

A stylized illustration of a workspace on a dark blue background. It includes a laptop with a teal screen and keyboard, a stack of books, a potted plant with long green leaves, a pen holder with three pens, and a tablet showing a map. The text is in a light blue, sans-serif font.

Challenge Platinum : API for Sentiment Analyst

Nama :

I Gede Dhani Pradipta Putra

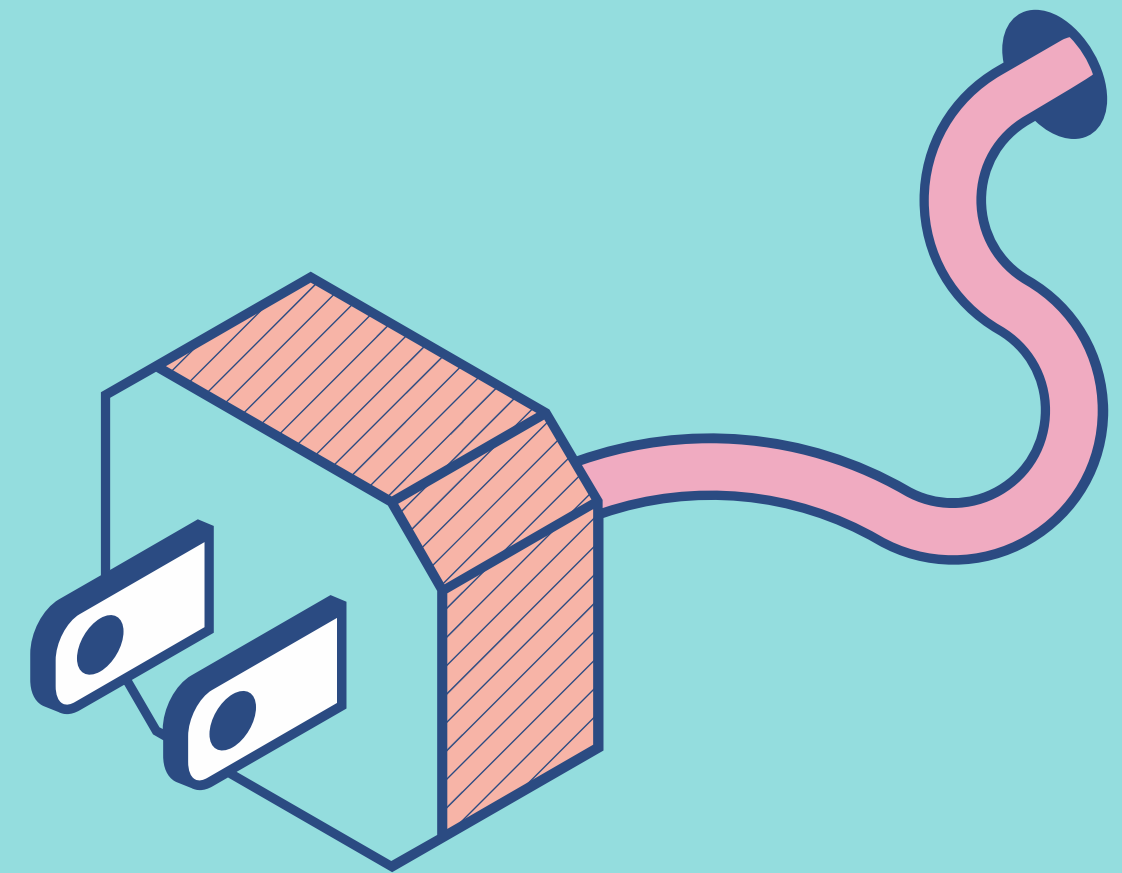
Junio Faathir Anand Ansyori

Mohammad Raditya Syahmaulana

Summary

Melakukan analisis sentimen dengan pembuatan model menggunakan Neural Network dan LSTM dengan mendeploynya dengan API Flask dan Swagger UI.

Pada endpoint dapat menginput teks dan mengupload file untuk menganalisis sentimen dan mengcleansingnya. Data yang di upload diberikan oleh pihak binar



Roadmap Pembuatan Model

1

Prepare Dataset

Menganalisa data, dan melihat label jumlah positif, negatif, dan netral.

