Membuat API untuk Analisis Sentimen dan Laporan Analisis Data berdasarkan Sentimen.

Di Presentasikan Oleh Kelompok 3:

Boby Z
Muhammad Ilham Zakaria
Muhammad Ramadhani Jatmika

Pendahuluan

LATAR BELAKANG MASALAH

Indonesia merupakan negara dengan pengguna internet terbanyak ke empat di dunia. Jumlah pengguna media sosial di Indonesia merupakan urutan ke 5 terbesar di dunia. Dalam era internet ini banyak sekali konten-konten atau informasi apapun khususnya didalam sosial media baik twitter, instagram ataupun yang lainnya.

RUMUSAN MASALAH

Berdasarkan latar belakang di atas, diketahui bahwa keaktifan pengguna sosial media di Indonesia sangat tinggi. Ini terlihat dari banyaknya komentar-komentar berbahasa Indonesia di media sosial. Oleh karena itu kita rasa perlu untuk membuat sebuah sistem analisis sentiment komentar-komentar tersebut untuk mengantisipasi keperluan-keperluan lebih lanjut yang mungkin muncul dimasa mendatang.

TUJUAN

Proyek ini bertujuan untuk membuat sebuah engine/API yang bisa memilah komentar positif, netral, dan negatif dari komentar netizen dari teks non-formal dengan menggunakan 2 model machine learning yaitu model neural network (MLPClassifier) dan model LSTM (Long Short Term Memory).

Metodologi

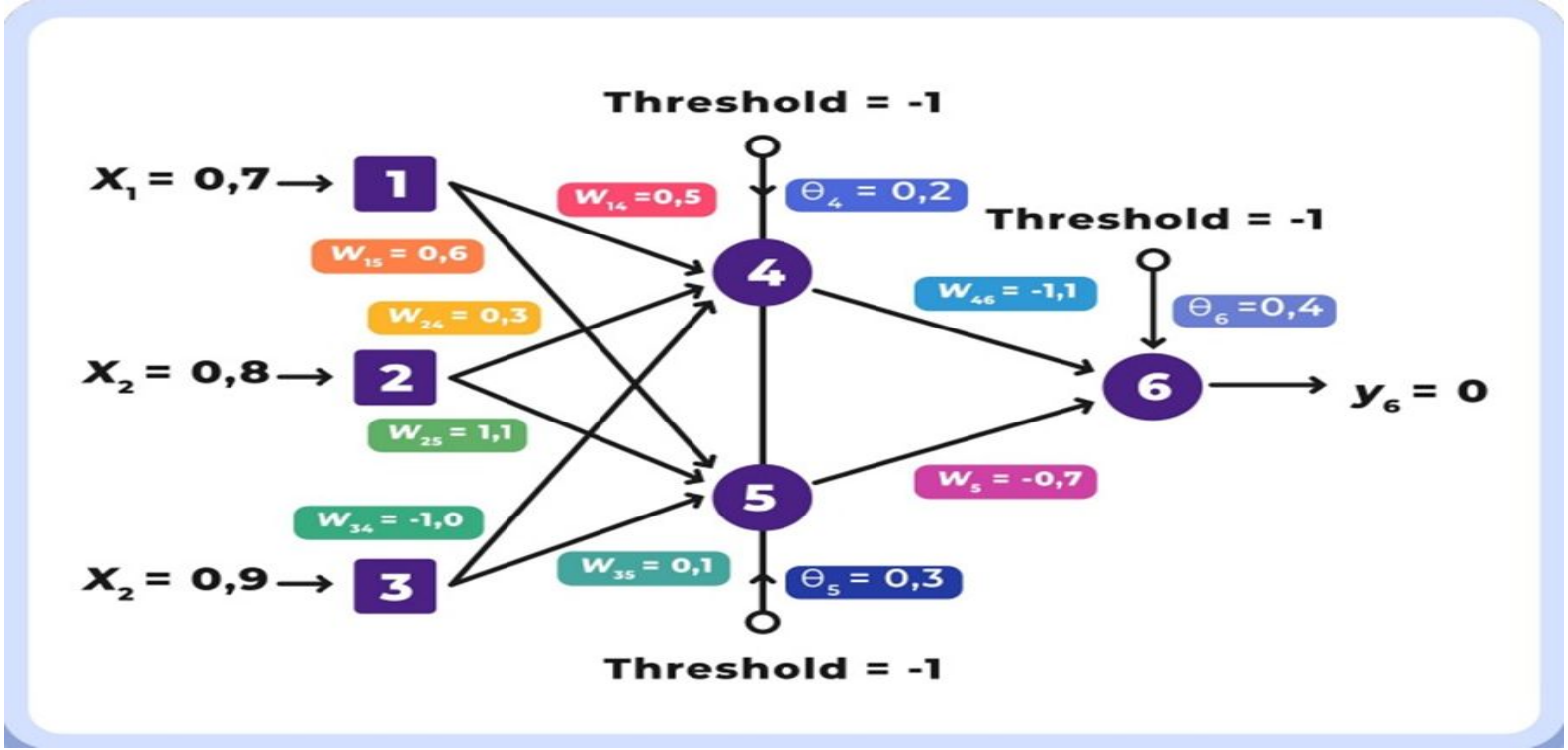
Berdasarkan dari rumusan masalah yang ada dan tujuan yang telah didefinisikan, serta sesuai dengan rekomendasi yang diberikan oleh Binar Academy. Kami menyusun langkah-langkah pengerjaan proyek ini sebagai berikut:

