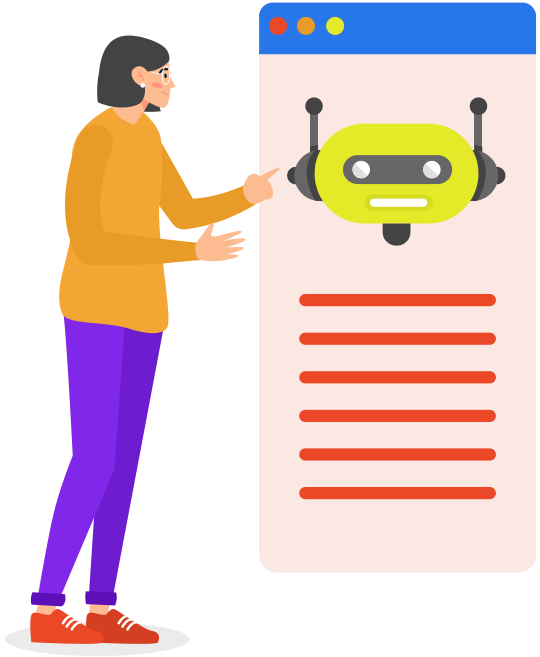
BINAR ACADEMY

ANALISIS SENTIMEN MENGGUNAKAN MODEL NEURAL NETWORK DAN LSTM

Oleh:
Andi, Atikah, dan Adi

BINAR ACADEMY

PENDAHULUAN



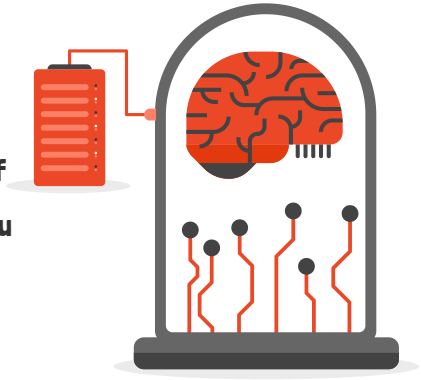
Dalam era informasi digital saat ini, di mana data teks teridentifikasi dalam volume yang besar seperti email, comment social media, review (google, transkrip obrolan, dan ecommers) analisis sentimen menjadi sangat relevan untuk memahami pandangan, respons, dan perasaan pengguna terhadap berbagai topik, produk, layanan, atau kejadian.

Apa yang dimaksud dengan Analisis Sentimen?

Analisis sentimen adalah proses menganalisis teks digital untuk menentukan apakah nada emosional pesan tersebut positif, negatif, atau netral.[1]

Manfaat Sentimen Analisis:

01 Memberikan wawasan yang objektif
02 Menemukan strategi marketing baru
03 Melacak performa kampanye
04 Meningkatkan persepsi media



METODE PENELITIAN

Metode yang digunakan pada penelitian/challenge kali ini adalah sebagai berikut :

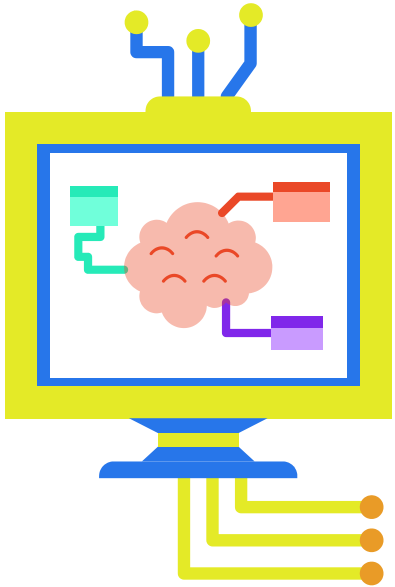
1. Cleansing data pada dataset Analisis Sentimen menggunakan Pandas dan RegEx

2. Feature extraction pada dataset Analisis Sentimen menggunakan Sklearn

3. Training
menggunakan 2 metode:

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

4. Kalkulasi analisis sentimen dari metode Neural Network



5. Evaluasi pada model Neural Network dan LSTM yang sudah di-training dengan Sklearn

6. Visualisasi evaluasi pada model Neural Network dan LSTM yang sudah di-training dengan menggunakan matplotlib dan seaborn

7. Membangun API untuk prediksi sentimen menggunakan model Neural Network dan LSTM dengan menggunakan Flask dan Swagger UI