2

Cleansing Text

Membersihkan teks dengan menggunakan cleansing function.

3

Feature Extraction

Mengubah setiap kata menjadi vector menggunakan CountVectorizer(N N) dan Tokenizer(LSTM).

Roadmap Pembuatan Model

4

5

6

Prepare and Validation Data

Menentukan jumlah data yang akan di train dan di test

Model Selection

Untuk NN menggunakan MLPClassifier from sklearn dan LSTM menggunakan deep learning seperti TensorFlow


Training Model dan Save Model

Melakukan training dan validation data setelah itu melakukan saving data

Hasil Model Dari Neural Network

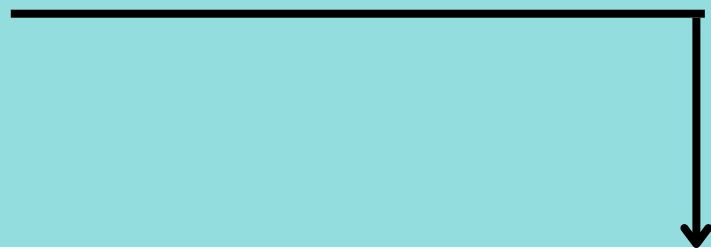
- Terdapat 11000 data label yang terdiri dari :

1.Positive : 6416
2.Negative : 3436
3.Netral : 1148




```
In [4]: df.Label.value_counts()
Out[4]: positive      6416
        negative     3436
        neutral      1148
        Name: Label, dtype: int64
```

- Melakukan Cleansing menggunakan functionl



```
In [5]: def cleansing(sent):
        # Mengubah kata menjadi huruf kecil semua dengan menggunakan fungsi lower()
        string = sent.lower()
        # Menghapus emoticon dan tanda baca menggunakan "RegEx" dengan script di bawah
        string = re.sub(r'^a-zA-Z0-9', ' ', string)
        return string
```

- Melakukan Feature Extraction



```
In [9]: # Untuk melakukan Feature Extraction, kita menggunakan library "Sklearn atau scikit-Learn".
# Sklearn adalah library untuk melakukan task-task Machine Learning.
# "CountVectorizer" merupakan salah satu modul untuk melakukan "Bow"

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 selesai")

Feature Extraction selesai
```

- Menentukan Data Yang Akan Di Train Dan Validation

Split dataset menjadi 80% untuk train dan 20% untuk test.



```
In [24]: from sklearn.model_selection import train_test_split


classes = df.Label

In [26]: classes[0:5]

Out[26]: 0    positive
1    neutral
2    positive
3    positive
4    negative
Name: Label, dtype: object

In [13]: X_train, X_test, y_train, y_test = train_test_split(X, classes, test_size = 0.2)
```

- Melakukan Saving CountVectorizer



```
In [23]: import pickle  
         pickle.dump(count_vect, open("feature.pkl", "wb"))
```

- Melakukan Saving Model Neural Network




```
In [15]: pickle.dump(model, open("model.pkl", "wb"))
```


Hasil Model Dari LSTM

- Terdapat 11000 data label yang terdiri dari :

1.Positive : 6416
2.Negative : 3436
3.Netral : 1148




```
In [57]: df.label.value_counts()

Out[57]: positive      6416
         negative     3436
         neutral      1148
         Name: label, dtype: int64
```

- Melakukan Cleansing menggunakan functionl

```
def fun(txt) :
    txt = str(txt).lower()
    txt = re.sub("[,]", " ,", txt)
    txt = re.sub("[.]", " .", txt)
    txt = re.sub("[?]", " ?", txt)
    txt = re.sub("[!]", " !", txt)
    txt = re.sub("[\\"]", " \\", txt)
    txt = re.sub("[']", " '", txt)
    txt = re.sub("[\\n]", " ", txt)
    txt = re.split(" ", txt)
```




```
txt = " ".join(txt)
txt = re.sub(" ,", ",", txt)
txt = re.sub(" \.", ".", txt)
txt = re.sub(" ?", "?", txt)
txt = re.sub(" !", "!", txt)
txt = re.sub(" \\", "\\", txt)
```

```
return txt
```

```
df_baru['content'] = df_baru['content'].apply(lambda txt : fun(txt))
df_baru.head()
```

```
for word in txt :
    if word in alayOri :
        txt[txt.index(word)] = txt[txt.index(word)].replace(word, alayFix[alayOri.index(word)])
```


