Import Library

```
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential
from tensorflow.keras import layers
import tensorflow as tf

from tensorflow.keras.layers import Dense, Embedding, Activation, Dropout
from tensorflow.keras.layers import Conv1D, MaxPooling1D, GlobalMaxPooling1D
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
```

Read CSV File

```
df = pd.read_csv('tweet.csv')
df.head()
```

	Abusive	Tweet
0	1	cowok usaha lacak perhati gue lantas remeh per...
1	1	telat tau edan sarap gue gaul cigax jifla cal ...
2	0	41 kadang pikir percaya tuhan jatuh kali kali ...
3	0	ku tau mata sipit lihat
4	1	kaum cebong kafir lihat dongok dungu haha

Drop Missing Rows

```
# drop missing rows
df.dropna(axis=0, inplace=True)
```

Print Lenght of Data

```
text = df["Tweet"].tolist()
print(len(text))
```

```
13121
```

Make it to Categorical

```
y = df["Abusive"]
y = to_categorical(y)
print(y)
#0 itu negatif, 1 itu positif
```

```
[[0. 1.]
 [0. 1.]
 [1. 0.]
 ...
 [1. 0.]
 [1. 0.]
 [0. 1.]]
```

Count Data Each Categorical

```
df["Abusive"].value_counts()
```

```
0    8088
1    5033
Name: Abusive, dtype: int64
```

Do Tokenizer

```
token = Tokenizer()
token.fit_on_texts(text)


# if you want to print tokenizer word, run code below
# token.index_word
```

Print Lenght of Index of Word

```
vocab = len(token.index_word)+1
print(vocab)
```

```
13268
```

Test Text to Tokenize Index

```
x = ['sinting kau ya']
token.texts_to_sequences(x)
```

```
[[558, 1035, 8]]
```

Encode Every Each Tweet Dataset

```
encode_text = token.texts_to_sequences(text)
# if you want to print every tokenizer tweet
# print(encode_text)
```

Do Padding Every Encode Tweet Dataset

```
max_kata = 100
x=pad_sequences(encode_text,maxlen = max_kata, padding="post")
print(x)
```

```
[[ 324  161 3546 ...    0    0    0]
 [1908   49  464 ...    0    0    0]
 [3547  598  101 ...    0    0    0]
 ...
 [  66   66  376 ...    0    0    0]
 [ 111 2819  291 ...    0    0    0]
 [ 569  325    8 ...    0    0    0]]
```

80 20 ratio

Performing learning for 80% data training and 20% data testing.

Split data test and test test

```
x_train,x_test,y_train,y_test = train_test_split(x,y,random_state=1, test_size = 0.2, stratif
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, test_size=0.2, random_sta
```

Change to Data to Array

```
x_train = np.asarray(x_train)
x_test = np.asarray(x_test)
y_train = np.asarray(y_train)
y_test = np.asarray(y_test)
```

Define Model

```
vec_size = 100

model = tf.keras.Sequential()
```

```
model.add(Embedding(vocab,vec_size,input_length=max_kata))
model.add(Conv1D(64,3,activation='relu'))

model.add(GlobalMaxPooling1D())
model.add(Dropout(0.5))

model.add(Dense(2,activation='softmax'))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 100, 100)          1326800
_________________________________________________________________
conv1d (Conv1D)              (None, 98, 64)            19264
_________________________________________________________________
global_max_pooling1d (Global (None, 64)                0
_________________________________________________________________
dropout (Dropout)            (None, 64)                0
_________________________________________________________________
dense (Dense)                (None, 2)                 130
=================================================================
Total params: 1,346,194
Trainable params: 1,346,194
Non-trainable params: 0
_________________________________________________________________
```

```
from keras.metrics import Precision, Recall
model.compile(optimizer="adam",loss="categorical_crossentropy", metrics=['accuracy', Precisio


model.fit(x_train,y_train, epochs=10, validation_data =(x_test,y_test))
```

```
Epoch 1/10
263/263 [==============================] - 12s 40ms/step - loss: 0.4481 - accuracy: 0.78
Epoch 2/10
263/263 [==============================] - 10s 38ms/step - loss: 0.1833 - accuracy: 0.93
Epoch 3/10
263/263 [==============================] - 10s 38ms/step - loss: 0.1118 - accuracy: 0.96
Epoch 4/10
263/263 [==============================] - 11s 40ms/step - loss: 0.0779 - accuracy: 0.97
Epoch 5/10
263/263 [==============================] - 11s 41ms/step - loss: 0.0537 - accuracy: 0.98
Epoch 6/10
263/263 [==============================] - 11s 40ms/step - loss: 0.0389 - accuracy: 0.98
Epoch 7/10
263/263 [==============================] - 11s 40ms/step - loss: 0.0295 - accuracy: 0.99
Epoch 8/10
263/263 [==============================] - 10s 37ms/step - loss: 0.0235 - accuracy: 0.99
Epoch 9/10
263/263 [==============================] - 10s 37ms/step - loss: 0.0210 - accuracy: 0.99
Epoch 10/10
263/263 [==============================] - 10s 38ms/step - loss: 0.0199 - accuracy: 0.99
<keras.callbacks.History at 0x7f4a36e39f90>
```

