK**

**11S4037 – PEMROSESAN BAHASA ALAMI**

**Multi-label Emotion Classification using Bi-LSTM with GloVe Word Embedding**



**Disusun oleh:**

|  |  |
| --- | --- |
| **12S18004** | **Rosalia Pane** |
| **12S18008** | **Indah Tri Anastasya Manik** |
| **12S18011** | **Nadya Putri Tambunan** |
| **12S18043** | **Roy Gunawan napitupulu** |
| **12S18048** | **Rifka Uli Siregar** |

**FAKULTAS INFORMATIKA DAN TEKNIK ELEKTRO**

**INSTITUT TEKNOLOGI DEL**

**2021**

**DAFTAR ISI**

[**BAB I PENDAHULUAN** 5](#_Toc89107066)

[**1.1.** **Latar Belakang** 5](#_Toc89107067)

[**1.2.** **Tujuan** 7](#_Toc89107068)

[**1.3.** **Manfaat** 7](#_Toc89107069)

[**1.4.** **Ruang Lingkup** 7](#_Toc89107070)

[**BAB II ISI** 8](#_Toc89107071)

[**2.1 Analisis** 8](#_Toc89107072)

[**2.1.1 Analisis Data** 8](#_Toc89107073)

[**2.1.2 Analisis Metode** 11](#_Toc89107074)

[**2.2 Desain** 11](#_Toc89107075)

[**2.2.1 *Mapping Label*** 12](#_Toc89107076)

[**2.2.2 *Data Preprocessing*** 12](#_Toc89107077)

[**2.2.3 *Word Embedding Glove*** 15](#_Toc89107078)

[**2.2.4 *Modelling with Bi-LSTM*** 15](#_Toc89107079)

[**2.2.5 *Evaluation and Results*** 16](#_Toc89107080)

[**2.3 Implementasi** 16](#_Toc89107081)

[**2.3.1 *Mapping Label*** 16](#_Toc89107082)

[**2.3.2 *Data Preprocessing*** 16](#_Toc89107083)

[**2.3.3 *Word Embedding Glove*** 22](#_Toc89107084)

[**2.3.3 *Modeling with Bi-LSTM*** 23](#_Toc89107085)

[**2.4 Hasil** 24](#_Toc89107086)

[**2.4.1 Evaluation BI-LSTM Model – Glove** 24](#_Toc89107087)

[**2.4.2 *Accuracy Bi-LSTM Model with GloVe*** 25](#_Toc89107088)

[**BAB III PENUTUP** 26](#_Toc89107089)

[**3.1 Pembagian Tugas dan Tanggung Jawab** 26](#_Toc89107090)

[**3.2** **Kesimpulan** 28](#_Toc89107091)

[**3.2** **Saran** 28](#_Toc89107092)

[**DAFTAR PUSTAKA** 29](#_Toc89107093)

**DAFTAR TABEL**

[**Tabel 1. Atribut pada dataset** 8](#_Toc89098506)

[**Tabel 2. Pembagian tugas dan tanggung jawab** 26](#_Toc89098507)

**DAFTAR GAMBAR**

[**Gambar 1. Distribusi Label pada Dataset** 10](#_Toc89098508)

[**Gambar 2. Flowchart desain** 12](#_Toc89098509)

[**Gambar 3. Contoh Remove words containing numbers** 14](#_Toc89098510)

[**Gambar 4. Contoh clean contractions** 14](#_Toc89098511)

[**Gambar 5.Contoh clean special characters** 14](#_Toc89098512)

[**Gambar 6.Contoh Correct Spelling** 14](#_Toc89098513)

[**Gambar 7.Modeling with Bi-LSTM** 15](#_Toc89098514)

[**Gambar 8. Code program mengecek nilai null pada data** 17](#_Toc89098515)

[**Gambar 9.Output pengecekan nilai null** 17](#_Toc89098516)

[**Gambar 10. Kode Program Clean Text** 18](#_Toc89098517)

[**Gambar 11. Hasil Clean Text** 18](#_Toc89098518)

[**Gambar 12. Kode Program Clean Constractions** 19](#_Toc89098519)

[**Gambar 13. Hasil Clean Constractions** 19](#_Toc89098520)

[**Gambar 14. Kode Program Clean Special Characters** 20](#_Toc89098521)

[**Gambar 15. Hasil Clean Special Character** 20](#_Toc89098522)

[**Gambar 16. Kode Program Correct Spelling** 20](#_Toc89098523)

[**Gambar 17. Hasil Correct Spelling** 21](#_Toc89098524)

[**Gambar 18. Kode Program Remove Space** 21](#_Toc89098525)

[**Gambar 19. Hasil Remove Space** 22](#_Toc89098526)

[**Gambar 20. Kode Program Embedding GloVe** 23](#_Toc89098527)

[**Gambar 21. Output Embedding GloVe** 23](#_Toc89098528)

[**Gambar 22. Kode Program Modelling dengan Bi-LSTM** 23](#_Toc89098529)

[**Gambar 23. Output Modelling dengan Bi-LSTM** 24](#_Toc89098530)

[**Gambar 24. Kode Evaluasi Bi-LSTM - Glove** 24](#_Toc89098531)

[**Gambar 25. Output Evaluasi Bi-LSTM - Glove** 24](#_Toc89098532)

[**Gambar 26. Kode Program Akurasi Bi-LSTM – Glove** 25](#_Toc89098533)

[**Gambar 27. Output Akurasi Bi-LSTM - Glove** 25](#_Toc89098534)

# **BAB I PENDAHULUAN**

Bagian ini menyajikan latar belakang, tujuan, manfaat dan ruang lingkup pengerjaan proyek.

## **Latar Belakang**

Emosi adalah keadaan pikiran yang berlangsung terus-menerus, yang ditandai dengan gejala mental, fisik dan perilaku. Emosi seseorang dapat diidentifikasi secara langsung melalui ekspresi wajah dan ucapannya. Mendeteksi emosi secara otomatis sangat penting karena dapat diterapkan di berbagai bidang. Misalnya dalam dunia pendidikan, analisis emosi dapat dimanfaatkan untuk lingkungan *e-learning*. Selain itu, dalam bisnis yang digunakan untuk mengidentifikasi keluhan pelanggan [1].