- Melakukan Feature Extraction

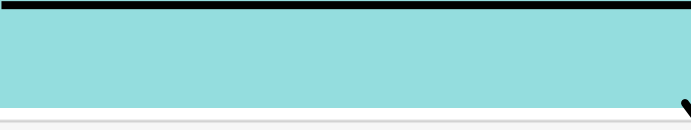


```
tokenizer = Tokenizer(num_words=5000, oov_token='x')
tokenizer.fit_on_texts(x)
```

```
sekuens_x = tokenizer.texts_to_sequences(x)
```

```
padded_x = pad_sequences(sekuens_x)
```

- Melakukan Train Dan Validation



```
In [53]: %%time
fit = model.fit(x_train,
                y_train,
                batch_size = 10,
                epochs=100,
                validation_data=(x_val, y_val),
                # validation_split=0.2,
                callbacks = [callbacks]
            )

Epoch 1/100
704/704 [=====] - 27s 33ms/step - loss: 5.7485 - categorical_accuracy: 0.6497 - val_loss: 3.2106 - val
_categorical_accuracy: 0.7909
Epoch 2/100
704/704 [=====] - 23s 32ms/step - loss: 2.0774 - categorical_accuracy: 0.7707 - val_loss: 1.3602 - val
_categorical_accuracy: 0.7960
Epoch 3/100
704/704 [=====] - 23s 32ms/step - loss: 1.0580 - categorical_accuracy: 0.8287 - val_loss: 0.8249 - val
_categorical_accuracy: 0.8330
Epoch 4/100
704/704 [=====] - 23s 32ms/step - loss: 0.6754 - categorical_accuracy: 0.8675 - val_loss: 0.6142 - val
_categorical_accuracy: 0.8568
CPU times: total: 5min 20s
Wall time: 1min 35s
```

- Model dan Hasil Validation Data

```
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(input_dim=5000, output_dim=64),

    tf.keras.layers.LSTM(64),

    tf.keras.layers.Dense(64, kernel_regularizer=L1L2(l1=0.01, l2=0.01), activation='relu'),
    tf.keras.layers.Dropout(0.1),
    tf.keras.layers.BatchNormalization(),

    tf.keras.layers.Dense(64, kernel_regularizer=L1L2(l1=0, l2=0.01), activation='relu'),
    tf.keras.layers.Dropout(0.1),
    tf.keras.layers.BatchNormalization(),

    tf.keras.layers.Dense(64, kernel_regularizer=L1L2(l1=0, l2=0.01), activation='relu'),
    tf.keras.layers.Dropout(0.1),
    tf.keras.layers.BatchNormalization(),

    tf.keras.layers.Dense(64, kernel_regularizer=L1L2(l1=0, l2=0.01), activation='relu'),
    tf.keras.layers.Dropout(0.1),
    tf.keras.layers.BatchNormalization(),

    tf.keras.layers.Dense(3, activation='softmax')
])
```