MODEL NEURAL NETWORK

01 Data Cleansing

Menggunakan library Pandas dan Regex unutk melakukan clensing data

02 Feature Extraction

Menggunakan library Sklearn dengan metode BoW (Bag of Words

03 Training

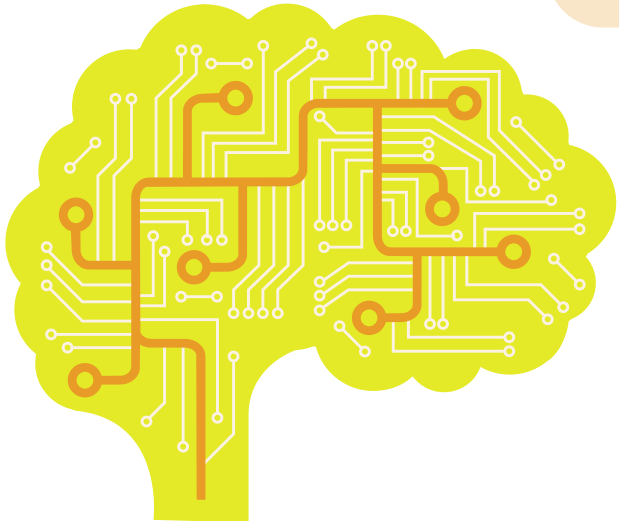
Menggunakan tool Sklearn untuk melakukan training dengan modul MLPClassifier

04 Evaluasi

Evaluasi menggunakan modul Classification_ Report kemudian melaukan Cross Validation

05 Predict

Melakukan tes apakah model yang kita buat bisa "berjalan" dengan baik atau tidak.

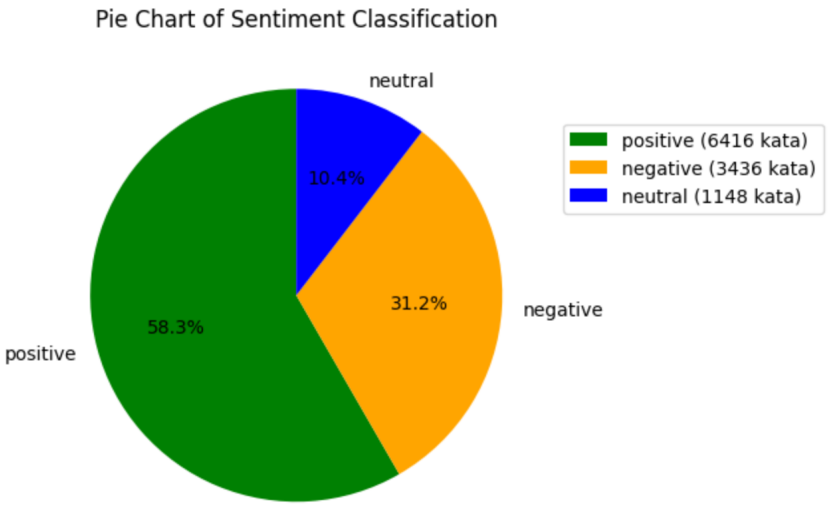


MODEL NEURAL NETWORK

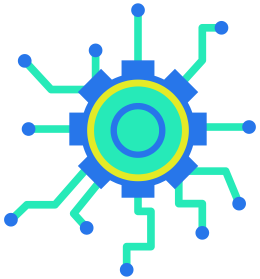
MACHINE LEARNING

Data Cleansing

Mencari value dari data set untuk mengetahui komposisi data



MODEL NEURAL NETWORK



MACHINE LEARNING

Data Cleansing

Menggunakan library Pandas dan Regex

Kemudian mebuat fungsi untuk raw text sehingga menghasikan text_clean

Text Cleansing

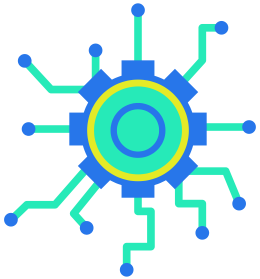
```
def cleansing(sent):
    string = sent.lower()
    string = re.sub(r'[^a-zA-Z0-9]',' ',string)
    return string
```

```
df['text_clean'] = df['text'].apply(cleansing)
```

```
df.head()
```

	text	sentiment	ln_text	text_clean
0	warung ini dimiliki oleh pengusaha pabrik tahu...	positive	404	warung ini dimiliki oleh pengusaha pabrik tahu...
1	mohon ulama lurus dan k212 mmbri hujjah partai...	neutral	102	mohon ulama lurus dan k212 mmbri hujjah partai...
2	lokasi strategis di jalan sumatera bandung . t...	positive	184	lokasi strategis di jalan sumatera bandung t...
3	betapa bahagia nya diri ini saat unboxing pake...	positive	93	betapa bahagia nya diri ini saat unboxing pake...
4	duh . jadi mahasiswa jangan sombong dong . kas...	negative	214	duh jadi mahasiswa jangan sombong dong kas...