1. Melakukan kalkulasi analisis sentimen menggunakan Neural Network
2. Membaca dan memahami studi kasus yang diberikan
3. Melakukan cleansing data pada dataset Analisis Sentimen menggunakan Pandas dan RegEx
4. Melakukan feature extraction pada dataset Analisis Sentimen menggunakan Sklearn
5. Melakukan training menggunakan 2 metode:
    a. Neural network (memakai tool Sklearn)
    b. LSTM (memakai tool Tensorflow)
6. Melakukan evaluasi pada model Neural Network dan LSTM yang sudah di-training dengan Sklearn
7. Membangun API untuk prediksi sentimen menggunakan model Neural Network dan LSTM dengan menggunakan Flask dan Swagger UI dengan 2 endpoint untuk masing-masing model.
8. Membuat laporan hasil pengerjaan proyek
9. Mendokumentasikan hasil koding (source code) termasuk API dan laporan dengan mengunggahnya di Git dan Github.

Hasil dan Kesimpulan

Langkah Pertama

Dalam rangka memahami bagaimana sebuah neural network bekerja pertama-tama kita perlu melakukan perhitungan backward propagation yang merupakan sebuah teknik untuk memperbaharui weight dan bias yang digunakan untuk mereduksi nilai error sampai ke level minimum. Proses inilah yang menjadi inti bagaimana neural network bekerja.
Arsitektur neural network yang akan dihitung telah disediakan oleh Binar Academy. Kami melakukan perhitungan ini dalam file Ms. Excel dan memindahkan hasilnya ke dalam Ms. Word


Threshold = -1
$X_1 = 0,7$
1
$W_{14} = 0,5$
$\Theta_4 = 0,2$
Threshold = -1
$W_{15} = 0,6$
4
$W_{46} = -1,1$
$\Theta_6 = 0,4$
$W_{24} = 0,3$
$X_2 = 0,8$
2
6
$y_6 = 0$
$W_{25} = 1,1$
$W_5 = -0,7$
5
$W_{34} = -1,0$
$X_2 = 0,9$
3
$W_{35} = 0,1$
$\Theta_5 = 0,3$
Threshold = -1

Step	Item	Symbol	Formula	Value
Initial Values	Input 1	x_1		0,7
	Input 2	x_2		0,8
	Input 3	x_3		0,9
	Learning Rate	α		0,1
	True Output	$Y_{d,6}$		0,0
Initial Random	Weight Untuk Neuron 1,4	$w_{1,4}$		0,5
	Weight Untuk Neuron 1,5	$w_{1,5}$		0,6
	Weight Untuk Neuron 2,4	$w_{2,4}$		0,3
	Weight Untuk Neuron 2,5	$w_{2,5}$		1,1
	Weight Untuk Neuron 3,4	$w_{3,4}$		-1,0
	Weight Untuk Neuron 3,5	$w_{3,5}$		0,1
	Weight Untuk Neuron 4,6	$w_{4,6}$		-1,1
	Weight Untuk Neuron 5,6	$w_{5,6}$		-0,7
	Threshold 4,5,6	T		-1,0
	Theta Untuk Neuron 4	θ_4		0,2
	Theta Untuk Neuron 5	θ_5		0,3
	Theta Untuk Neuron 6	θ_6		0,4