Evaluate and print Accuracy

```
import keras.backend as K

def f1_score(precision, recall):
    ''' Function to calculate f1 score '''

    f1_val = 2*(precision*recall)/(precision+recall+K.epsilon())
    return f1_val


# Evaluate model on the test set
loss, accuracy, precision, recall = model.evaluate(x_test, y_test, verbose=0)
# Print metrics
print('')
print('Accuracy  : {:.4f}'.format(accuracy))
print('Precision : {:.4f}'.format(precision))
print('Recall    : {:.4f}'.format(recall))
print('F1 Score  : {:.4f}'.format(f1_score(precision, recall)))
```

```
Accuracy  : 0.9131
Precision : 0.9131
Recall    : 0.9131
F1 Score  : 0.9131
```

Get Encode of Predict Data

```
def get_encode(x):
  x = token.texts_to_sequences(x)
  x = pad_sequences(x,maxlen = max_kata, padding = "post")
  return x
```

Get Sentiment Classesof Predict Data

```
def get_sentiment_classes(x):
  x = get_encode(x)
  predict_x=model.predict(x)
  classes_x=np.argmax(predict_x,axis=1)
  sentiment_classes = ['tidak kasar','kasar']
  print('kata tersebut mengandung konotasi',sentiment_classes[classes_x[0]])
```

Predict Data 1

```
# untuk melakukan prediksi kata yang tidak kasar
```

```
get_sentiment_classes(['ibu peri hari ini cantik banget ya'])
```

```
     kata tersebut mengandung konotasi tidak kasar
```

Predict Data 2

```
# untuk melakukan prediksi kata yang kasar
get_sentiment_classes(['bangsat cok raimu koyok asu'])
```

```
     kata tersebut mengandung konotasi kasar
```

70 30 ratio

Performing learning for 70% data training and 30% data testing.

Split data test and test test

```
x_train,x_test,y_train,y_test = train_test_split(x,y,random_state=1, test_size = 0.3, stratif
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, test_size=0.3, random_sta
```

Change to Data to Array

```
x_train = np.asarray(x_train)
x_test = np.asarray(x_test)
y_train = np.asarray(y_train)
y_test = np.asarray(y_test)
```

Define Model

```
vec_size = 100

model = tf.keras.Sequential()
model.add(Embedding(vocab,vec_size,input_length=max_kata))
model.add(Conv1D(64,3,activation='relu'))

model.add(GlobalMaxPooling1D())
model.add(Dropout(0.5))

model.add(Dense(2,activation='softmax'))
model.summary()
```

```
     Model: "sequential_1"
     _________________________________________________________________
     Layer (type)                 Output Shape              Param #
```

```
=================================================================
embedding_1 (Embedding)      (None, 100, 100)          1326800
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 98, 64)            19264
_________________________________________________________________
global_max_pooling1d_1 (Glob (None, 64)                0
_________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 130
=================================================================
Total params: 1,346,194
Trainable params: 1,346,194
Non-trainable params: 0
_________________________________________________________________
```

```
from keras.metrics import Precision, Recall
model.compile(optimizer="adam",loss="categorical_crossentropy", metrics=['accuracy', Precisio
```

```
model.fit(x_train,y_train, epochs=10, validation_data =(x_test,y_test))
```

```
Epoch 1/10
201/201 [==============================] - 9s 42ms/step - loss: 0.4976 - accuracy: 0.751
Epoch 2/10
201/201 [==============================] - 8s 40ms/step - loss: 0.1849 - accuracy: 0.934
Epoch 3/10
201/201 [==============================] - 8s 40ms/step - loss: 0.1006 - accuracy: 0.976
Epoch 4/10
201/201 [==============================] - 8s 42ms/step - loss: 0.0619 - accuracy: 0.981
Epoch 5/10
201/201 [==============================] - 9s 46ms/step - loss: 0.0465 - accuracy: 0.987
Epoch 6/10
201/201 [==============================] - 8s 41ms/step - loss: 0.0278 - accuracy: 0.993
Epoch 7/10
201/201 [==============================] - 8s 39ms/step - loss: 0.0253 - accuracy: 0.992
Epoch 8/10
201/201 [==============================] - 8s 39ms/step - loss: 0.0215 - accuracy: 0.993
Epoch 9/10
201/201 [==============================] - 8s 39ms/step - loss: 0.0172 - accuracy: 0.994
Epoch 10/10
201/201 [==============================] - 8s 40ms/step - loss: 0.0133 - accuracy: 0.997
<keras.callbacks.History at 0x7f4a26b90a50>
```