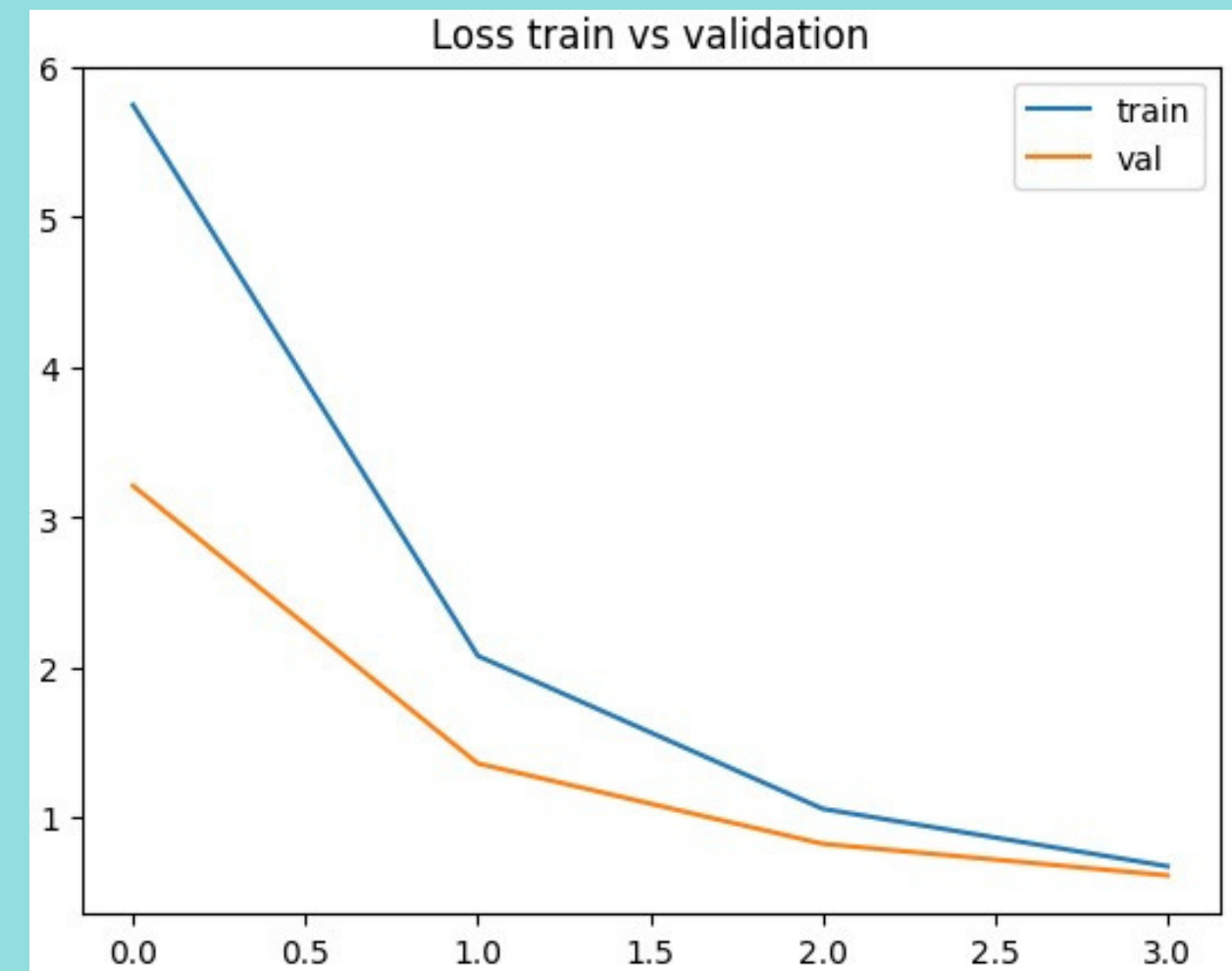
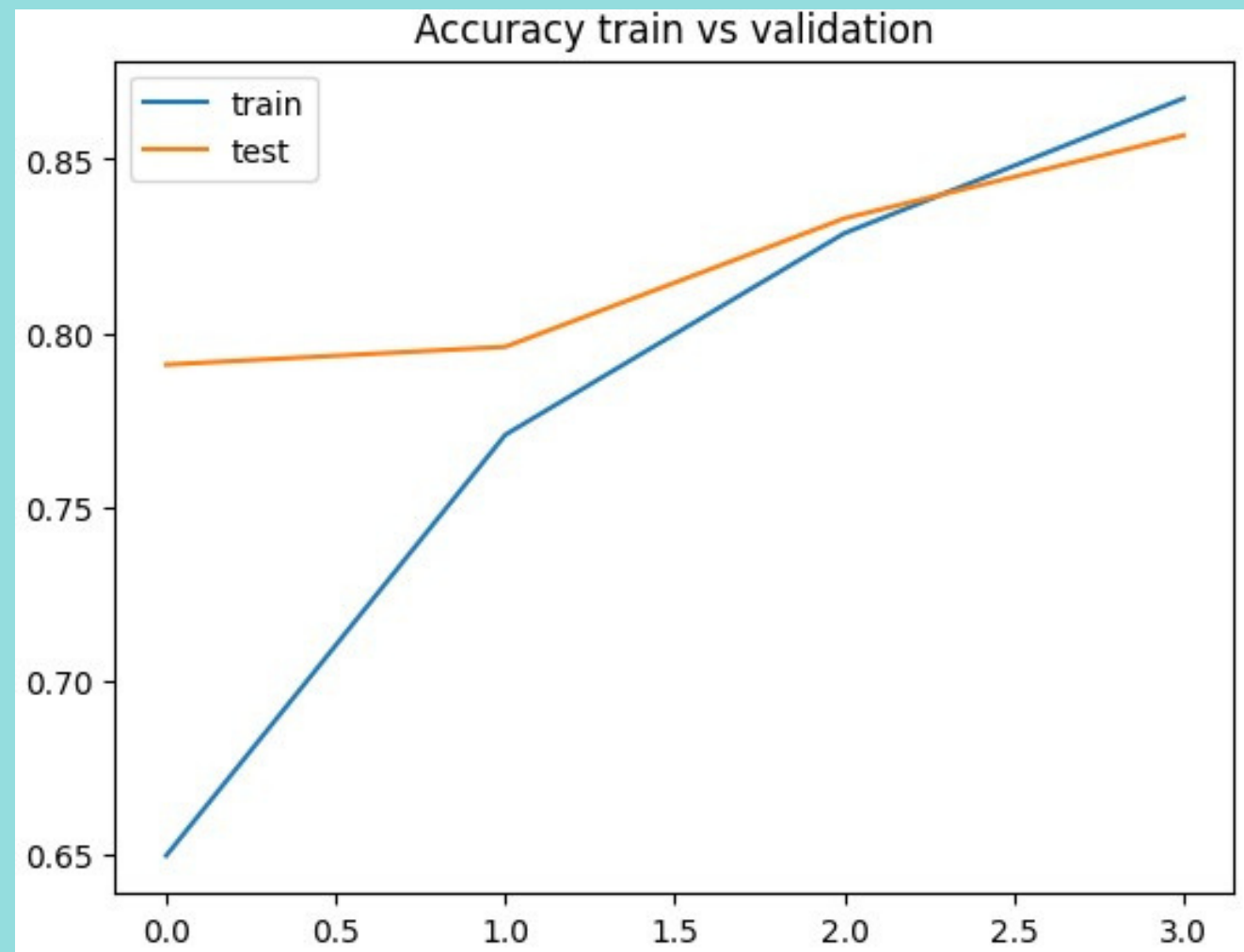
```
train_loss, train_acc = model.evaluate(x_train, y_train, steps=16)
val_loss, val_acc = model.evaluate(x_val, y_val, steps=16)
test_loss, test_acc = model.evaluate(x_test, y_test, steps=16)

print('\nTrain: %.3f, val: %.3f, test: %.3f,' % (train_acc, val_acc, test_acc))

16/16 [=====] - 2s 73ms/step - loss: 0.4706 - categorical_accuracy: 0.9226
16/16 [=====] - 0s 28ms/step - loss: 0.6142 - categorical_accuracy: 0.8568
16/16 [=====] - 1s 31ms/step - loss: 0.6539 - categorical_accuracy: 0.8268

Train: 0.923, val: 0.857, test: 0.827,
```

- Hasil Akurasi dan Loss



- Melakukan Saving Model dan Tokenizer



```
In [21]: filename = 'lstm.h5'  
         model.save(filename)
```

```
In [22]: tokenizer_json = tokenizer.to_json()  
         with io.open('tokenizer.json', 'w', encoding='utf-8') as f:  
             f.write(json.dumps(tokenizer_json, ensure_ascii=False))
```

Conclusion

Machine Learning mampu mengklasifikasikan sentimen dari ribuan data tweet dalam hitungan menit. Meskipun akurasi hanya 84% - 87%, Machine Learning mampu melakukan sentimen dalam waktu yang cepat.