Proses 1	Output Neuron 4	y_4	$y_4 = 1/(1+e^{-(x_1 * w_{1,4} + x_2 * w_{2,4} + x_3 * w_{3,4} + T\theta_4)})$	0,6177
	Output Neuron 5	y_5	$y_5 = 1/(1+e^{-(x_1 * w_{1,5} + x_2 * w_{2,5} + x_3 * w_{3,5} + T\theta_5)})$	0,7484
	Output Neuron 6	y_6	$y_6 = 1/(1+e^{-(y_4 * w_{4,6} + y_5 * w_{5,6} + T\theta_6)})$	0,1675
	Error di sisi output (Neuron 6)	e	$e = y_{d,6} - y_6$	-0,1675
Proses 2	Error Gradient Neuron 6	δ_6	$\delta_6 = y_6 * (1-y_6) * e$	-0,0234
	Weight Correction Untuk Neuron 4,6	$\Delta w_{4,6}$	$\Delta w_{4,6} = \alpha * y_4 * \delta_6$	-0,0014
	Weight Correction Untuk Neuron 5,6	$\Delta w_{5,6}$	$\Delta w_{5,6} = \alpha * y_5 * \delta_6$	-0,0017
	Theta Correction Untuk Neuron 6	$\Delta\theta_6$	$\Delta\theta_6 = \alpha * \theta_6 * \delta_6$	-0,0009
Proses 3	Error Gradient Neuron 4	δ_4	$\delta_4 = y_4 * (1-y_4) * \delta_6 * w_{4,6}$	0,0061
	Error Gradient Neuron 5	δ_5	$\delta_5 = y_5 * (1-y_5) * \delta_6 * w_{5,6}$	0,0031
Proses 4	Weight Correction Untuk Neuron 1,4	$\Delta w_{1,4}$	$\Delta w_{1,4} = \alpha * x_1 * \delta_4$	0,0004
	Weight Correction Untuk Neuron 2,4	$\Delta w_{2,4}$	$\Delta w_{2,4} = \alpha * x_2 * \delta_4$	0,0005
	Weight Correction Untuk Neuron 3,4	$\Delta w_{3,4}$	$\Delta w_{3,4} = \alpha * x_3 * \delta_4$	0,0005
	Theta Correction Untuk Neuron 4	$\Delta\theta_4$	$\Delta\theta_4 = \alpha * \theta_4 * \delta_4$	0,0001
	Weight Correction Untuk Neuron 1,5	$\Delta w_{1,5}$	$\Delta w_{1,5} = \alpha * x_1 * \delta_5$	0,0002
	Weight Correction Untuk Neuron 2,5	$\Delta w_{2,5}$	$\Delta w_{2,5} = \alpha * x_2 * \delta_5$	0,0002
	Weight Correction Untuk Neuron 3,5	$\Delta w_{3,5}$	$\Delta w_{3,5} = \alpha * x_3 * \delta_5$	0,0003
	Theta Correction Untuk Neuron 5	$\Delta\theta_5$	$\Delta\theta_5 = \alpha * \theta_5 * \delta_5$	0,0001

Proses 5	Updated Weight Untuk Neuron 1,4	$w_{1,4}'$	$w_{1,4}' = w_{1,4} + \Delta w_{1,4}$	0,5004
	Updated Weight Untuk Neuron 1,5	$w_{1,5}'$	$w_{1,5}' = w_{1,5} + \Delta w_{1,5}$	0,6002
	Updated Weight Untuk Neuron 2,4	$w_{2,4}'$	$w_{2,4}' = w_{2,4} + \Delta w_{2,4}$	0,3005
	Updated Weight Untuk Neuron 2,5	$w_{2,5}'$	$w_{2,5}' = w_{2,5} + \Delta w_{2,5}$	1,1002
	Updated Weight Untuk Neuron 3,4	$w_{3,4}'$	$w_{3,4}' = w_{3,4} + \Delta w_{3,4}$	-0,9995
	Updated Weight Untuk Neuron 3,5	$w_{3,5}'$	$w_{3,5}' = w_{3,5} + \Delta w_{3,5}$	0,1003
	Updated Weight Untuk Neuron 4,6	$w_{4,6}'$	$w_{4,6}' = w_{4,6} + \Delta w_{4,6}$	-1,1014
	Updated Weight Untuk Neuron 4,6	$w_{5,6}'$	$w_{5,6}' = w_{5,6} + \Delta w_{5,6}$	-0,7017
	Updated Theta Untuk Neuron 4	θ_4'	$\theta_4' = \theta_4 + \Delta\theta_4$	0,2001
	Updated Theta Untuk Neuron 5	θ_5'	$\theta_5' = \theta_5 + \Delta\theta_5$	0,3001
	Updated Theta Untuk Neuron 6	θ_6'	$\theta_6' = \theta_6 + \Delta\theta_6$	0,3991

Hasil dan Kesimpulan

Langkah Kedua

Membaca dan memahami studi kasus yang telah kita pelajari dari hasil penelitian-penelitian sebelumnya yang berhubungan dengan kasus yang akan kita pecahkan. Beberapa hasil penelitian-penelitian sebelumnya yang telah kami pelajari adalah sebagai berikut:

- https://www.kaggle.com/code/atillasilva/sentiment-analysis-using-neural-network
- https://www.kaggle.com/code/ngyptr/lstm-sentiment-analysis-keras
- https://www.kaggle.com/code/jth359/imbalanced-target-variable-with-text-data/notebook?scriptVersionId=53696145

Hasil dan Kesimpulan

Langkah Ketiga

Kami melakukan cleansing data pada dataset Analisis Sentimen menggunakan Pandas dan RegEx dengan langkah-langkah berikut :

- Prepare Data

  Dataset yang kami gunakan adalah dataset yang kami ambil dari dokumen challenge https://drive.google.com/file/d/1RCHGfn9JJyyReAh8PIIoF8Ch0H3miP0u/view?usp=sharing. Setelah kami eksplorasi data tersebut ternyata hasilnya imbalanced, oleh karena itu kami melakukan *text augmentation* untuk mengatasi problem data imbalanced tersebut. Teknik ini praktis dilakukan dengan mentranslasi teks yang berkategori tertentu yang jumlahnya relatif lebih sedikit dibandingkan dengan teks yang berkategori dominan, dari bahasa indonesia ke bahasa lain kemudian ditranslasi kembali ke bahasa indonesia. Hal ini dilakukan untuk menyeimbangkan jumlah teks untuk semua kategori.

