TWEET SENTIMENT ANALYSIS USING MACHINE LEARNING (NEURAL NETWORK & LSTM)

Faza Hanifandra

https://www.linkedin.com/in/faza-hanifandra-b843a4134/

https://github.com/fazahanifandra

outline

background

A Twitter sentiment analysis determines negative, positive, or neutral emotions within the text of a tweet using Neural Network and LSTM deep learning models.

Sentiment analysis or opinion mining refers to identifying as well as classifying the sentiments that are expressed in the text source. Tweets are often useful in generating a vast amount of sentiment data upon analysis. These data are useful in understanding the opinion of people on social media for a variety of topics.

objectives

- Gain insights into public sentiment towards specific words, topics, products, events, or brands. This involves statistical measurements to understand the scale and frequency of the specific word(s).
- Identify prevalent sentiments (positive, negative, neutral) associated with particular keywords or hashtags.

data

Data

“TWEET SENTIMENT ANALYSIS”

- Source: https://drive.google.com/file/d/1RCHGfn9JJyyReAh8PIIoF8Ch0H3miP0u/view?usp=drive_link

DETAILS

- 11,000 rows of order data where has no missing value in the database
- 3 features / columns

	text	sentiment
0	warung ini dimiliki oleh pengusaha pabrik tahu...	positive
1	mohon ulama lurus dan k212 mmbri hujjah partai...	neutral
2	lokasi strategis di jalan sumatera bandung . t...	positive
3	betapa bahagia nya diri ini saat unboxing pake...	positive
4	duh . jadi mahasiswa jangan sombong dong . kas...	negative

Methodology

PREPARE DATASETS
- Text Normalization / Cleansing
- Feature Extraction using Tokenizer

TRAIN THE MODEL
- Split the data (Train & Test)
- Training using MLPClassifier (NN)
- Training using Tensorflow (LSTM)

CROSS VALIDATION
- Using SKLearn
- Define the training splits
- Calculate the accuracy
- Visualize the training model

CREATE API ENDPOINT
Using Flask and Swagger
- Sentiment Analysis using NN from Text
- Sentiment Analysis using NN from File
- Sentiment Analysis using LSTM from Text
- Sentiment Analysis using LSTM from File

ANALYZE THE RESULTS
- Input file “data.csv”

Prepare datasets

TEXT NORMALIZATION

- Cleansing the data to change all sentences to small letters and erase the symbols & emoticons

SORT AND ADD TO LIST

- Adding to a lists to provide a convenient way to gather and store data from various sources, such as user input, file reads, or API responses. By adding items to a list, I can progressively collect and aggregate data for further analysis or processing.

Text Normalization / Cleansing

In [6]:
```python
import re

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
```

In [7]:
```python
df['text_clean'] = df.text.apply(cleansing)
```

In [8]:
```python
df.head()
```

Out[8]:

	text	sentiment	text_clean
0	warung ini dimiliki oleh pengusaha pabrik tahu...	positive	warung ini dimiliki oleh pengusaha pabrik tahu...
1	mohon ulama lurus dan k212 mmbri hujjah partai...	neutral	mohon ulama lurus dan k212 mmbri hujjah partai...
2	lokasi strategis di jalan sumatera bandung . t...	positive	lokasi strategis di jalan sumatera bandung t...
3	betapa bahagia nya diri ini saat unboxing pake...	positive	betapa bahagia nya diri ini saat unboxing pake...
4	duh . jadi mahasiswa jangan sombong dong . kas...	negative	duh jadi mahasiswa jangan sombong dong kas...

Sort the data and lable based on sentiments

```python
neg = df.loc[df['sentiment'] == 'negative'].text_clean.tolist()
neu = df.loc[df['sentiment'] == 'neutral'].text_clean.tolist()
pos = df.loc[df['sentiment'] == 'positive'].text_clean.tolist()

neg_sentiment = df.loc[df['sentiment'] == 'negative'].sentiment.tolist()
neu_sentiment = df.loc[df['sentiment'] == 'neutral'].sentiment.tolist()
pos_sentiment = df.loc[df['sentiment'] == 'positive'].sentiment.tolist()
```

```python
total_data = pos + neu + neg
sentiments = pos_sentiment + neu_sentiment + neg_sentiment

print("Pos: %s, Neu: %s, Neg: %s" % (len(pos), len(neu), len(neg)))
print("Total data: %s" % len(total_data))
```

```
Pos: 6416, Neu: 1148, Neg: 3436
Total data: 11000
```

```python
data_preprocessed = df['text_clean'].tolist()
```

