

The background is a dark navy blue. On the left, there is a large, semi-transparent circular graphic containing a detailed image of a printed circuit board (PCB) with various electronic components. Overlaid on the top left of this circle are two overlapping triangles: a blue one in the foreground and a light green one behind it. In the top right corner, there is a faint, grey, 3D-rendered pattern of interlocking cubes or a circuit board layout. The main title is centered in the upper right area in a large, white, sans-serif font.

BINAR PLATINUM CHALLENGE

Kelompok 1 - DSC 10

Anggota

Firdaus Romandhanu

Januardo Panggabean

Muhammad Syayiq Alqadri





Pendahuluan

Hasil penelitian dalam laporan ini menempatkan netizen Indonesia pada urutan 5 terbawah untuk tingkat kesopanan.

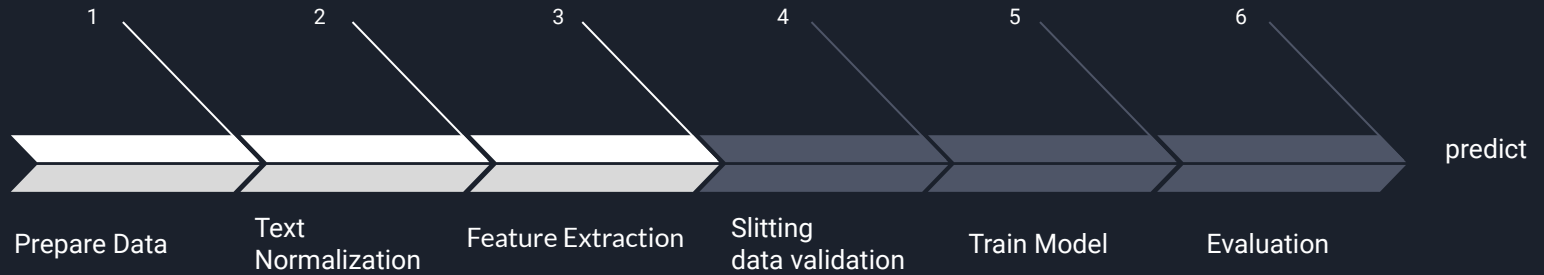
Hal ini terjadi karena pertumbuhan dunia digital yang sangat pesat akhir-akhir ini. Membuat laju arus informasi begitu terbuka lebar. Sehingga dapat menjangkau sampai ke pelosok. Namun pertumbuhan ini tidak di ikuti dengan kesadaran akan etika dalam berinteraksi dalam dunia digital.


Oleh karena itu penelitian ini bertujuan untuk menganalisis kata yang sering digunakan dalam komentar sentimen negative atau hate speech, dan kata yang sering digunakan dalam komentar tweeter. Harapannya dari hasil analisis yang dilakukan menjadi bahan pertimbangan berbagai pihak kedepannya.



METODE

Alur






Metode Umum - Dataset Cleansing & Preparation

```
def lowercase(text):  
    return text.lower()  
  
def remove_unnecessary_char(text):  
    text = re.sub('\n', '', text)  
    text = re.sub('rt', '', text)  
    text = re.sub('user', '', text)  
    text = re.sub('((www\.[^\s]+)|(https?://[^\s]+)|(http?://[^\s]+))', '', text)  
    text = re.sub(' +', ' ', text)  
    return text  
  
def remove_nonaplphanumeric(text):  
    text = re.sub('[^0-9a-zA-Z]+', ' ', text)  
    text = re.sub(' +', ' ', text)  
    return text  
  
def textcleansing(text):  
    lowertext = lowercase(text)  
    lowercharfix = remove_unnecessary_char(lowertext)  
    lowercharalpha = remove_nonaplphanumeric(lowercharfix)  
    return lowercharalpha  
  
textdatabase_df['text-cleansed'] = textdatabase_df['text'].apply(lambda x: textcleansing(x))
```

Dalam persiapan dataset yang akan dijadikan sebagai bahan pembelajaran model baik dengan metode NLP atau menggunakan LSTM, secara umum data teks yang didapat harus melewati beberapa tahap pembersihan seperti yang disajikan di samping.