- Load Data

  Setelah dataset diunduh kemudian dimasukan kedalam dataframe dengan menggunakan library Pandas Python.

- Clean Data

  Proses pembersihan data dilakukan dengan mengubah seluruh teks ke format lowercase, menghilangkan karakter non alpha numeric, normalisasi kata-kata “alay”, menghilangkan stopword dan terakhir proses stemming.

Testing Selesai!

	precision	recall	f1-score	support
negative	0.77	0.83	0.80	867
neutral	0.72	0.59	0.65	281
positive	0.88	0.87	0.87	1404
accuracy			0.82	2552
macro avg	0.79	0.76	0.77	2552
weighted avg	0.82	0.82	0.82	2552

Testing Selesai!

	precision	recall	f1-score	support
negative	0.84	0.86	0.85	1570
neutral	0.93	0.96	0.95	1626
positive	0.89	0.83	0.86	1530
accuracy			0.89	4726
macro avg	0.89	0.89	0.89	4726
weighted avg	0.89	0.89	0.89	4726

```
from googletrans import Translator

translator = Translator(service_urls = ["translate.googleapis.com"])

# translate to English
to_en = translator.translate(positive_list[2], dest = 'en')
# translate back to Indonesian
to_id = translator.translate(to_en.text, dest = 'id').text
to_id
```

[32]

```
'betapa senangnya saya ketika saya membuka kotak paket dan barangnya bagus! atur beli lagi!'
```

```
# Generating new negative text, from existing negative text, by translating it to English and back to Indonesian
# ~20:xxpm-
translated_negative_list = []

for data in negative_list[:]:

    # translate to English and back to Indonesian
    to_en = translator.translate(data, dest = 'en')
    to_id = translator.translate(to_en.text, dest = 'id').text
    translated_negative_list.append(to_id)

translated_negative_list
```

[65]

```python
def cleansing(text):

    # Lower Case Operation
    text = text.lower()

    # Removing Unnecessary Characters
    text = re.sub('\n', ' ', text)
    text = re.sub('rt', ' ', text)
    text = re.sub('user', ' ', text)
    text = re.sub(r'((www\.[^\s]+)|(http?://[^\s]+)|(https?://[^\s]+))', ' ', text)
    text = re.sub(' +', ' ', text)

    # Removing Non-Alphanumeric Characters
    text = re.sub('[^a-zA-Z0-9]+', ' ', text)

    # Normalizing Alay Words
    text = ' '.join([alay_dict_map[word] if word in alay_dict_map else word for word in text.split(' ')])

    # Removing Stopword
    text = ' '.join(['' if word in id_stopword_dict.stopword.values else word for word in text.split(' ')])
    text = text.strip()

    # Stemming
    text = stemmer.stem(text)

    return text
```

Hasil dan Kesimpulan

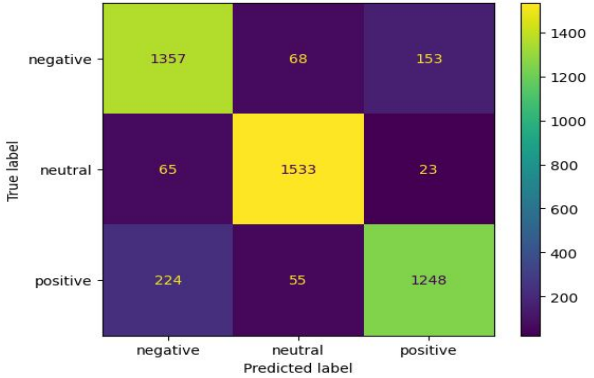
Langkah Keempat

Kami melakukan feature extraction untuk model neural network (MLP Classifier) dengan menggunakan CountVectorizer() dan menggunakan TfidfVectorizer(). Kami menemukan penggunaan TfidfVectorizer() menghasilkan nilai akurasi yang sedikit lebih baik dibandingkan dengan CountVectorizer(). Sehingga untuk model neural network (MLP Classifier) kami memilih menggunakan TfidfVectorizer().
Sedangkan untuk model LSTM (Long Short Term Memory) kami melakukan feature extraction menggunakan Tokenizer() dan pad_sequences().

```python
from sklearn.feature_extraction.text import CountVectorizer

# Kita proses Feature Extraction
count_vect = CountVectorizer()
count_vect.fit(data_preprocessed)

X = count_vect.transform(data_preprocessed)
print ("Feature Extraction Berhasil!")
```

```
Testing Selesai!
              precision    recall  f1-score   support

    negative       0.82      0.86      0.84      1578
     neutral       0.93      0.95      0.94      1621
    positive       0.88      0.82      0.85      1527

    accuracy                           0.88      4726
   macro avg       0.88      0.87      0.87      4726
weighted avg       0.88      0.88      0.88      4726
```