Prepare datasets

FEATURE EXTRACTION

- Import the count vectorizer object that contains the vectorization process of the entire training data
- So that, before the prediction is performed on the new data later, the new text data can be converted into a vector/vectorization

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

```python
count_vect = CountVectorizer()
count_vect.fit(data_preprocessed)

x = count_vect.transform(data_preprocessed)
```

```python
# Import the countvectorizer object that contains the vectori
# So that, before the prediction is performed on the new data

import pickle

pickle.dump(count_vect, open("feature.p", "wb"))
```

```python
# how to open feature.p (pickle dump result)

file = open("feature.p",'rb')
count_vect_import = pickle.load(file)
file.close()
```

OUTPUT

- Neural Network -> using CountVectorizer to extract feature.p
- LSTM -> using Tokenizer to extract x_pad_sequences.pickle

```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 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 created!")
```

```
WARNING:tensorflow:From C:\Users\Faza\Documents\Binar\venv_test\new_plat_
me tf.losses.sparse_softmax_cross_entropy is deprecated. Please use tf.co

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

```python
Y = pd.get_dummies(sentiments)
Y = Y.values

with open('y_labels.pickle', 'wb') as handle:
    pickle.dump(Y, handle, protocol=pickle.HIGHEST_PROTOCOL)
    print("y_labels.pickle has created!")
```

```
y_labels.pickle has created!
```

Train the model

- Precision is the proportion of positive predictions that are actually correct,
- Recall is the proportion of actual positives that are correctly predicted
- F1-score is the harmonic mean of precision and recall.
- The accuracy of the model is 0.87, which means that **it correctly classified 87% of the documents.**
- The macro average of precision, recall, and F1-score is 0.86, which means that the model performed well on average across all three sentiment categories.
- The weighted average of precision, recall, and F1-score is 0.87, which is the same as the overall accuracy because the positive sentiment category has the largest number of documents.

SPLIT THE DATASETS

- Split the datasets into train and test

Prepare train & test datasets/Splitting Dataset

Before modeling, we need to split the existing data into 'data train' and 'data test'

```
In [16]: from sklearn.model_selection import train_test_split

         classes = df['sentiment']

In [31]: classes

Out[31]: 0        positive
         1         neutral
         2        positive
         3        positive
         4        negative
                    ...
         10995    positive
         10996    positive
         10997     neutral
         10998    negative
         10999    positive
         Name: sentiment, Length: 11000, dtype: object
```

MODELING USING MLP CLASSIFIER

- Modeling using algorithm machine learning MLPClassifier (Basic Neural Network)
- The result shows that the F1 Score for Negative is in 0.81, 0.74 for Neutral, and 0.9 for Positive.

```
In [21]: # dump model into the pickle file

         pickle.dump(model, open("model.p","wb"))

In [22]: from sklearn.metrics import classification_report

         y_pred = model.predict(x_test)

In [23]: print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
negative	0.80	0.82	0.81	689
neutral	0.80	0.69	0.74	230
positive	0.89	0.90	0.90	1281
accuracy			0.85	2200
macro avg	0.83	0.81	0.82	2200
weighted avg	0.85	0.85	0.85	2200

MODELING USING LSTM

- Modeling using algorithm deep learning Tensorflow
- The result shows that the F1 Score for Negative is in 0.82, 0.79 for Neutral, and 0.91 for Positive

Confussion Matrix, Accuracy, F1, Recall, Precision

```
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)
```

```
69/69 [==============================] - 1s 13ms/step
Testing selesai
```

	precision	recall	f1-score	support
0	0.86	0.78	0.82	685
1	0.84	0.74	0.79	233
2	0.88	0.94	0.91	1282
accuracy			0.87	2200
macro avg	0.86	0.82	0.84	2200
weighted avg	0.87	0.87	0.87	2200

Cross validation

- To estimate the generalizability of a machine learning model.
- It involves dividing the available data into multiple subsets, training the model on a subset of the data, and evaluating its performance on the remaining subset.
- This process is repeated multiple times, using different subsets for training and evaluation each time.
- The average performance across all folds is used to estimate the model's generalization performance.