MODEL NEURAL NETWORK



MACHINE LEARNING

Feature Extraction

Menggunakan library Sklearn dengan metode BoW (Bag of Words)

Feature Extraction Bag of Word

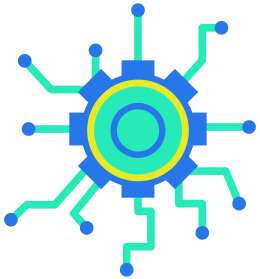
```
data_preprocessed = df['text_clean'].tolist()
```

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

x = count_vect.transform(data_preprocessed)
```

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

Type Markdown and LaTeX: α^2

MODEL NEURAL NETWORK



MACHINE LEARNING

Data Training

Menggunakan tool Sklearn untuk melakukan training dengan modul MLPClassifier

Training(Splitting Dataset)

```
#berikut ini adalah split existing data menjadi 80% data trainning dan 20% data test
```

```
classes = df['sentiment']
```

```
x_train, x_test, y_train, y_test = train_test_split(x, classes, test_size = 0.2)
```

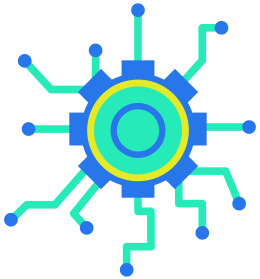
```
print(f"x_train size: {x_train.shape[0]}")
print(f"y_train size: {y_train.shape[0]}")
print(f"x_test size: {x_test.shape[0]}")
print(f"y_test size: {y_test.shape[0]}")
```

```
x_train size: 8800
y_train size: 8800
x_test size: 2200
y_test s  : 2200
```

```
model    assifier()
model.fit(x_train, y_train)
```

```
▾ MLPClassifier
MLPClassifier()
```

MODEL NEURAL NETWORK



MACHINE LEARNING

Evaluasi

Evaluasi menggunakan modul Classification_ Report kemudian melaukan Cross Validation

Evaluasi Model

```
y_pred = model.predict(x_test)

print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

    negative       0.77      0.78      0.78       682
     neutral       0.74      0.68      0.71       218
    positive       0.89      0.90      0.89      1300

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

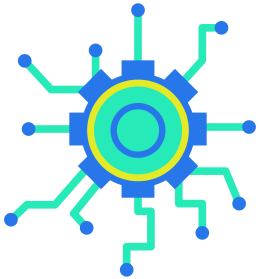
Cross Validation

```
#KF ini adalah unutk melihat konsistensi nilai presisi, recall, dan f1

kf = KFold(n_splits=5,random_state=42,shuffle=True)

accuracies = []

y = classes

for iteration, data in enumerate(kf.split(x), start=1):

    data_train   = x[data[0]]
    target_train = y[data[0]]

    data_test    = x[data[1]]
    target_test  = y[data[1]]

    clf = MLPClassifier()
    clf.fit(data_train,target_train)

    preds = clf.predict(data_test)

    # for the current fold only
    accuracy = accuracy_score(target_test,preds)

    print("Training ke-", iteration)
    print(classification_report(target_test,preds))
    print("==================================================")

    accuracies.append(accuracy)

# this is the average accuracy over all folds
average_accuracy = np.mean(accuracies)

print()
print()
print()
print("Rata-rata Accuracy: ", average_accuracy)
```

```
Rata-rata Accuracy:  0.8447272727272728
```

Type Markdown and LaTeX: α^2

MODEL NEURAL NETWORK



MACHINE LEARNING

Evaluasi

Evaluasi menggunakan modul Classification_ Report kemudian melaukan Cross Validation