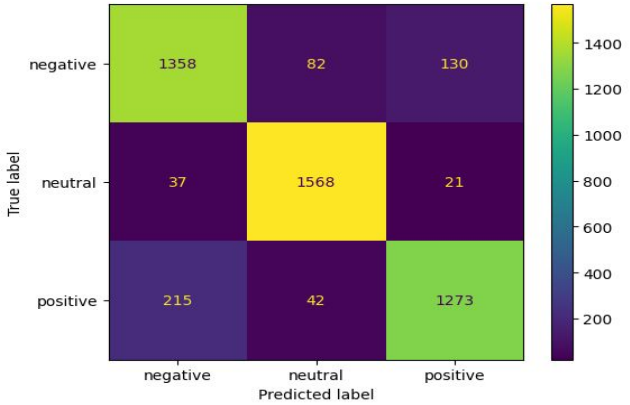
Sentiment Analysis With CountVectorizer and Neural Network (MLPClassifier)



```python
from sklearn.feature_extraction.text import TfidfVectorizer

# Proses Feature Extraction
tfidf_vect = TfidfVectorizer()
tfidf_vect.fit(data_preprocessed)

X = tfidf_vect.transform(data_preprocessed)
print ("Feature Extraction Berhasil!")
```

```
Testing Selesai!
              precision    recall  f1-score   support

    negative       0.84      0.86      0.85      1570
     neutral       0.93      0.96      0.95      1626
    positive       0.89      0.83      0.86      1530

    accuracy                           0.89      4726
   macro avg       0.89      0.89      0.89      4726
weighted avg       0.89      0.89      0.89      4726
```

Sentiment Analysis With TFIDF Vectorizer and Neural Network (MLPClassifier)

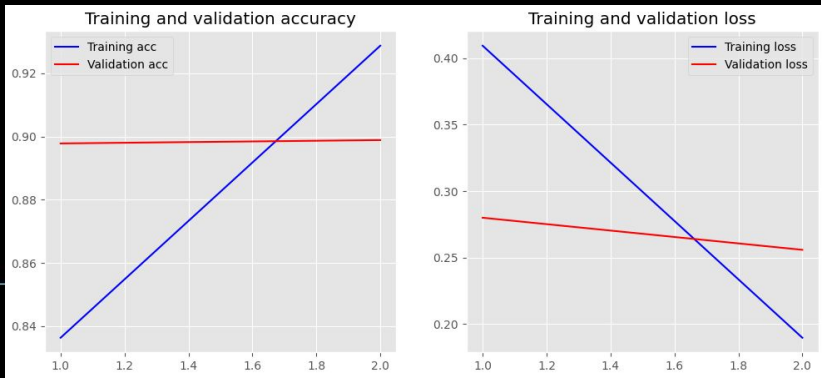


```python
import pickle
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from collections import defaultdict

max_features = 100000
tokenizer = Tokenizer(num_words=max_features, split=' ', lower=True)
tokenizer.fit_on_texts(total_data)
with open('tokenizer.pickle', 'wb') as handle:
    pickle.dump(tokenizer, handle, protocol=pickle.HIGHEST_PROTOCOL)
    print("tokenizer.pickle has been created!")

X = tokenizer.texts_to_sequences(total_data)

vocab_size = len(tokenizer.word_index)
maxlen = max(len(x) for x in X)

X = pad_sequences(X)
with open('x_pad_sequences.pickle', 'wb') as handle:
    pickle.dump(X, handle, protocol=pickle.HIGHEST_PROTOCOL)
    print("x_pad_sequences.pickle has been created!")
```

```
tokenizer.pickle has been created!
x_pad_sequences.pickle has been created!
```

Testing selesai

	precision	recall	f1-score	support
0	0.90	0.84	0.87	1567
1	0.95	0.94	0.95	1625
2	0.87	0.93	0.90	1534
accuracy			0.91	4726
macro avg	0.91	0.91	0.91	4726
weighted avg	0.91	0.91	0.91	4726

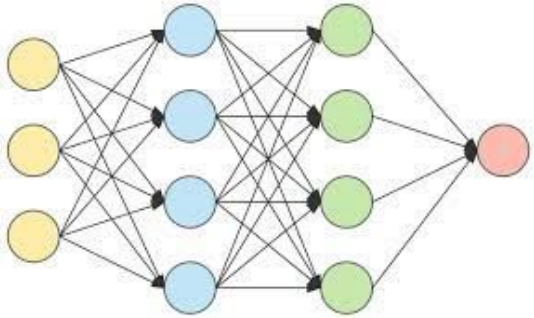


Hasil dan Kesimpulan

Langkah Kelima

Melakukan training menggunakan 2 metode:

a. Neural network (memakai tool Sklearn)
b. LSTM (memakai tool Tensorflow)



Neural Network

```
from sklearn.neural_network import MLPClassifier

model = MLPClassifier()
model.get_params()
```

```
model.fit(X_train, y_train)
pickle.dump(model, open("nn_aug_tfidf.model", "wb"))

print ("Training Berhasil!")
```

```
Training Berhasil!
```