Dari pengalaman sehari-hari, beberapa emosi tampaknya berbeda dan terjadi secara independen. Emosi yang secara inheren kontradiktif, seperti love dan hate mungkin memerlukan serangkaian kelas yang terpisah untuk mengakomodasikan aspek dari setiap kelas. Disisi lain, emosi yang identik biasanya berada dibawah valensi emosional yang sama dan sering muncul bersamaan dalam situasi tertentu. Oleh karena itu, berbagai emosi ini dapat dikelompokkan bersama. Deteksi emosi, yang berperan sebagai masalah klasifikasi *multi-label* dapat membantu menjelaskan sifat kompleks dari emosi yang terjadi bersamaan, sehingga memberikan pemahaman tentang karakteristik setiap emosi [2].

Pendeteksian emosi merupakan salah satu masalah yang muncul di bidang *Natural Language Processing* (NLP). NLP digunakan untuk memproses data seperti teks yang terstruktur maupun tidak menjadi pengetahuan bermakna untuk berbagai masalah bisnis. NLP telah banyak digunakan untuk solusi masalah seperti klasifikasi, pemodelan topik, *text generation*, QA *system*, rekomendasi, dan lainnya[3].

Baru-baru ini, masalah klasifikasi *multi-label* telah menarik minat yang cukup besar karena penerapannya ke berbagai *domain*, termasuk klasifikasi teks, klasifikasi adegan dan video, dan bioinformatika [4]. Berbeda dengan masalah klasifikasi tradisional label tunggal (yaitu, multi-kelas atau biner), di mana sebuah *instance* dikaitkan dengan hanya satu label dari satu set label yang terbatas, dalam masalah klasifikasi *multi-label*, sebuah *instance* dikaitkan dengan *subset* dari label [5].

Pendeteksian emosi menggunakan klasifikasi *multi-label* menjadi masalah karena suatu kalimat cenderung melibatkan lebih dari satu kategori emosi. Sehingga, tantangan utama yang muncul adalah bagaimana memodelkan ketergantungan antar label menggunakan pendekatan klasifikasi. Misalnya, emosi dengan label “*angry*” dan “*disgust*” memiliki ketergantungan daripada emosi “*sad*” dan “*joy*” yang saling bertentangan [6]. Analisis emosi melalui sebuah teks tampaknya juga menjadi tantangan karena faktanya bahwa ekspresi tekstual tidak selalu secara langsung melibatkan kata-kata yang berhubungan dengan emosi, tetapi seringkali suatu kalimat perlu dipahami untuk memberikan sebuah makna [7].

Salah satu pendekatan yang dapat digunakan untuk mengatasi masalah klasifikasi multi-label adalah metode *Bidirectional Long Short Term Memory* (Bi-LSTM). Metode Bi-LSTM merupakan perkembangan dari model LSTM dengan dua lapisan, dimana lapisan pertama bergerak maju (*forward*) untuk memahami dan memproses dari kata pertama menuju kata terakhir, demikian sebaliknya lapisan atasnya bergerak mundur (*backward*) untuk memahami dan memproses dari kata terakhir menuju kata pertama. Oleh karena itu, Bi-LSTM sangat baik digunakan untuk mengenali pola dalam kalimat, dikarenakan setiap kata dalam kalimat diproses secara sekuensial [8].

Kemudian dalam melakukan klasifikasi, metode Bi-LSTM akan digabungkan dengan salah satu pendekatan pembobotan kata yaitu *pre-trained word embeddings GloVe*. Dimana, *GloVe* merupakan salah satu pendekatan yang memiliki akurasi yang baik untuk memproses pembobotan kata dalam data dibanding model *word embeddings* lain seperti CBOW dan *skip-grams*. Secara keseluruhan, *GloVe* mengungguli model lain dalam hal analogi kata, kemiripan kata dan tugas *named entity recognition* [9]

Oleh karena itu, berdasarkan uraian dari permasalahan sebelumnya, penulis berfokus pada pengklasifikasian emosi *multi-label*, yang bertujuan untuk mengembangkan sistem otomatis untuk mengkategorikan kalimat ke dalam netral dan 27 emosi seperti *admiration*, *amusement*, *anger* dan emosi lainnya. Penggunaan pendekatan Bi-LSTM dan *word embedding GloVe* akan membantu dalam membentuk matriks *embedding* pada masalah klasifikasi *multi-label* dengan menggunakan *dataset GoEmotions* yang diperoleh dari *Hugging Face.*

## **Tujuan**

Tujuan dari proyek *multi-label emotion classification* ini, antara lain:

1. Menerapkan metode *Bidirectional Long Short Term Memory* (Bi-LSTM) dengan *GloVe* sebagai *word embedding* dalam menganalisis teks apakah teks tersebut termasuk ke dalam teks dengan beberapa label (*multi-label*).
2. Untuk mengetahui bagaimana tingkat akurasi menggunakan metode *Bidirectional Long Short Term Memory* (Bi-LSTM) dengan *GloVe* sebagai *word embedding* dalam melakukan pengklasifkasian *multi-label*.

## **Manfaat**

Berikut adalah manfaat dari pembuatan *multi-label emotion classification* menggunakan metode Bi-LSTM dan *GloVe* antara lain:

1. Mengetahui cara dan proses klasifikasi *multi-label emotion* menggunakan metode Bi-LSTM dan *GloVe* sebagai *word embedding*.
2. Mengetahui tingkat akurasi menggunakan metode Bi-LSTM dan *GloVe* sebagai *word embedding* dalam melakukan klasifikasi *multi-label emotion*.

## **Ruang Lingkup**

Ruang lingkup dalam pengerjaan proyek ini yaitu menggunakan metode Bi-LSTM dan *word embedding* *GloVe* dengan menggunakan *dataset GoEmotions* yang diperoleh dari *Hugging Face* [10]*.*

# **BAB II ISI**

Pada bab ini mencakup analisis yaitu analisis terhadap data dan analisis terhadap metode.

## **2.1 Analisis**

Pada *subbab* ini dijelaskan analisis yang dilakukan terhadap data dan metode yang digunakan dalam pengimplementasian *multi-label* klasifikasi emosi.