Evaluasi Model

```
y_pred = model.predict(x_test)

print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
negative	0.77	0.78	0.78	682
neutral	0.74	0.68	0.71	218
positive	0.89	0.90	0.89	1300
accuracy			0.84	2200
macro avg	0.80	0.79	0.79	2200
weighted avg	0.84	0.84	0.84	2200

Training ke- 1

	precision	recall	f1-score	support
negative	0.78	0.77	0.77	680
neutral	0.75	0.64	0.69	239
positive	0.87	0.89	0.88	1281
accuracy			0.83	2200
macro avg	0.80	0.77	0.78	2200
weighted avg	0.83	0.83	0.83	2200

Training ke- 2

	precision	recall	f1-score	support
negative	0.81	0.77	0.79	706
neutral	0.74	0.71	0.73	220
positive	0.88	0.91	0.90	1274
accuracy			0.85	2200
macro avg	0.81	0.80	0.80	2200
weighted avg	0.84	0.85	0.85	2200

Training ke- 3

	precision	recall	f1-score	support
negative	0.80	0.80	0.80	682
neutral	0.86	0.73	0.79	215
positive	0.89	0.91	0.90	1303
accuracy			0.86	2200
macro avg	0.85	0.82	0.83	2200
weighted avg	0.86	0.86	0.86	2200

Training ke- 4

	precision	recall	f1-score	support
negative	0.78	0.81	0.79	698
neutral	0.79	0.65	0.71	229
positive	0.89	0.90	0.89	1273
accuracy			0.84	2200
macro avg	0.82	0.79	0.80	2200
weighted avg	0.84	0.84	0.84	2200

Training ke- 5

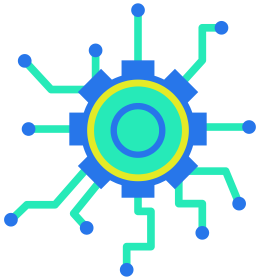
	precision	recall	f1-score	support
negative	0.77	0.81	0.79	670
neutral	0.79	0.67	0.72	245
positive	0.90	0.90	0.90	1285
accuracy			0.85	2200
macro avg	0.82	0.79	0.80	2200
weighted avg	0.85	0.85	0.85	2200

Rata-rata Accuracy: 0.8447272727272728

– Performa Keseluruhan:
Model menunjukkan kinerja yang baik dengan rata-rata akurasi sebesar 84.47%. Hal ini menunjukkan bahwa model secara umum mampu melakukan klasifikasi sentimen dengan tepat pada dataset yang digunakan.

– Konsistensi Performa:
Hasil evaluasi pada masing-masing fold menunjukkan konsistensi yang relatif baik dalam nilai precision, recall, dan F1-score. Variasi antara fold-fold tidak terlalu besar, yang menandakan bahwa model memiliki kemampuan yang stabil dalam melakukan klasifikasi sentimen.

MODEL NEURAL NETWORK



MACHINE LEARNING

Predict

Melakukan tes apakah model yang kita buat bisa "berjalan" dengan baik atau tidak. Dan hasilnya cukup akurat

Predict

```
# Melakukan prediksi dengan model yang telah dibuat
original_text = '''
saya mau makan
'''

text = count_vect.transform([cleansing(original_text)])

result = model.predict(text)[0]
print("Sentiment:")
print()
print(result)
```

```
Sentiment:

neutral
```

MODEL LSTM

01 Data Cleansing

Menggunakan library Pandas dan Regex unutk melakukan clensing data

02 Feature Extraction

Menggunakan Tokenizer dan pad_sequences

03 Training

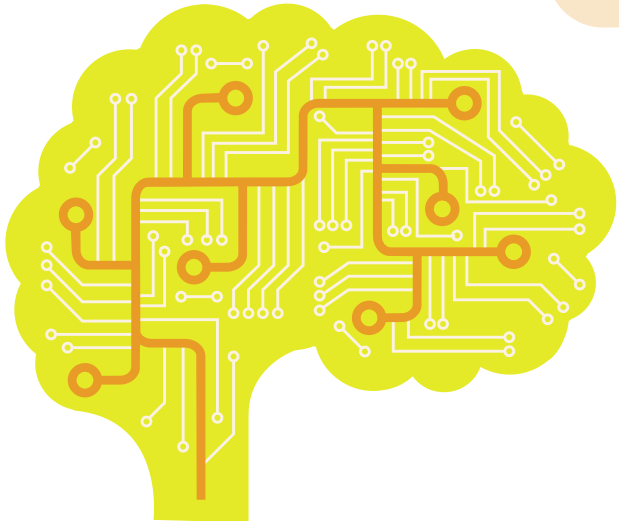
Menggunakan tool Tensorflow untuk melakukan training

04 Evaluasi

Evaluasi menggunakan confusion matrix, accuracy, F1, recall, dan precision. Kemudian mekaukan cross validation

05 Predict

Melakukan tes apakah model yang kita buat bisa "berjalan" dengan baik atau tidak.

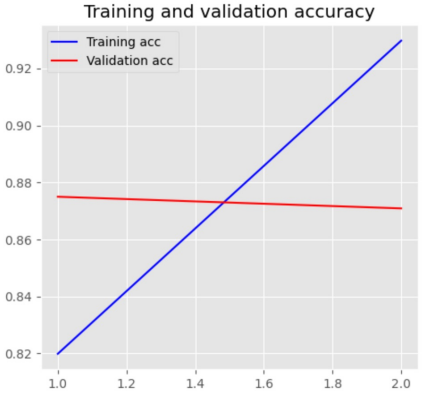


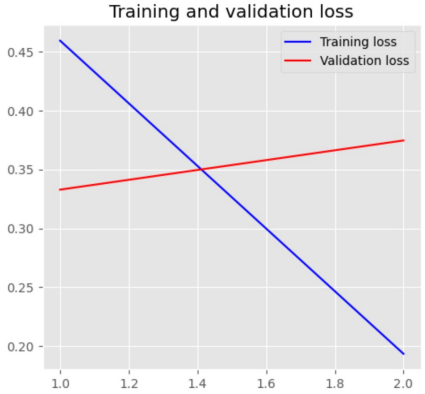
MODEL LSTM

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

           0       0.84      0.81      0.82       685
           1       0.81      0.77      0.79       233
           2       0.89      0.92      0.91      1282

    accuracy                           0.87      2200
   macro avg       0.85      0.83      0.84      2200
weighted avg       0.87      0.87      0.87      2200

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


Rata-rata Accuracy:  0.8714545454545455
```

```
1/1 [==============================] - 0s 395ms/step
Text:
orang itu seperti bajingan tolol

Sentiment:  negative
```