LSTM

```
import numpy as np
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Embedding, LSTM, SpatialDropout1D, SimpleRNN, Activation
from tensorflow.keras import optimizers
from tensorflow.keras.callbacks import EarlyStopping, TensorBoard
from tensorflow.keras.layers import Flatten
from tensorflow.keras import backend as K

embed_dim = 100
units = 64

model = Sequential()
model.add(Embedding(max_features, embed_dim, input_length=X.shape[1]))
model.add(LSTM(units, dropout=0.2))
model.add(Dense(3,activation='softmax'))
model.compile(loss = 'binary_crossentropy', optimizer='adam',metrics = ['accuracy'])
print(model.summary())

adam = optimizers.Adam(lr = 0.001)
model.compile(loss = 'categorical_crossentropy', optimizer = adam, metrics = ['accuracy'])

es = EarlyStopping(monitor='val_loss', mode='min', verbose=1)
history = model.fit(X_train, y_train, epochs=10, batch_size=10, validation_data=(X_test, y_test), verbose=1, callbacks=[es])
```

Hasil dan Kesimpulan

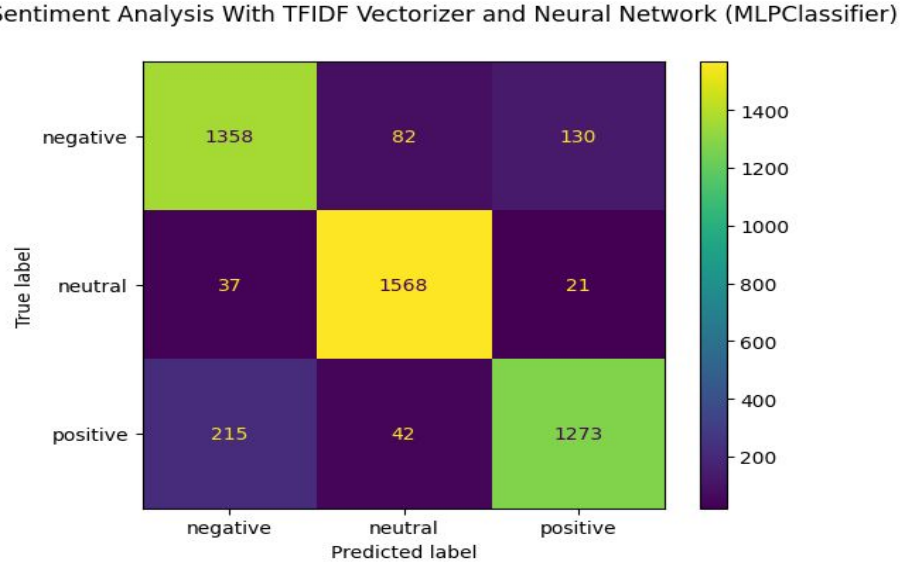
Langkah Keenam

Melakukan evaluasi pada model Neural Network dan LSTM yang sudah di-training dengan Sklearn

Neural Network

```
from sklearn.metrics import classification_report

test = model.predict(X_test)

print ("Testing Selesai!")

print(classification_report(y_test, test))
```

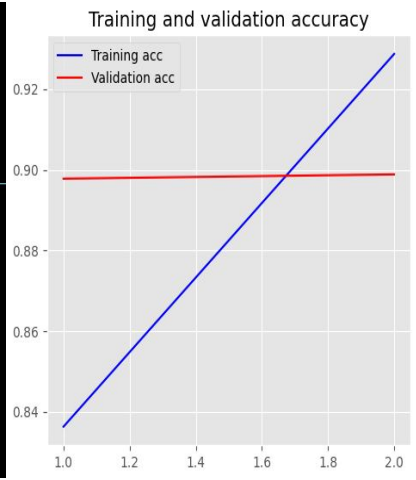
```
Testing Selesai!
              precision    recall  f1-score   support

    negative       0.84      0.86      0.85      1570
     neutral       0.93      0.96      0.95      1626
    positive       0.89      0.83      0.86      1530

    accuracy                           0.89      4726
   macro avg       0.89      0.89      0.89      4726
weighted avg       0.89      0.89      0.89      4726
```

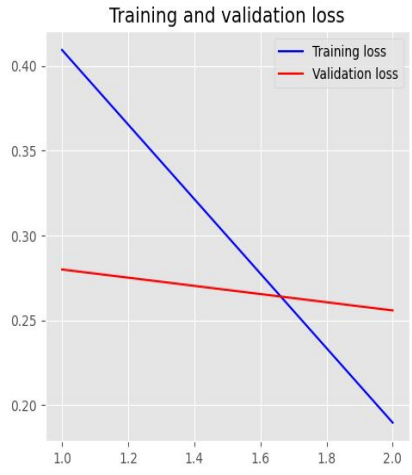


LSTM

```python
from sklearn import metrics

predictions = model.predict(X_test)
y_pred = predictions
matrix_test = metrics.classification_report(y_test.argmax(axis=1), y_pred.argmax(axis=1))
print("Testing selesai")
print(matrix_test)
```

```
148/148 [==============================] - 5s 33ms/step
Testing selesai
              precision    recall  f1-score   support

           0       0.90      0.84      0.87      1567
           1       0.95      0.94      0.95      1625
           2       0.87      0.93      0.90      1534

    accuracy                           0.91      4726
   macro avg       0.91      0.91      0.91      4726
weighted avg       0.91      0.91      0.91      4726
```