BOTH CROSS VALIDATION IS ACCURATE, BUT LSTM IS MORE ACCURATE

NEURAL NETWORK RESULT

```
Training ke- 5
              precision    recall  f1-score   support

    negative       0.76      0.82      0.79       670
     neutral       0.80      0.66      0.72       245
    positive       0.90      0.89      0.89      1285

    accuracy                           0.84      2200
   macro avg       0.82      0.79      0.80      2200
weighted avg       0.84      0.84      0.84      2200

============================================

Rata-rata Accuracy:  0.8437272727272728
```

```
1 The validation training shows that the model have an average accuracy of 84.3% which is indicates a good model
```

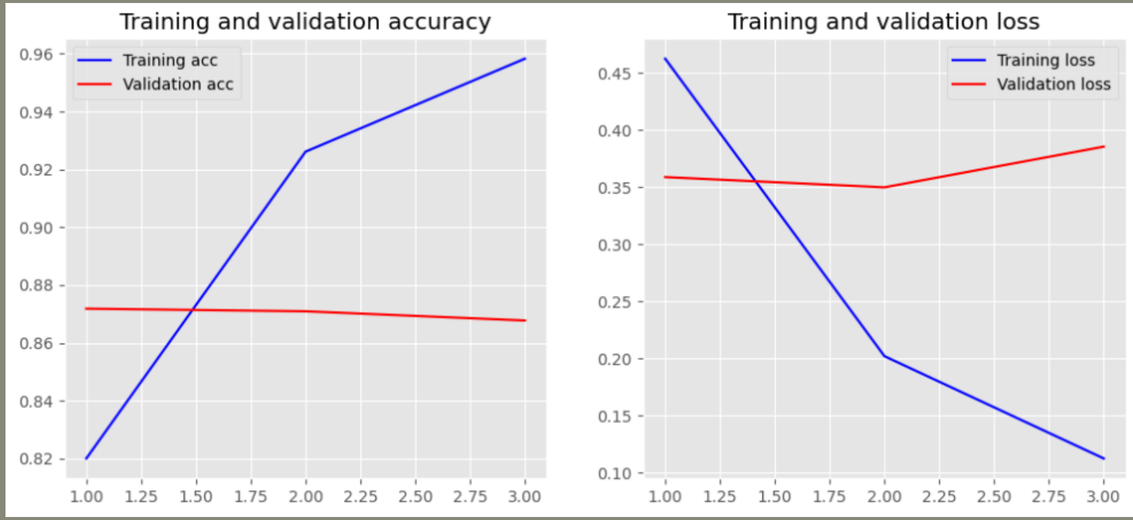
LSTM RESULT

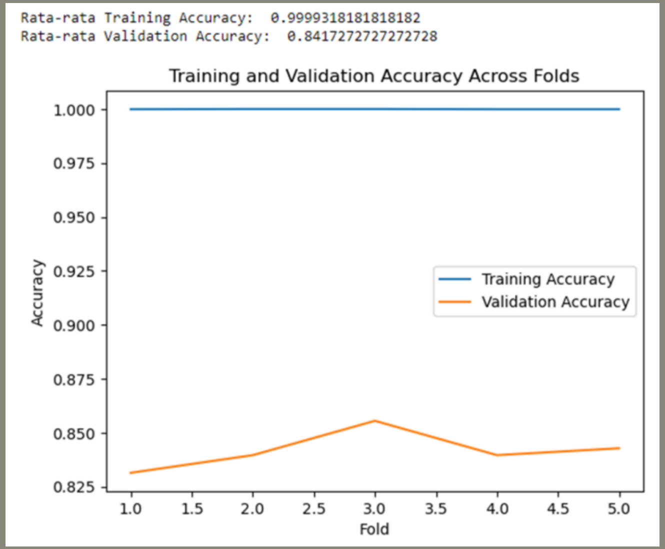
```
69/69 [==============================] - 1s 11ms/step
Training ke- 5
              precision    recall  f1-score   support

           0       0.81      0.83      0.82       685
           1       0.81      0.80      0.80       233
           2       0.91      0.90      0.91      1282

    accuracy                           0.87      2200
   macro avg       0.84      0.84      0.84      2200
weighted avg       0.87      0.87      0.87      2200

==================================================

Rata-rata Accuracy:  0.8748181818181819
```

The validation training shows that the model have an average accuracy of 87.5% which is indicates a good model

Visualize

- To determine if my training data is underfitting or underfitting based on the cross validation results





- The LSTM training model indicates that this training data can be categorized as **underfitting**.