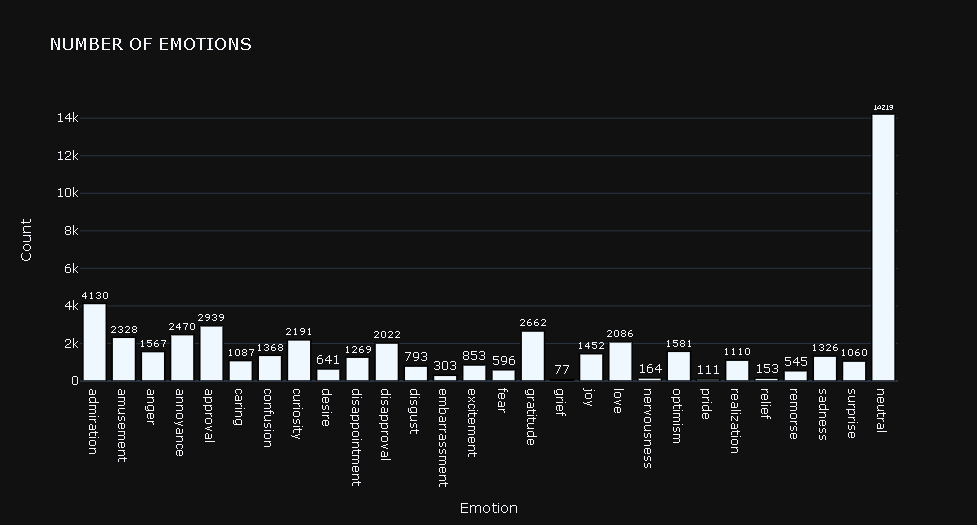
### **2.1.1 Analisis Data**

*Dataset* yang digunakan dalam proyek ini menggunakan *dataset GoEmotions* yang diperoleh dari *Hugging Face* [10]. *Dataset GoEmotions* terdiri dari 43410 baris dan 37 kolom. *Dataset* tersebut telah diberikan beberapa kelas/label yaitu pada setiap teks dalam data tersebut. Pada Tabel 1 berikut menampilkan gambaran dari dataset yang digunakan yang terdiri dari nama atribut, tipe atribut dan keterangan.

**Tabel 1. Atribut pada dataset**

|  |  |  |  |
| --- | --- | --- | --- |
| No | Nama Atribut | Tipe Atribut | Keterangan |
| 1 | *text* | Kategorikal | Komentar yang berasal dari Reddit |
| 2 | *id* | Kategorikal | Berisi id text |
| 3 | *author* | Kategorikal | Nama pengguna penulis komentar di Reddit |
| 4 | *subreddit* | Kategorikal | Subreddit tempat komentar berasal |
| 5 | *link\_id* | Kategorikal | ID tautan komentar |
| 6 | *parent\_id* | Kategorikal | ID *parent* dari komentar |
| 7 | *created\_utc* | Numerik | *Timestamp* komentar |
| 8 | *rater\_id* | Numerik | ID unik dari annotator |
| 9 | *example\_very\_unclear* | Kategorikal | Apakah annotator menandai label yang tidak jelas atau sulit untuk diberi label (dalam hal ini mereka tidak memilih label emosi) |
| 10 | *admiration* | Kategorikal | Berisi pengelompokan *text* antara *admiration* (1) dan *non-admiration* (0) |
| 11 | *amusement* | Kategorikal | Berisi pengelompokan *text* antara *amusement*(1) dan *non-amusement* (0) |
| 12 | *anger* | Kategorikal | Berisi pengelompokan *text* antara *anger*(1) dan *non-anger* (0) |
| 13 | *annoyance* | Kategorikal | Berisi pengelompokan *text* antara *annoyance*(1) dan *non-annoyance* (0) |
| 14 | *approval* | Kategorikal | Berisi pengelompokan *text* antara *approval* (1) dan *non-approval* (0) |
| 15 | *caring* | Kategorikal | Berisi pengelompokan *text* antara *caring* (1) dan *non-caring* (0) |
| 16 | *confusion* | Kategorikal | Berisi pengelompokan *text* antara *confusion*(1) dan *non-confusion* (0) |
| 17 | *curiosity* | Kategorikal | Berisi pengelompokan *text* antara *curiosity* (1) dan *non-curiosity* (0) |
| 18 | *desire* | Kategorikal | Berisi pengelompokan *text* antara *desire* (1) dan *non-desire* (0) |
| 19 | *disappointment* | Kategorikal | Berisi pengelompokan *text* antara *disappointment* (1) dan *non-disappointment* (0) |
| 20 | *disapproval* | Kategorikal | Berisi pengelompokan *text* antara *disapproval* (1) dan *non-disapproval* (0) |
| 21 | *disgust* | Kategorikal | Berisi pengelompokan *text* antara *disgust* (1) dan *non-disgust* (0) |
| 22 | *embarrassment* | Kategorikal | Berisi pengelompokan *text* antara *embarrassment* (1) dan *non-embarrassment* (0) |
| 23 | *excitement* | Kategorikal | Berisi pengelompokan *text* antara *excitement* (1) dan *non-excitement* (0) |
| 24 | *fear* | Kategorikal | Berisi pengelompokan *text* antara *fear* (1) dan *non-fear* (0) |
| 25 | *gratitude* | Kategorikal | Berisi pengelompokan *text* antara *gratitude* (1) dan *non-gratitude* (0) |
| 26 | *grief* | Kategorikal | Berisi pengelompokan *text* antara *grief* (1) dan *non-grief* (0) |
| 27 | *joy* | Kategorikal | Berisi pengelompokan *text* antara *joy* (1) dan *non-joy* (0) |
| 28 | *love* | Kategorikal | Berisi pengelompokan *text* antara *love* (1) dan *non-love* (0) |
| 29 | *nervousness* | Kategorikal | Berisi pengelompokan *text* antara *nervousness* (1) dan *non-nervousness* (0) |
| 30 | *optimism* | Kategorikal | Berisi pengelompokan *text* antara *optimism* (1) dan *non-optimism* (0) |
| 31 | *pride* | Kategorikal | Berisi pengelompokan *text* antara *pride* (1) dan *non-pride* (0) |
| 32 | *realization* | Kategorikal | Berisi pengelompokan *text* antara *realization* (1) dan *non-realization* (0) |
| 33 | *relief* | Kategorikal | Berisi pengelompokan *text* antara *relief* (1) dan *non-relief* (0) |
| 34 | *remorse* | Kategorikal | Berisi pengelompokan *text* antara *remorse* (1) dan *non-remorse* (0) |
| 35 | *sadness* | Kategorikal | Berisi pengelompokan *text* antara *sadness* (1) dan *non-sadness* (0) |
| 36 | *surprise* | Kategorikal | Berisi pengelompokan *text* antara *surprise* (1) dan *non-surprise* (0) |
| 37 | *neutral* | Kategorikal | Berisi pengelompokan *text* antara *neutral* (1) dan *non-neutral* (0) |

Berikut adalah distribusi label pada *dataset* ditunjukkan pada Gambar 1.