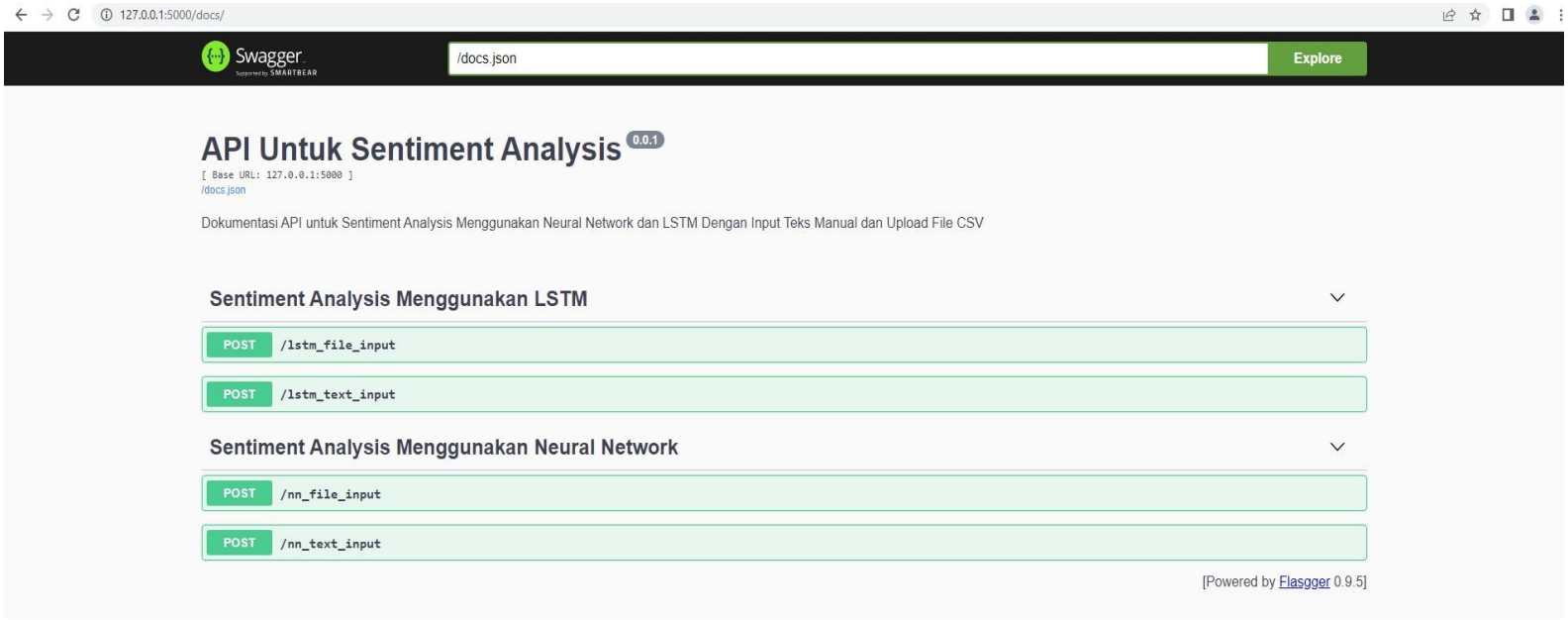
Hasil dan Kesimpulan

Langkah Ketujuh

Membangun API untuk prediksi sentimen menggunakan model Neural Network dan LSTM dengan menggunakan Flask dan Swagger UI dengan 2 endpoint untuk masing-masing model.

```python
from functions import *

app = Flask(__name__)
app.json_encoder = LazyJSONEncoder
swagger_template = dict(
info = {
    'title': LazyString(lambda: 'API Untuk Sentiment Analysis'),
    'version': LazyString(lambda: '0.0.1'),
    'description': LazyString(lambda: 'Dokumentasi API untuk Sentiment Analys
    },
    host = LazyString(lambda: request.host)
)
swagger_config = {
    "headers": [],
    "specs": [
        {
            "endpoint": 'docs',
            "route": '/docs.json'
        }
    ],
    "static_url_path": '/flasgger_static',
    "swagger_ui": True,
    "specs_route": '/docs/',
    # "ui_params": {
    #   "operationsSorter": "method",
    #   "tagsSorter": "method"
    # }
}
swagger = Swagger(app, template=swagger_template, config=swagger_config)
```

```python
swagger = Swagger(app, template=swagger_template, config=swagger_config)


@swag_from("D:\BINAR ZOOM\platinum\docs\nn_text_input.yml", methods=['POST'])
@app.route('/nn_text_input', methods=['POST'])
def nn_text():

    original_text = request.form.get('Text')

    return nn_text_process(original_text)

@swag_from("D:\BINAR ZOOM\platinum\docs\nn_file_input.yml", methods=['POST'])
@app.route('/nn_file_input', methods=['POST'])
def nn_file():

    file = request.files['Upfile']

    return nn_file_process(file)

@swag_from("D:\BINAR ZOOM\platinum\docs\lstm_text_input.yml", methods=['POST'])
@app.route('/lstm_text_input', methods=['POST'])
def lstm_text():

    original_text = request.form.get('Text')

    return lstm_text_process(original_text)

@swag_from("D:\BINAR ZOOM\platinum\docs\lstm_file_input.yml", methods=['POST'])
@app.route('/lstm_file_input', methods=['POST'])
def lstm_file():

    file = request.files['Upfile']

    return lstm_file_process(file)

if __name__ == '__main__':
    app.run()
```