Metode Umum - Dataset Cleansing & Preparation

```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
stopwordsindo = nltk.corpus.stopwords.words(['indonesian'])
def stopwordscleanse(text):
    tokenizedwords= word_tokenize(text)
    filteredtokens = [tokenwords for tokenwords in tokenizedwords if tokenwords.lower() not in stopwordsindo]
    filteredtext = ' '.join(filteredtokens)
    return filteredtext

textdatabase_df['text-stopwd-clean'] = textdatabase_df['text-cleansed'].apply(lambda x:stopwordscleanse(x))
processed_data = pd.DataFrame({
    'clean_text' : textdatabase_df['text-cleansed'],
    'stopwclean_text' : textdatabase_df['text-stopwd-clean'],
    'labels' : textdatabase_df['sentimentlabel01']
})

processed_data.to_csv("/Users/januardopanggabean/VSCE Platinum Challenge/data/processed_data.csv", index=False)
```

Setelah melakukan cleansing terhadap dataset, hasil yang sudah dianggap bersih lalu dimasukkan ke dalam dataframe baru sehingga dapat siap diproses dengan berbagai metode yang akan digunakan selanjutnya



NLP (Natural Language Processing)

```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split, KFold
from sklearn.naive_bayes import MultinomialNB
from sklearn.neural_network import MLPClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.metrics import classification_report, accuracy_score
import numpy as np
from PlatinumGroup1BaseData import fullcleanse, textcleansing
import pickle
```

Penggunaan NLP untuk memodelisasi sentiment analisis yang dimulai dengan melakukan import library pada model NLP ini.



NLP (Natural Language Processing)

```
processed_data = pd.read_csv('H:/Challange Platinum/Challange/data/Processed_data.csv')
train_data, test_data = train_test_split(processed_data, test_size=0.2)
train_data, val_data = train_test_split(train_data, test_size=0.2)
merged_data = pd.concat((train_data, val_data), axis=0)
```

Setelah melakukan import library, model akan melakukan load data yang sudah dilakukan cleansing pada proses sebelumnya. Data kemudian dilakukan splitting dan dites masing masing yang kemudian digabung kembali



NLP (Natural Language Processing)

```
cv = CountVectorizer()  
train_transformed = cv.fit_transform(train_data['clean_text'])  
test_transformed = cv.transform(test_data['clean_text'])  
val_transformed = cv.transform(val_data['clean_text'])
```

Data yang ada dirubah dalam bentuk matriks vektor numerik yang kemudian dilakukan pengetesan untuk menjaga dan memvalidasi konsistensi frekuensi data



NLP (Natural Language Processing)

```
le = LabelEncoder()  
train_label = le.fit_transform(train_data['labels'])  
test_label = le.transform(test_data['labels'])  
val_label = le.transform(val_data['labels'])
```

Penggunaan label encoder untuk melakukan labeling pada kategori dalam data, dimana label data diubah menjadi label kategori yang diinginkan agar merepresantasikan hasil yang konsisten



NLP (Natural Language Processing)

```
mnb = MultinomialNB()  
mlp = MLPClassifier()  
lr = LogisticRegression(max_iter=1000)  
svc = SVC()
```

Model memanggil multinomial NB, MPLC, LR dan SVC untuk memproses data dengan probabilitas, prediksi, regresi logistik dan memisahkan data kedalam kelas.



NLP (Natural Language Processing)

```
MLPC_model = mlp.fit(train_transformed, train_label)
LR_Model = lr.fit(train_transformed, train_label)
MNB_model = mnb.fit(train_transformed, train_label)
SVC_model = svc.fit(train_transformed, train_label)
```

Model kemudian dilakukan train dengan data train pada model MLPC, LR, MNB dan SVC



NLP (Natural Language Processing)

```
[34] pickle.dump(MLPC_model, open('model_mlpc.h5', 'wb'))
```

```
[35] pickle.dump(cv, open('CountVectorizer.pkl', 'wb'))
```

```
[36] pickle.dump(le, open('LabelEncoder.pkl', 'wb'))
```

Model yang sudah dilatih kemudian disimpan dengan pickle yang akan digunakan pada app.py



NLP (Natural Language Processing)

```
y_predict_mlp = mlp.predict(val_transformed)
y_predict_lr = lr.predict(val_transformed)
y_predict_mnb = mnb.predict(val_transformed)
y_predict_svc = svc.predict(val_transformed)
```

Setelah melakukan pelatihan pada model, kami menguji performa dan evaluasi model dengan parameter akurasi, presisi, recall dan skor F!