**Gambar 1. Distribusi Label pada Dataset**

Setiap label dari dataset *GoEmotions*, selanjutnya akan dilakukan pengklasifikasian dengan di *mapping* ke dalam 7 label. Berikut daftar label yang digunakan untuk klasifikasi.

* *anger: anger, annoyance, disapproval*
* *disgust: disgust*
* *fear: fear, nervousness*
* *joy: joy, amusement, approval, excitement, gratitude, love, optimism, relief, pride, admiration, desire, caring*
* *sadness: sadness, disappointment, embarrassment, grief, remorse*
* *surprise: surprise, realization, confusion, curiosity*
* *neutral: neutral*

### **2.1.2 Analisis Metode**

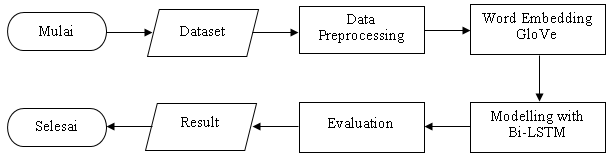
Pada klasifikasi kelas (seperti *binary* dan *multi-class*), *output* yang diterima akan tunggal dari beberapa opsi yang diberikan. *Binary class* akan memberikan kemungkinan kelas "N" sebanyak 2 (N = 2) sementara *multi-class* akan memberi kemungkinan sebanyak N > 2. Berbeda dengan *Multi-label text classification*, jenis pengklasifikasian ini memprediksi beberapa kemungkinan label yang akan dilibatkan dalam teks tertentu. Artinya, *output* yang dihasilkan akan lebih dari satu.

*Bidirectional Long Short-Term Memory* (Bi-LSTM) adalah perkembangan dari model LSTM dimana terdapat dua lapisan yang prosesnya saling berkebalikan arah, model ini sangat baik untuk mengenali pola dalam kalimat karena setiap kata dalam kalimat diproses secara sekuensial. Dengan adanya lapisan dua arah yang saling berlawanan ini maka model dapat memahami dan mengambil perspektif dari kata terdahulu dan kata terdepan, sehingga proses pembelajaran akan semakin dalam yang berdampak pada model akan lebih memahami konteks pada klasifikasi emosi tersebut. *GloVe* merupakan salah satu pendekatan yang memiliki akurasi yang baik untuk memproses pembobotan kata dalam data dibanding model *word embeddings* lain seperti CBOW dan *skip-grams*. Sehingga dalam melakukan klasifikasi, metode Bi-LSTM akan digabungkan dengan salah satu pendekatan pembobotan kata yaitu *pre-trained word embeddings GloVe*.

Analisis *multi-label emotion classification* pada komentar Reddit yang terdapat dalam *dataset GoEmotions* diklasifikasikan dengan metode Bi-LSTM yang akan digabungkan dengan salah satu pendekatan pembobotan kata yaitu *pre-trained word embeddings GloVe*.

## **2.2 Desain**

Pada subbab ini dijelaskan desain pemrosesan bahasa alami yaitu yang ditampilkan dalam bentuk *flowchart* atau diagram alir seperti ditunjukkan pada Gambar 2 berikut ini.



**Gambar 2. Flowchart desain**

### **2.2.1 *Mapping Label***

Pada tahap ini akan dilakukan *mapping* (pemetaan) untuk setiap label dari dataset *GoEmotions* ke dalam 7 label. Berikut daftar label yang digunakan untuk klasifikasi.

* *anger: anger, annoyance, disapproval*
* *disgust: disgust*
* *fear: fear, nervousness*
* *joy: joy, amusement, approval, excitement, gratitude, love, optimism, relief, pride, admiration, desire, caring*
* *sadness: sadness, disappointment, embarrassment, grief, remorse*
* *surprise: surprise, realization, confusion, curiosity*
* *neutral: neutral*

### **2.2.2 *Data Preprocessing***

*Preprocessing* adalah proses pengubahan bentuk data yang belum terstruktur menjadi data yang terstruktur. Tahap *preprocessing* mengubah data tekstual menjadi data yang siap dijadikan model *text mining* [12]. Ada beberapa tahapan yang biasa dilakukan pada tahap ini, yaitu *tokenization*, *stop-word removal*, *lowercase conversion*, dan *lemmatization* [12]. Oleh karena itu perlu dilakukan data *preprocessing* untuk menghilangkan kata-kata pada teks atau dokumen yang mengandung beberapa format yang keberadaannya tidak penting dalam *text mining.*

#### **2.2.1.1 Data Cleaning**

Data yang diperoleh dari dataset memiliki beberapa *noise* yang perlu dibersihkan, misalnya string yang kosong (*incomplete data/missing value*). *Data cleaning* dilakukan ketika data yang diperoleh tidak lengkap (*missing value*), terdapat *error* (*noisy data*) dan juga tidak konsisten. *Data cleaning* perlu dilakukan karena ketiga masalah di atas dapat mengakibatkan hasil prediksi dalam klasifikasi menjadi tidak akurat. Untuk mengatasi *noisy* data dilakukan beberapa cara yakni *binning, regression, clustering* dan *semi supervised method* [11].

##### **2.2.1.1.1 *Clean Text***

Pada tahap *clean text* dilakukan beberapa proses *data cleanig,* seperti:

* *Clean emoji*

Tahapan ini merupakan tahapan *preprocessing* untuk menghapus emoji karena tidak dapat dianalisis. Dilakukan dengan mengkodekan *string* menggunakan pengkodean ASCII yang selanjutnya akan didekodekan untuk menghapus emoji.

* *Make text lowercase*

*Lower Casing* atau *case folding* atau adalah salah satu tahap dari *preprocessing* untuk *text mining*, dimana semua huruf diubah menjadi huruf kecil untuk mencegah sensitivitas huruf besar-kecil. Dengan cara ini, kita dapat meningkatkan kinerja *classifier* tanpa mempertimbangkan ketidakkonsistenan teks.

* *Remove text in square brackets*

Karakter [ dan ] merupakan karakter khusus dalam *regex*. Karakter tersebut digunakan untuk menampilkan daftar karakter yang cocok. Proses ini akan menghapus teks dalam kurung {}, kotak [], dan/atau bulat (), serta kurung itu sendiri.