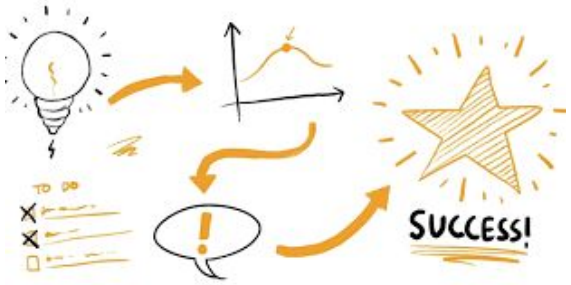

127.0.0.1:5000/docs/
Swagger
Supported by SMARTBEAR
/docs.json
Explore
API Untuk Sentiment Analysis
0.0.1
[Base URL: 127.0.0.1:5000]
/docs.json
Dokumentasi API untuk Sentiment Analysis Menggunakan Neural Network dan LSTM Dengan Input Teks Manual dan Upload File CSV
Sentiment Analysis Menggunakan LSTM
POST /lstm_file_input
POST /lstm_text_input
Sentiment Analysis Menggunakan Neural Network
POST /nn_file_input
POST /nn_text_input
[Powered by Flasgger 0.9.5]

Hasil dan Kesimpulan

Dari hasil pengerjaan dan analisis yang dilakukan, berikut beberapa hal yang dapat kami simpulkan:

1. Neural network bekerja dengan mengkombinasikan proses *forward propagation* dan *backward propagation*. Proses *forward propagation* berjalan dengan menginisiasi nilai awal dari *weight* dan *bias* dari setiap koneksi dan *node* neural network. Lalu proses *backward propagation* dilakukan untuk memperbaharui nilai *weight* dan *bias* dari neural network dalam upaya mereduksi nilai *error/loss*. Proses ini dilakukan berulang-ulang hingga mencapai nilai *minimum* atau hingga *epoch* berakhir.
2. Dataset awal yang kita unduh memiliki ketidak seimbangan jumlah data antara label positif, netral dan negatif. Hal ini dapat diperbaiki salah satunya dengan cara *text augmentation*.
3. Proses *data cleansing* perlu dilakukan untuk menormalisasi dan menstandarisasi data.
4. Terdapat perbedaan akurasi model yang menggunakan teknik *feature extraction* CountVectorizer dan TfidfVectorizer. TfidfVectorizer menghasilkan tingkat akurasi yang sedikit lebih baik.
5. Proses *training* model merupakan proses dari *forward propagation* dan *backward propagation* yang dilakukan berulang-ulang hingga mencapai nilai *error* minimum atau hingga *epoch* berakhir.
6. Hasil evaluasi model LSTM (Long Short Term Memory) memiliki tingkat akurasi yang lebih tinggi dari hasil evaluasi model neural network (MLP Classifier).
7. Proses *deployment* model yang telah di *train* dapat dilakukan dengan mudah menggunakan Flask API dan Swagger.

Saran



Berdasarkan hasil pengerjaan proyek ini berikut beberapa hal yang dapat kami sarankan untuk proyek-proyek sejenis dimasa mendatang:

1. Kita perlu mengeksplorasi teknik-teknik lain untuk menyikapi fenomena *unbalanced* dataset.
2. Kita perlu mengeksplorasi teknik-teknik lain untuk proses *feature extraction*.
3. Kita perlu mengeksplorasi teknik *hyper parameter tuning* untuk meningkatkan akurasi model.
4. Kita perlu mengeksplorasi penggunaan model-model lain untuk keperluan *sentiment analysis*.
5. Kita perlu mengeksplotasi penggunaan *tools* lain untuk *deployment* model yang telah di *train*.

Penutup

Sebagai penutup presentasi ini, kami ini mengucap terimakasih kepada:

1. Binar Academy
2. Fasilitator
3. Teman-teman sekelas
4. Teman-teman satu tim
5. Dan *support system* lainnya.