NLP (Natural Language Processing)

```
print("#####")

print('Accuracy for Logistic Regression')
print(classification_report(y_pred=y_predict_lr, y_true=val_label))

print("#####")

print('Accuracy for Multi Layer Process')
print(classification_report(y_pred=y_predict_mlp, y_true=val_label))

print("#####")

print('Accuracy for Naive Bayes Method')
print(classification_report(y_pred=y_predict_mnb, y_true=val_label))

print("#####")

print('Accuracy for SVC')
print(classification_report(y_pred=y_predict_svc, y_true=val_label))

print("#####")
```

Setelah dilakukan evaluasi, data yang telah diolah diprint untuk melihat hasil yang evaluasi pada proses sebelumnya



NLP (Natural Language Processing)

```
#####
Accuracy for Multi Layer Process
      precision    recall  f1-score   support

     0       0.79      0.79      0.79        570
     1       0.79      0.67      0.72        195
     2       0.87      0.90      0.88        995

 accuracy                   0.84        1760
 macro avg       0.82      0.78      0.80        1760
 weighted avg    0.83      0.84      0.83        1760

#####
```

Kemudian dilanjutkan dengan pemrosesan data dan pelatihan model, pengukuran akurasi dimana akan dinilai tiap iterasi dan mengukur variabilitas.



NLP (Natural Language Processing)

```
k_fold = KFold(n_splits=5,shuffle=True,random_state=0)
```

Penggunaan K-Fold untuk Cross-Validation dimana data dibagi dan dilatih kembali untuk mengukur akurasi

NLP (Natural Language Processing)

```
for train_index, test_index in k_fold.split(train_data):  
    ...  
    train_data_fold = train_data.iloc[train_index]  
    test_data_fold = train_data.iloc[test_index]  
    ...  
    train_kdata_transformed = cv.fit_transform(train_data_fold['clean_text'])  
    test_kdata_transformed = cv.transform(test_data_fold['clean_text'])  
    ...  
    train_klabel = le.fit_transform(train_data_fold['labels'])  
    test_klabel = le.transform(test_data_fold['labels'])  
    ...  
    mlp.fit(train_kdata_transformed, train_klabel)  
    y_kpred = mlp.predict(test_kdata_transformed)  
    accuracy = accuracy_score(y_pred=y_kpred, y_true=test_klabel)  
    print("#####")  
    print("")  
    print(classification_report(y_pred = y_kpred, y_true=test_klabel))  
    ...  
    print("Nilai akurasi model adalah: ")  
    print(accuracy)  
    print("")  
    print("#####")
```

Kemudian dilanjutkan dengan pemrosesan data dan pelatihan model, pengukuran akurasi dimana akan dinilai tiap iterasi dan mengukur variabilitas.

NLP (Natural Language Processing)

	precision	recall	f1-score	support
0	0.77	0.79	0.78	435
1	0.79	0.61	0.69	158
2	0.88	0.90	0.89	815
accuracy			0.83	1408
macro avg	0.81	0.77	0.79	1408
weighted avg	0.83	0.83	0.83	1408

Nilai akurasi model adalah:
0.8330965909090909

	precision	recall	f1-score	support
0	0.81	0.76	0.78	450
1	0.67	0.65	0.66	125
2	0.87	0.90	0.89	833
accuracy			0.83	1408
macro avg	0.78	0.77	0.78	1408
weighted avg	0.83	0.83	0.83	1408

Nilai akurasi model adalah:
0.8338068181818182

	precision	recall	f1-score	support
0	0.76	0.79	0.77	428
1	0.79	0.59	0.68	150
2	0.89	0.90	0.89	830
accuracy			0.84	1408
macro avg	0.81	0.76	0.78	1408
weighted avg	0.84	0.84	0.83	1408

Nilai akurasi model adalah:
0.8366477272727273

	precision	recall	f1-score	support
0	0.78	0.77	0.77	450
1	0.73	0.55	0.62	150
2	0.85	0.90	0.87	808
accuracy			0.82	1408
macro avg	0.79	0.74	0.76	1408
weighted avg	0.82	0.82	0.82	1408