* *Remove links*

Penggunaan *remove links* akan membantu dalam menghapus *url* atau *link* yang terdapat dalam teks. Munculnya *remove links* membuat data tidak efektif dan tidak memiliki arti.

* *Remove punctuation*

Penggunaan *punctuation* (tanda baca) seperti spasi, tanda konvensional atau tipografi tertentu biasanya membantu pembaca untuk memahami teks tertulis. Tetapi dalam pemrosesan data, tanda baca tersebut perlu dihapus untuk menghilangkan bagian data yang tidak membantu, atau *noise*.

* *Remove words containing numbers*

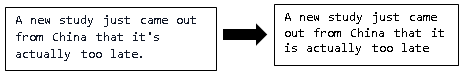
Selama proses pembersihan data biasanya diperlukan penghapusan angka dari data di *Natural Language Processing*. Misalkan data memiliki string abcd1234efg567, dan dilakukan penghapusan digit/angka dari string untuk mendapatkan string seperti abcdefg.



**Gambar 3. Contoh Remove words containing numbers**

##### **2.2.1.1.2 *Clean Contractions***

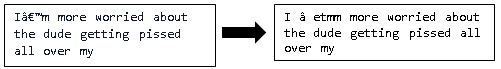
Kata-kata yang ditulis dengan apostrof (‘) disebut sebagai *contractions*. Tujuannya adalah untuk membakukan teks aar lebih masuk akal. Misalnya: *don’t* menjadi *do not, can’t* menjadi *cannot*.



**Gambar 4. Contoh clean contractions**

##### **2.2.1.1.3 *Clean Special Characters***

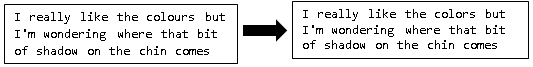
Proses ini bertujuan untuk menghapus spesial karakter yang terdiri dari *non alphanumeric* karakter seperti ! ++ << [ % , <=<= ] & — <><> | ‘ . = ~ ( / == ~= ) /! >>// >>= ! { ? ` \* } @ : ; ^ |= &= += -= = /= \*=

****

**Gambar 5.Contoh clean special characters**

##### **2.2.1.1.4 *Correct Spelling***

Kesalahan ejaan sering terjadi dan telah banyak fitur perangkat lunak yang menyediakan perbaikan dari kesalahan tersebut. *Python* menawarkan banyak modul yang bertujuan untuk membuat penulisan pemeriksa ejaan sederhana menjadi mudah. Dalam hal ini akan digunakan kamus sederhana yang sudah didefinisikan sebelumnya untuk melakukan perbaikan kesalahan ejaan kata.

****

**Gambar 6.Contoh Correct Spelling**

##### **2.2.1.1.5 *Remove Space***

Terkadang sebuah data sering memiliki karakter spasi di depan, di akhir, atau beberapa karakter spasi yang disematkan dimana karakter ini terkadang bisa menyebabkan hasil yang tidak diharapkan saat mengurutkan, memfilter, atau mencari data. Keberadaan spasi berlebih juga bisa menjadi kendala dalam pengolahan data. Maka pada tahap ini tanda spasi yang berlebih akan dihapus untuk membenahi teks-teks tersebut supaya terlihat lebih rapi dan lebih konsisten yang akhirnya akan mempermudah dalam pengolahan data lebih lanjut.

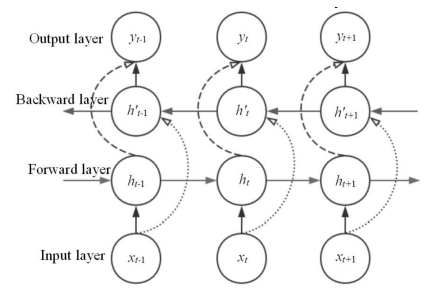
### **2.2.3 *Word Embedding Glove***

Teks memiliki dimensi dan tidak terstruktur, artinya setiap kata unik dapat dilihat sebagai dimensi yang terpisah. Oleh karena itu, *feature extraction* menjadi salah satu kebutuhan untuk pendeteksian objek, *data mining*, serta pengenalan pola dalam *machine learning* yang digunakan untuk mengekstrak fitur berbeda yang ada dalam dataset untuk mewakili dan menggambarkan sebuah data [12].

*Glove* merupakan metode *unsupervised* yang menggunakan matriks *co-occurrence* untuk menghasilkan representasi ruang vektor dari kata-kata. Dilakukan dengan cara menghitung seberapa sering kata-kata yang berbeda muncul dalam sebuah korpus. Metode *GloVe* ini membutuhkan semantik dan konteks yang digunakan untuk menjadi pertimbangan dan tidak menggunakan N-gram yang diterapkan pada data [13].

### **2.2.4 *Modelling with Bi-LSTM***

Setelah dilakukan *word embedding*, selanjutnya akan dilakukan klasifikasi teks menggunakan Bi-LSTM. Input *forward* dan input *backward* merupakan dua jenis masukan yang akan dimasukkan ke dalam arsitektur Bi-LSTM. Bi-LSTM akan sangat bermanfaat dalam hal pelabelan sekuensial apabila memiliki akses terhadap kedua informasi dari sebelum dan sesudahnya.

****

**Gambar 7.Modeling with Bi-LSTM**

### **2.2.5 *Evaluation and Results***

Setelah model selesai dibangun, selanjutnya adalah melakukan evaluasi. Pada tahapan ini proses evaluasi dari hasil yang didapatkan, dilakukan dengan *F1 score*. *F1 score* digunakan karena data yang digunakan sangat tidak seimbang.

## **2.3 Implementasi**

Pada sub bab ini dijelaskan tahap implementasi pemrosesan bahasa alami, yaitu *multi-label classification* menggunakan algoritma Bi-LSTM (*Bidirectional Long Short Term Memory*) dengan *GloVe* sebagai *word embedding*.