– Performa Keseluruhan:
Hasil nilai rat-rata accuracy pada angka 0.87. Hal ini menunjukkan model cukup stabil di angka 0.87






```
None
Epoch 1/10
880/880 [==============================]
Epoch 2/10
880/880 [==============================]
Epoch 2: early stopping
```

Dari hasil model training bisa dilihat bahwa model yang dikembangkan tergolong underfitting, karena:

•Data yang di-training terlalu sedikit
•Proses training berhenti terlalu cepat.

MEMBANGUN API

Membuat API prediksi sentimen menggunakan NN dan LSTM

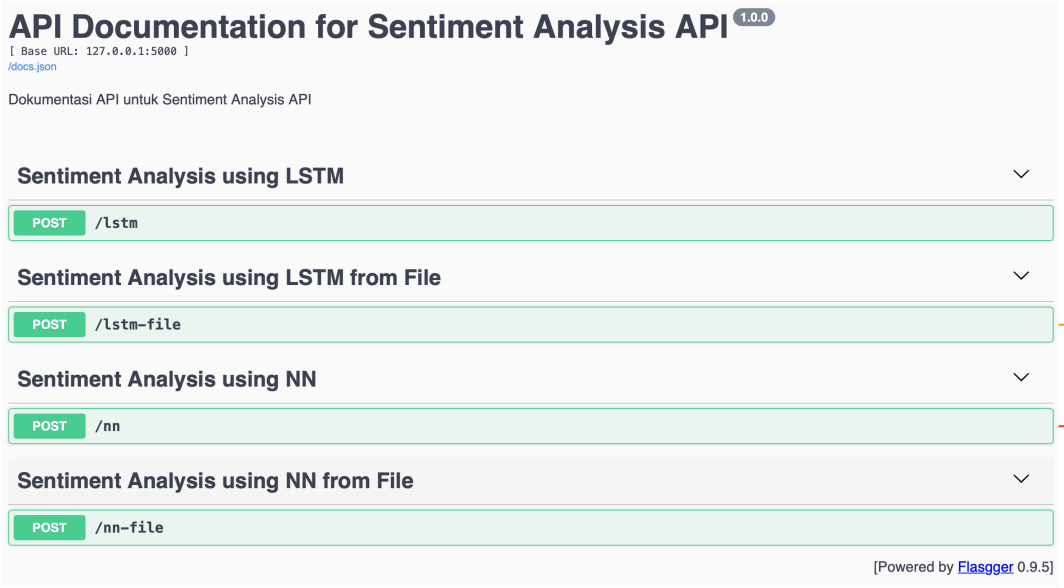
01 **Neural Network**

Membuat dua endpoint untuk model NN berupa text dan file

02 **LSTM**

Membuat dua endpoint untuk model LSTM berupa text dan file

MEMBANGUN API



End point LSTM untuk masukan text

End point LSTM untuk masukan file

End point NN untuk masukan text

End point NN untuk masukan file

KESIMPULAN

Pada penelitian ini, telah dilakukan pembuatan model neural network menggunakan tensorflow dan sklearn. Pada saat dilakukan pembuatan model menggunakan neural network pada pustaka tensorflow, didapatkan accuracy sebesar <0.8, dan dengan sklearn didapatkan accuracy sebesar 0.77.

Selain itu, API untuk melakukan klasifikasi telah dibuat meggunakan swager ui.