Nilai akurasi model adalah:
0.8188920454545454

	precision	recall	f1-score	support
0	0.77	0.78	0.78	421
1	0.75	0.67	0.71	127
2	0.89	0.90	0.90	860
accuracy			0.84	1408
macro avg	0.80	0.78	0.79	1408
weighted avg	0.84	0.84	0.84	1408

Nilai akurasi model adalah:
0.8444602272727273

Berikut hasil pemrosesan data dan pelatihan model, pengukuran akurasi menggunakan K-Fold dan diperoleh rerata akurasi model yaitu 0.83288



NLP (Natural Language Processing)

```
original_text = "saya pergi ke kantor untuk bekerja"

text = cv.transform([textcleansing(original_text)])

result = MLPC_model.predict(text)[0]
decoded_result = le.inverse_transform([result])
print("Sentiment Encoded:")
print(result)

print("Sentiment Decoded:")
print(decoded_result)

print('Safe Progress')

print('Progress aman sampai disini')
```

Kode ini digunakan untuk melakukan sentiment analisis pada teks menggunakan model yang telah dilatih dan melihat hasil yang diberikan oleh model sebelumnya

```
Sentiment Encoded:
1
Sentiment Decoded:
['neutral']
Safe Progress
Progress aman sampai disini
```



LSTM (Long Short Term Memory)

```
import pandas as pd
import pickle
import re

from keras.preprocessing.text import Tokenizer, text_to_word_sequence
from keras.preprocessing.sequence import pad_sequences
from collections import defaultdict
from sklearn.model_selection import train_test_split
from sklearn import metrics
import numpy as np
from keras import layers
from keras.models import Sequential, load_model, save_model
from keras.layers import Dense, Embedding, LSTM, SpatialDropout1D, SimpleRNN, Activation
from keras import optimizers
from keras.callbacks import EarlyStopping, TensorBoard
from keras.layers import Flatten
from keras import backend as K
import numpy as np
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.model_selection import KFold
import matplotlib.pyplot as plt
import re
from keras.models import load_model
from sklearn.preprocessing import OneHotEncoder
```

Dalam penggunaan metode LSTM untuk membuat model sentiment analysis yang diperlukan, berikut library dan fungsi yang digunakan sebagai bahan pendukung untuk merancang modelnya.

LSTM (Long Short Term Memory)

Dataset yang sudah dibersihkan lalu
disiapkan untuk diproses lebih lanjut

```
df = pd.read_csv('/Users/januardopanggabean/VSCE Platinum Challenge/data/processed_data.csv', header=None ).drop(0)
df.columns=['clean_text', 'stopw_clean','label']
df
```

✓ 0.1s

	clean_text	stopw_clean	label
1	warung ini dimiliki oleh pengusaha pabrik tahu...	warung dimiliki pengusaha pabrik puluhan terke...	positive
2	mohon ulama lurus dan k212 mmbri hujjah paai a...	mohon ulama lurus k212 mmbri hujjah paai diwlh...	neutral
3	lokasi strategis di jalan sumatera bandung tem...	lokasi strategis jalan sumatera bandung nya ny...	positive
4	betapa bahagia nya diri ini saat unboxing pake...	betapa bahagia nya unboxing paket barang nya b...	positive
5	duh jadi mahasiswa jangan sombong dong kasih k...	duh mahasiswa sombong kasih kau kuning belajar...	negative
...
10996	tidak kecewa	kecewa	positive
10997	enak rasa masakan nya apalagi kepiting yang me...	enak masakan nya kepiting menyenangkan memilih...	positive
10998	hormati paai paai yang telah berkoalisi	hormati paai paai berkoalisi	neutral
10999	pagi pagi di tol pasteur sudah macet parah bik...	pagi pagi tol pasteur macet parah bikin jengkel	negative
11000	meskipun sering belanja ke yoga di riau junct...	belanja yoga riau junction peama kali lihat f...	positive

11000 rows x 3 columns

LSTM (Long Short Term Memory)

```
[10] neg = df.loc[df['label'] == 'negative'].clean_text.tolist()
      neu = df.loc[df['label'] == 'neutral'].clean_text.tolist()
      pos = df.loc[df['label'] == 'positive'].clean_text.tolist()

      neg_label = df.loc[df['label'] == 'negative'].label.tolist()
      neu_label = df.loc[df['label'] == 'neutral'].label.tolist()
      pos_label = df.loc[df['label'] == 'positive'].label.tolist()

✓ 0.0s

[11] total_data = pos + neu + neg
      labels = pos_label + neu_label + neg_label

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

✓ 0.0s

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

[12] train_data, test_data = train_test_split(df, test_size=0.2, random_state=0)
      train_data, val_data = train_test_split(train_data, test_size=0.2, random_state=0)

✓ 0.0s
```