### **2.3.1 *Mapping Label***

Berikut merupakan kode program untuk *mapping* (pemetaan) label kedalam 7 label yang telah dikategorikan sebelumnya.

|  |
| --- |
| emotion\_list = ['admiration', 'amusement', 'anger', 'annoyance', 'approval', 'caring', 'confusion', 'curiosity', 'desire', 'disappointment', 'disapproval', 'disgust', 'embarrassment', 'excitement', 'fear','gratitude', 'grief', 'joy', 'love', 'nervousness', 'optimism','pride', 'realization', 'relief', 'remorse', 'sadness', 'surprise', 'neutral']    enkman\_mapping = {          "anger": ["anger", "annoyance", "disapproval"],          "disgust": ["disgust"],          "fear": ["fear", "nervousness"],          "joy": ["joy", "amusement", "approval", "excitement",  "gratitude",  "love", "optimism", "relief", "pride", "admiration", "desire", "caring"],          "sadness": ["sadness", "disappointment", "embarrassment", "grief",  "remorse"],          "surprise": ["surprise", "realization", "confusion", "curiosity"],          "neutral": ["neutral"],          }  enkman\_mapping\_rev = {v:key for key, value in enkman\_mapping.items() for v in value} |

### **2.3.2 *Data Preprocessing***

Pada bagian ini akan dibahas data preprocessing yang dilakukan sebelum digunakan dalam pemodelan, mencakup *data cleaning* yang terdiri atas *clean text*, *clean contractions*, *clean special characters*, *correct spelling*, dan *remove space*.

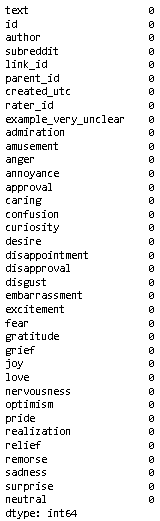
#### **2.3.2.1 Data cleaning**

Pada bagian ini, sebelum melakukan tahapan lainnya perlu dilakukan pemeriksaan terhadap missing value dari data yang digunakan. Berikut adalah kode program dalam mendeteksi *missing value* pada data ditunjukkan pada Gambar 8.



**Gambar 8. Code program mengecek nilai null pada data**

Hasil pendeteksian *missing value* pada data ditampilkan pada Gambar 9. Pada gambar tersebut dapat dilihat ringkasan nilai *null* untuk setiap atribut dan dapat dilihat bahwa setiap atribut pada data tidak memiliki *missing value*.



**Gambar 9.Output pengecekan nilai null**

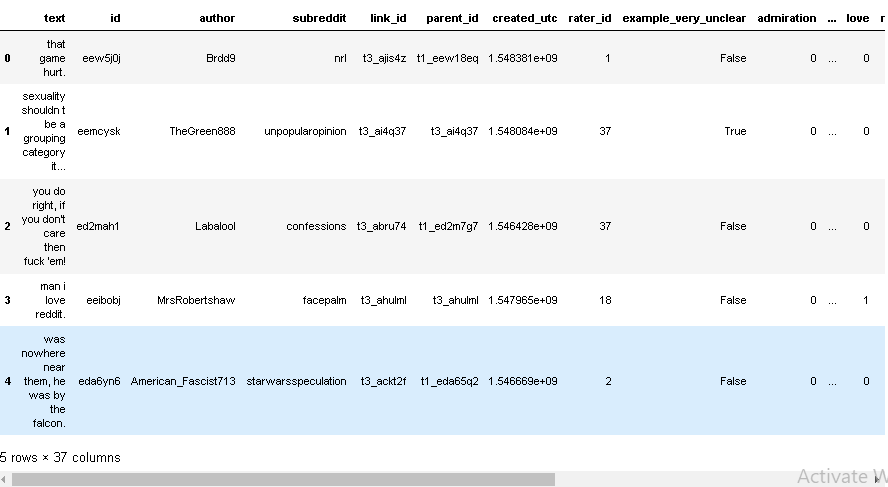
##### **2.3.2.1.1 *Clean Text***

Pada bagian ini *clean text* dilakukan untuk menghapus emoji, mengubah semua teks dalam format huruf kecil (*lowercase*), menghapus teks dalam tanda kurung siku, menghapus link, menghapus tanda baca. Berikut adalah kode program yang dilakukan dalam *clean text* pada data.

|  |
| --- |
| def clean\_text(text):      '''Clean emoji, Make text lowercase, remove text in square brackets, remove links, remove punctuation      and remove words containing numbers.'''      text = emoji.demojize(text)      text = re.sub(r'\:(.\*?)\:','',text)      text = str(text).lower()    #Making Text Lowercase      text = re.sub('\[.\*?\]', '', text)      # the next 2 lines remove html text      text = re.sub('https?://\S+|www\.\S+', '', text)      text = re.sub('<.\*?>+', '', text)      text = re.sub('\n', '', text)      text = re.sub('\w\*\d\w\*', '', text)      # replacing everything with space except (a-z, A-Z, ".", "?", "!", ",", "'")      text = re.sub(r"[^a-zA-Z?.!,¿']+", " ", text)      return text |

**Gambar 10. Kode Program Clean Text**

Hasil *clean text* yang dilakukan pada data yang digunakan ditunjukkan pada Gambar 11.



**Gambar 11. Hasil Clean Text**

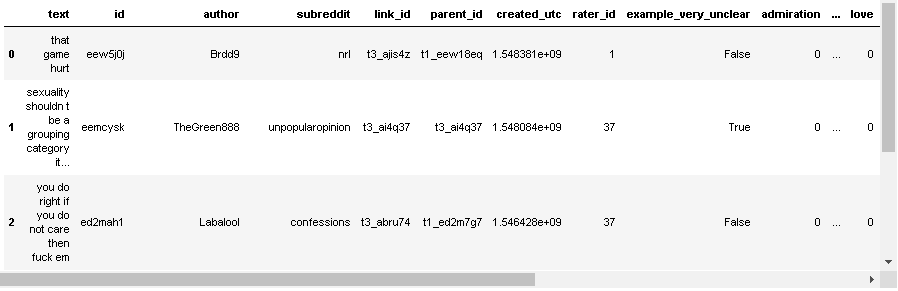
##### **2.3.2.1.2 *Clean Constractions***

Pada bagian ini *clean contractions* dilakukan untuk membakukan teks seperti kata yang ditulis dengan apostrof(‘). Berikut adalah kode program yang dilakukan dalam *clean contractions* pada data.

|  |
| --- |
| def clean\_contractions(text, mapping):      '''Clean contraction using contraction mapping'''      specials = ["’", "‘", "´", "`"]      for s in specials:          text = text.replace(s, "'")      for word in mapping.keys():          if ""+word+"" in text:              text = text.replace(""+word+"", ""+mapping[word]+"")      #Remove Punctuations      text = re.sub('[%s]' % re.escape(string.punctuation), '', text)      # creating a space between a word and the punctuation following it      # eg: "dia sangat manis." => "dia sangat manis ."      text = re.sub(r"([?.!,¿])", r" \1 ", text)      text = re.sub(r'[" "]+', " ", text)      return text |

**Gambar 12. Kode Program Clean Constractions**

Hasil *clean contractions* yang dilakukan pada data yang digunakan ditunjukkan pada Gambar 13.