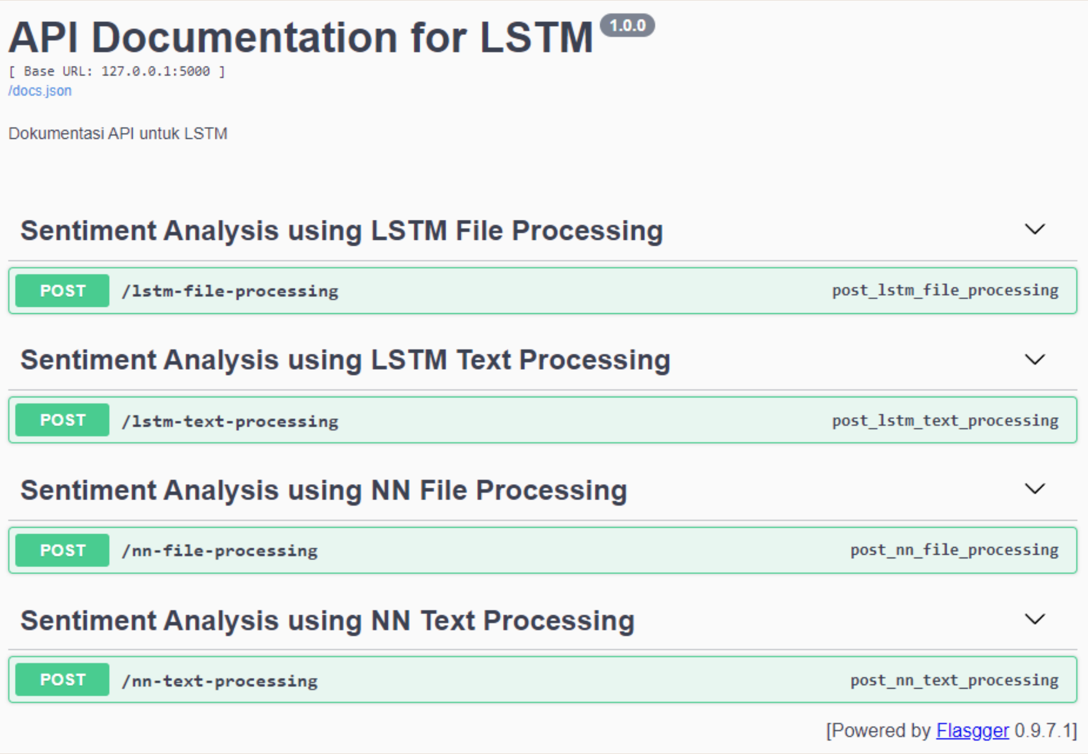
- The Neural Network training model indicates that this training data can be categorized as **underfitting**.

This can be caused by several points:

a. The data is too small or does not adequately represent the diversity of the real-world data that the model will be applied to.

b. The training process stopped prematurely, the model may not have converged to the optimal solution

API integration



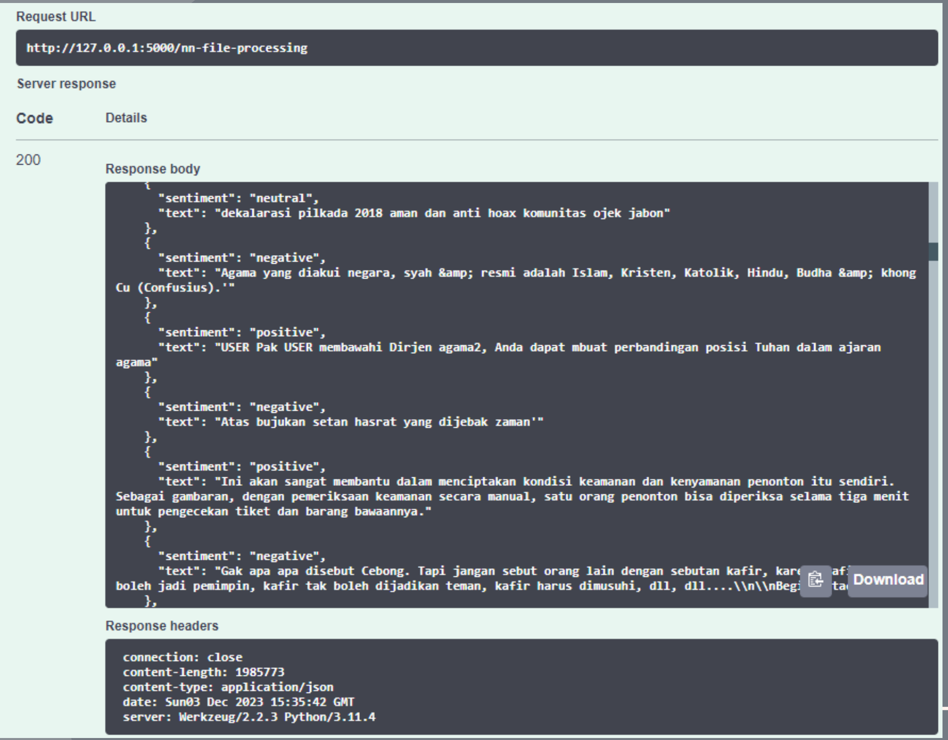
- To facilitate seamless model testing, we'll integrate the model into an API endpoint using the Flask framework and Swagger documentation.
- The 'POST' method will be employed to transmit data to the server.