Pada bagian ini, masih dalam agenda melihat struktur dan susunan dataset yang akan digunakan. Dapat dilihat bahwa input kalimat yang memiliki label positif memiliki jumlah terbesar. Diikuti data dengan label negatif, dan netral.

Setelah itu semua ditinjau maka dataset dipisahkan menjadi data training, test, dan validasi untuk proses lebih lanjut.

LSTM (Long Short Term Memory)

```
[14] ✓ 0.0s  
tokenizer = Tokenizer(oov_token='<UNK>')  
  
[15] ✓ 0.4s  
tokenizer.fit_on_texts(train_data['clean_text'])  
  
[16] ✓ 0.4s  
train_data_tf = tokenizer.texts_to_sequences(train_data['clean_text'])  
val_data_tf = tokenizer.texts_to_sequences(val_data['clean_text'])  
test_data_tf = tokenizer.texts_to_sequences(test_data['clean_text'])
```

```
max_len = int(np.quantile([len(x) for x in train_data_tf], 0.9))  
  
train_padded = pad_sequences(sequences=train_data_tf,  
                             padding='post', maxlen = max_len)  
val_padded = pad_sequences(sequences=val_data_tf, padding='post', maxlen=max_len)  
test_padded = pad_sequences(sequences=test_data_tf, padding='post', maxlen=max_len)  
✓ 0.0s
```

Disini kita memulai proses Feature Extraction terhadap variabel input text yang akan digunakan dengan menggunakan fungsi Tokenizer dan pengurutan melalui Padding Sequence sehingga variabel text input sudah siap untuk digunakan



LSTM (Long Short Term Memory)

```
onehot = OneHotEncoder()
```

✓ 0.0s

```
train_labels = onehot.fit_transform(train_data[['label']]).toarray()
```

```
valid_labels = onehot.transform(val_data[['label']]).toarray()
```

```
test_labels = onehot.transform(test_data[['label']]).toarray()
```

✓ 0.0s

Tidak lupa juga, variabel output label kita eksekusi proses Feature Extractionnya dengan menggunakan fungsi OneHotEncoder baik untuk fitting parameter data training dan transformasi atas data test dan validasi.

LSTM (Long Short Term Memory) - Model Architecture's Snippets

```
from tensorflow.keras.models import Sequential  
model = Sequential()
```

✓ 0.0s

```
max_features = len(tokenizer.index_word)  
batch_sizes=16  
output_dims=64  
labels_tmp=32  
input_len = max_len
```

✓ 0.0s

```
len(tokenizer.index_word)
```

✓ 0.0s

```
model = Sequential()  
model.add(layers.Embedding(input_dim=len(tokenizer.index_word)+1,  
                           output_dim=output_dims,  
                           input_length=max_len))  
model.add(layers.LSTM(128, dropout=0.2))  
model.add(layers.Dense(64, activation='relu'))  
model.add(layers.Dense(32, activation='relu'))  
model.add(layers.Dense(3, activation='softmax'))
```

✓ 0.5s

```
from keras.optimizers import SGD  
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

✓ 0.0s

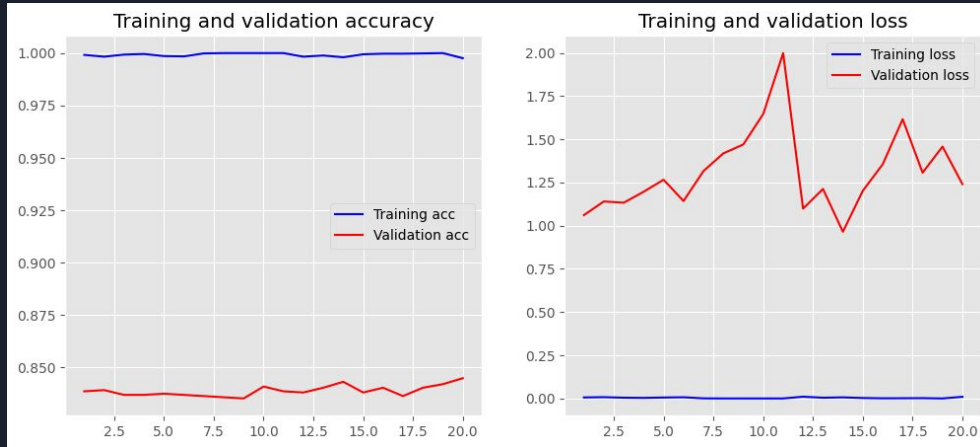
+ Code

+ Markdown