****

**Gambar 13. Hasil Clean Constractions**

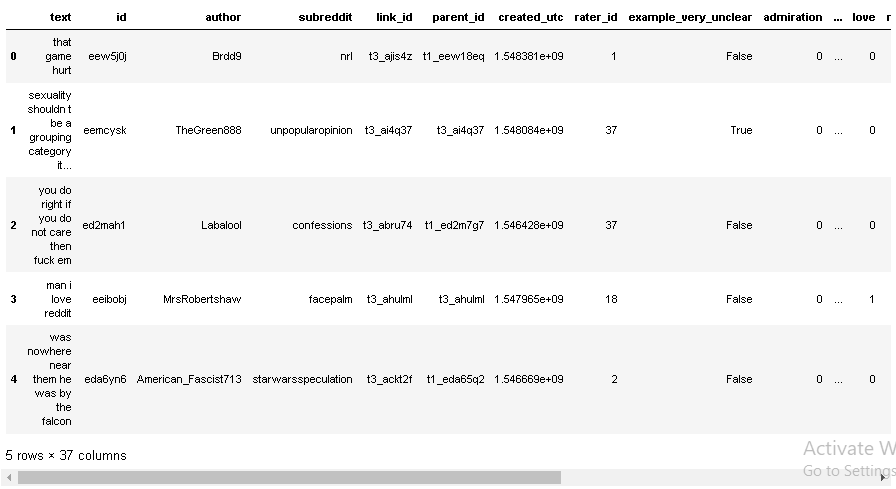
##### **2.3.2.1.3 *Clean Special Charaacters***

Pada bagian ini *clean special characters* dilakukan untuk menghapus *non alphanumeric.* Berikut adalah kode program yang dilakukan dalam *clean special characters* pada data.

|  |
| --- |
| def clean\_special\_chars(text, punct, mapping):      '''Cleans special characters present(if any)'''      for p in mapping:          text = text.replace(p, mapping[p])        for p in punct:          text = text.replace(p, f' {p} ')        specials = {'\u200b': ' ', '…': ' ... ', '\ufeff': '', 'करना': '', 'है': ''}      for s in specials:          text = text.replace(s, specials[s])      return text |

**Gambar 14. Kode Program Clean Special Characters**

Hasil *clean* *special characters* yang dilakukan pada data yang digunakan ditunjukkan pada Gambar 15.

****

**Gambar 15. Hasil Clean Special Character**

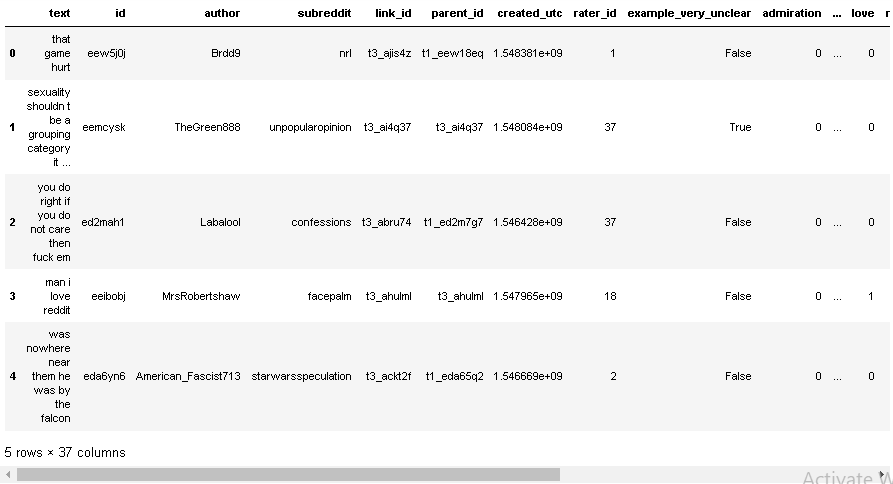
##### **2.3.2.1.4 *Correct Spelling***

Pada bagian ini *correct spelling* dilakukan untuk memperbaiki penulisan ejaan yang salah. Berikut adalah kode program yang dilakukan dalam *correct spelling* pada data.

|  |
| --- |
| def correct\_spelling(x, dic):      '''Corrects common spelling errors'''      for word in dic.keys():          x = x.replace(word, dic[word])      return x |

**Gambar 16. Kode Program Correct Spelling**

Hasil *correct spelling* yang dilakukan pada data yang digunakan ditunjukkan pada Gambar 17.

****

**Gambar 17. Hasil Correct Spelling**

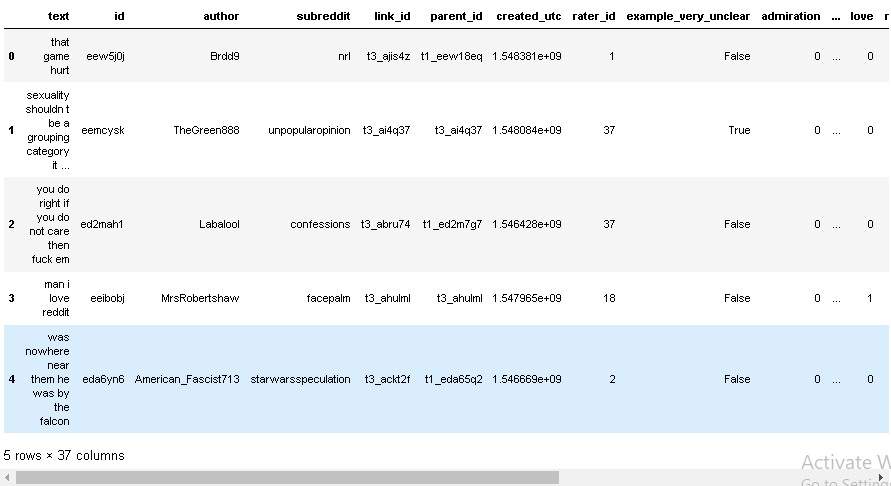
##### **2.3.2.1.5 *Remove Space***

Pada bagian ini *remove space* dilakukan untuk menghapus spasi yang berlebihan agar teks terlihat lebih rapi dan lebih konsisten. Berikut adalah kode program yang dilakukan dalam *remove space* pada data.

|  |
| --- |
| def remove\_space(text):      '''Removes awkward spaces'''      # removes awkward spaces      text = text.strip()      text = text.split()      return " ".join(text) |

**Gambar 18. Kode Program Remove Space**

Hasil *remove space* yang dilakukan pada data yang digunakan ditunjukkan pada Gambar 19.