Results

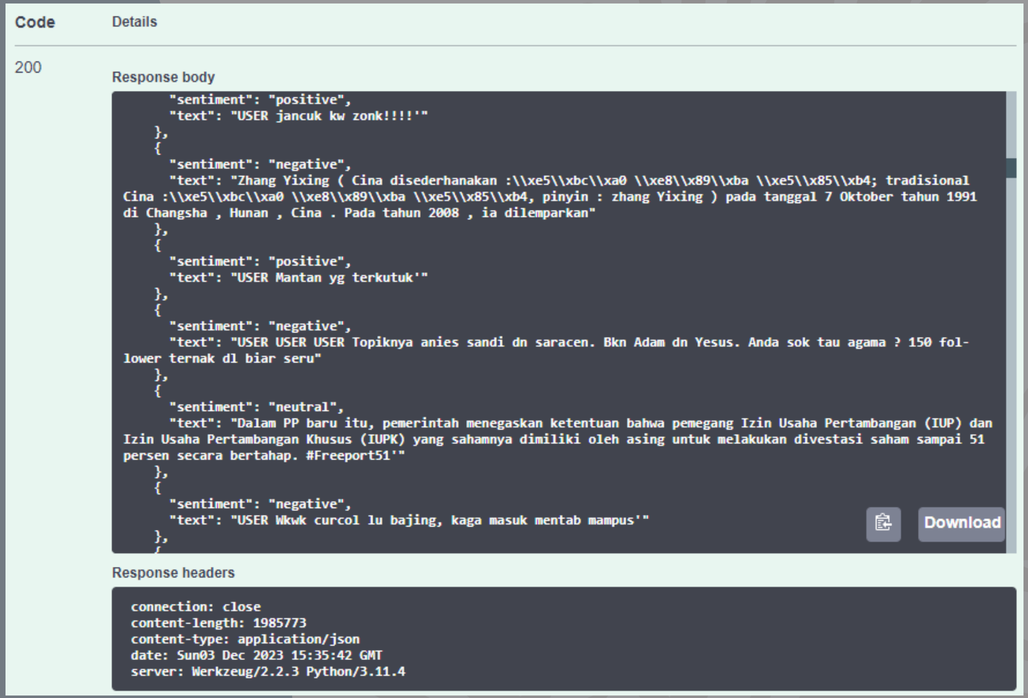
- We input the “data.csv” to perform twitter text sentiment analysis using our Neural Network and LSTM models in the API

BOTH METHOD ARE ACCURATE, BUT LSTM TAKES LONGER TO PROCESS

NEURAL NETWORK SENTIMENT ANALYSIS



LSTM SENTIMENT ANALYSIS



summary

- Both Neural Network and LSTM methods can be used to conduct text sentiment analysis
- LSTM is more accurate to predict sentiment analysis than traditional Neural Network, with the cost of longer processing
  - NN training accuracy : 85%, cross validation accuracy : 84%
  - LSTM training accuracy : 87%, cross validation accuracy : 87.5%
  - LSTM is specifically designed to handle sequential data, such as time series data or natural language. They are more accurate than traditional RNNs because they are able to better capture long-term dependencies in the data.
- Both cross validation data is accurate, but still categorized as underfitting due to the training stopped prematurely
- For improvement, writer shall explore LSTM method further by implementing more complex model, increase training epochs and adjust regularization parameters

thank you

Faza Hanifandra

https://www.linkedin.com/in/faza-hanifandra-b843a4134/

https://github.com/fazahanifandra