```
maximum_index_training = int(np.floor(train_padded.shape[0]/batch_sizes)*batch_sizes)  
model.fit(x=train_padded[:maximum_index_training],  
          y=train_labels[:maximum_index_training],  
          batch_size=batch_sizes,  
          epochs=20,  
          shuffle=False)
```

✓ 5m 40.9s

LSTM (Long Short Term Memory) - Model Architecture's Snippets



Ketika model yang telah dibuat dianalisis konsistensi akurasi dan loss nya antara data training dan data validasi, hasil visual menunjukkan adanya indikasi underfitting. Hal ini perlu didalami lebih lanjut apakah dikarenakan jumlah sample model yang terbatas atau dikarenakan oleh kesalahan spesifikasi model.



Hasil

```
def predict_paragraph(model, model_no, paragraph):  
    if model_no in [1, 2]:  
        paragraph = text_normalization(paragraph)  
        test_data_transformed = cv.transform([paragraph])  
        y_pred = model.predict(test_data_transformed)  
        y_preds = le.inverse_transform(y_pred)  
        probability = model.predict_proba(test_data_transformed)  
        return y_preds[0], probability [0] [1]
```

Berikut merupakan function untuk mengolah model MLP Classifier



Hasil

```
elif model_no in [3,4]:
    paragraph1 = text_normalization(paragraph)
    paragraph2 = tokenizer.texts_to_sequences([paragraph1])
    print (paragraph2)
    padded_paragraph = pad_sequences(paragraph2,padding='post',maxlen=input_len)

    y_pred = model.predict(padded_paragraph, batch_size=1)

    probability = np.max(y_pred, axis=1)

    y_pred = onehot.inverse_transform(y_pred).reshape(-1)
    return y_pred[0], probability[0]
else:
    return "Model not supported", 0
```

Berikut merupakan function untuk mengolah model LSTM yang telah di input pada app.py

Hasil

```
JSON Data Mentah Header
Simpan Salin Ciutkan Semua Bentangkan Semua Filter JSON
{
  "paragraph": "saya pergi ke kantor untuk bekerja",
  "predicted_category": "neutral",
  "probability": "84%"
}
```

Berikut merupakan hasil dari prediksi dan probabilitas dari MLP Classifier.

Hasil

JSON	Data Mentah	Header
Simpan	Salin	Ciutkan Semua
Bentangkan Semua	Filter JSON	
paragraph:	"saya pergi ke kantor untuk bekerja"	
predicted_category:	"neutral"	
probability:	"91%"	

Berikut merupakan hasil prediksi dan probabilitas dari LSTM.



Kesimpulan

Secara keseluruhan model NLP melalui metode MLP Classifier dan LSTM terlihat sudah dapat berjalan sesuai dengan target awal. Adapun hasil analisis sentimen yang diberikan dengan kedua model tersebut menunjukkan hasil analisis sentimen yang cukup relevan.

Namun perlu diperhatikan lebih lanjut mengenai perlunya fungsi text cleansing atas stopwords karena terdapat temuan mengenai dampak cleansing stopwords yang menunjukkan konten teks dan labelling yang jadi keliru.



Temuan Tambahan - Dampak Stopwords Dalam Cleansing Datasets

1287 tidak terhibur,terhibur,negative

1288 saya tidak senang dengan produk dari tokopedia,senang produk tokopedia,negative

tidak malas,malas,positive

baru sampai lama sekali pengiriman ya,pengiriman ya,negative

tidak jelek,jelek,positive

tidak buruk,buruk,positive

frisian flag rasa kacang ijo kok tidak enak begini sih rasanya,frisian flag kacang ijo enak sih,negative