****

**Gambar 19. Hasil Remove Space**

### **2.3.3 *Word Embedding Glove***

*Word embedding GloVe* digunakan untuk mempelajari hubungan kata-kata dengan menghitung seberapa sering kata-kata muncul bersama satu sama lain dalam sebuah korpus yang diberikan. Dalam *multi-class classification* ini, maka nilai akurasi yang diperoleh dari *embedding GloVe* adalah seperti dibawah ini.

|  |
| --- |
| def create\_embedding\_matrix(filepath, word\_index, embedding\_dim):      vocab\_size = len(word\_index)+1      embedding\_matrix = np.zeros((vocab\_size, embedding\_dim))        with open(filepath,encoding='utf-8') as f:          for line in f:              word, \*vector = line.split()              if word in word\_index:                  idx = word\_index[word]                  embedding\_matrix[idx] = np.array(vector, dtype=np.float32)[:embedding\_dim]        return embedding\_matrix    embedding\_dim = 300  embedding\_matrix = create\_embedding\_matrix('glove.6B.300d.txt', tokenizer.word\_index, embedding\_dim)    nonzero\_elements = np.count\_nonzero(np.count\_nonzero(embedding\_matrix, axis=1))  embedding\_accuracy = nonzero\_elements / vocab\_size  print('embedding accuracy: ' + str(embedding\_accuracy)) |

**Gambar 20. Kode Program Embedding GloVe**

Output:

|  |
| --- |
| embedding accuracy: 0.977577834904313 |

**Gambar 21. Output Embedding GloVe**

### **2.3.3 *Modeling with Bi-LSTM***

|  |
| --- |
| # create the model  model = Sequential()  model.add(Embedding(vocab\_size, embedding\_dim, weights=[embedding\_matrix], input\_length=maxlen, trainable=True))    model.add(Bidirectional(LSTM(256, dropout=0.2, recurrent\_dropout=0.2)))  model.add(Dense(128, activation='relu'))  model.add(Dropout(0.50))  model.add(Dense(28, activation='softmax'))  # Adam Optimiser  model.compile(loss='binary\_crossentropy',optimizer='adam', metrics=['accuracy'])  model.summary() |

**Gambar 22. Kode Program Modelling dengan Bi-LSTM**

Output:

|  |
| --- |
| Model: "sequential"  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Layer (type)                 Output Shape              Param #  =================================================================  embedding (Embedding)        (None, 20, 300)           2100600  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  bidirectional (Bidirectional (None, 512)               1140736  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  dense (Dense)                (None, 128)               65664  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  dropout (Dropout)            (None, 128)               0  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  dense\_1 (Dense)              (None, 28)                3612  ======================Total params: 3,310,612===========================================  Trainable params: 3,310,612  Non-trainable params: 0 |

**Gambar 23. Output Modelling dengan Bi-LSTM**

## **2.4 Hasil**

Pada subbab ini akan dijelaskan evaluasi kuantitatif berdasarkan implementasi *Natural Language Processing*, yaitu *multi-class classification* dengan menggunakan metode Bi-LSTM dan *word embedding GloVe*.

### **2.4.1 Evaluation BI-LSTM Model – Glove**

Berikut adalah kode program evaluasi terhadap model Bi-LSTM dan *word embedding GloVe* yang telah dibangun.

|  |
| --- |
| from sklearn.metrics import precision\_score  from sklearn.metrics import recall\_score  from sklearn.metrics import f1\_score  from sklearn.metrics import classification\_report    #making predictions  y\_pred = model.predict(X\_test)    thresholds=[0.1,0.2,0.25,0.3,0.4,0.5,0.6,0.7,0.8,0.9]  for val in thresholds:      pred=y\_pred.copy()        pred[pred>=val]=1      pred[pred<val]=0        precision = precision\_score(y\_test, pred, average='micro')      recall = recall\_score(y\_test, pred, average='micro')      f1 = f1\_score(y\_test, pred, average='micro')        print("Threshold: {:.4f}, Precision: {:.4f}, Recall: {:.4f}, F1-measure: {:.4f}".format(val, precision, recall, f1)) |

**Gambar 24. Kode Evaluasi Bi-LSTM - Glove**

Output:

|  |
| --- |
| Threshold: 0.1000, Precision: 0.5056, Recall: 0.5540, F1-measure: 0.5287  Threshold: 0.2000, Precision: 0.5200, Recall: 0.5305, F1-measure: 0.5252  Threshold: 0.2500, Precision: 0.5269, Recall: 0.5244, F1-measure: 0.5256  Threshold: 0.3000, Precision: 0.5335, Recall: 0.5164, F1-measure: 0.5248  Threshold: 0.4000, Precision: 0.5432, Recall: 0.5075, F1-measure: 0.5248  Threshold: 0.5000, Precision: 0.5493, Recall: 0.4972, F1-measure: 0.5219  Threshold: 0.6000, Precision: 0.5571, Recall: 0.4878, F1-measure: 0.5202  Threshold: 0.7000, Precision: 0.5602, Recall: 0.4737, F1-measure: 0.5134  Threshold: 0.8000, Precision: 0.5727, Recall: 0.4587, F1-measure: 0.5094  Threshold: 0.9000, Precision: 0.5902, Recall: 0.4394, F1-measure: 0.5038 |

**Gambar 25. Output Evaluasi Bi-LSTM - Glove**

### **2.4.2** ***Accuracy Bi-LSTM Model with GloVe***

Berikut merupakan kode program untuk melihat akurasi yang dihasilkan melalui penggabungan Bi-LSTM dengan *GloVe word embedding* :

|  |
| --- |
| pred = y\_pred.copy()  pred[pred>=0.2] = 1  pred[pred<0.2] =0  print("Average F1-Score across Multi-Lables: {}".format(f1\_score(y\_test, pred, average='micro'))) |

**Gambar 26. Kode Program Akurasi Bi-LSTM – Glove**

Output:

|  |
| --- |
| Average F1-Score across Multi-Labels: 0.5252149663025795 |

**Gambar 27. Output Akurasi Bi-LSTM - Glove**