Evaluate and print Accuracy

```
import keras.backend as K

def f1_score(precision, recall):
    ''' Function to calculate f1 score '''
```

```
    f1_val = 2*(precision*recall)/(precision+recall+K.epsilon())
    return f1_val
```

```
# Evaluate model on the test set
loss, accuracy, precision, recall = model.evaluate(x_test, y_test, verbose=0)
# Print metrics
print('')
print('Accuracy  : {:.4f}'.format(accuracy))
print('Precision : {:.4f}'.format(precision))
print('Recall    : {:.4f}'.format(recall))
print('F1 Score  : {:.4f}'.format(f1_score(precision, recall)))
```

```
Accuracy  : 0.9035
Precision : 0.9035
Recall    : 0.9035
F1 Score  : 0.9035
```

Get Encode of Predict Data

```
def get_encode(x):
  x = token.texts_to_sequences(x)
  x = pad_sequences(x,maxlen = max_kata, padding = "post")
  return x
```

Get Sentiment Classesof Predict Data

```
def get_sentiment_classes(x):
  x = get_encode(x)
  predict_x=model.predict(x)
  classes_x=np.argmax(predict_x,axis=1)
  sentiment_classes = ['tidak kasar','kasar']
  print('kata tersebut mengandung konotasi',sentiment_classes[classes_x[0]])
```

Predict Data 1

```
# untuk melakukan prediksi kata yang tidak kasar
get_sentiment_classes(['ibu peri hari ini cantik banget ya'])
```

```
kata tersebut mengandung konotasi tidak kasar
```

Predict Data 2

```
# untuk melakukan prediksi kata yang kasar
get_sentiment_classes(['bangsat cok raimu koyok asu'])
```

```
kata tersebut mengandung konotasi kasar
```

60 40 ratio

Performing learning for 60% data training and 40% data testing.

Split data test and test test

```
x_train,x_test,y_train,y_test = train_test_split(x,y,random_state=1, test_size = 0.4, stratif
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, test_size=0.4, random_sta
```

Change to Data to Array

```
x_train = np.asarray(x_train)
x_test = np.asarray(x_test)
y_train = np.asarray(y_train)
y_test = np.asarray(y_test)
```

Define Model

```
vec_size = 100

model = tf.keras.Sequential()
model.add(Embedding(vocab,vec_size,input_length=max_kata))
model.add(Conv1D(64,3,activation='relu'))

model.add(GlobalMaxPooling1D())
model.add(Dropout(0.5))

model.add(Dense(2,activation='softmax'))
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
embedding_2 (Embedding)      (None, 100, 100)          1326800   
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 98, 64)            19264     
_________________________________________________________________
global_max_pooling1d_2 (Glob (None, 64)                0         
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0         
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 130       
=================================================================
Total params: 1,346,194
```

```
Trainable params: 1,346,194
Non-trainable params: 0
_________________________________________________________________
```

```
from keras.metrics import Precision, Recall
model.compile(optimizer="adam",loss="categorical_crossentropy", metrics=['accuracy', Precisio
```

```
model.fit(x_train,y_train, epochs=10, validation_data =(x_test,y_test))
```

```
Epoch 1/10
148/148 [==============================] - 8s 46ms/step - loss: 0.5558 - accuracy: 0.716
Epoch 2/10
148/148 [==============================] - 6s 43ms/step - loss: 0.2300 - accuracy: 0.926
Epoch 3/10
148/148 [==============================] - 6s 43ms/step - loss: 0.1212 - accuracy: 0.961
Epoch 4/10
148/148 [==============================] - 7s 46ms/step - loss: 0.0706 - accuracy: 0.978
Epoch 5/10
148/148 [==============================] - 6s 43ms/step - loss: 0.0511 - accuracy: 0.986
Epoch 6/10
148/148 [==============================] - 6s 43ms/step - loss: 0.0330 - accuracy: 0.990
Epoch 7/10
148/148 [==============================] - 6s 43ms/step - loss: 0.0205 - accuracy: 0.994
Epoch 8/10
148/148 [==============================] - 6s 42ms/step - loss: 0.0206 - accuracy: 0.992
Epoch 9/10
148/148 [==============================] - 6s 42ms/step - loss: 0.0109 - accuracy: 0.996
Epoch 10/10
148/148 [==============================] - 6s 43ms/step - loss: 0.0099 - accuracy: 0.997
<keras.callbacks.History at 0x7f4a267c9f10>
```

Evaluate and print Accuracy

```
import keras.backend as K

def f1_score(precision, recall):
    ''' Function to calculate f1 score '''

    f1_val = 2*(precision*recall)/(precision+recall+K.epsilon())
    return f1_val


# Evaluate model on the test set
loss, accuracy, precision, recall = model.evaluate(x_test, y_test, verbose=0)
# Print metrics
print('')
print('Accuracy  : {:.4f}'.format(accuracy))
print('Precision : {:.4f}'.format(precision))
print('Recall    : {:.4f}'.format(recall))
print('F1 Score  : {:.4f}'.format(f1_score(precision, recall)))
```

```
Accuracy  : 0.9025
Precision : 0.9025
Recall    : 0.9025
F1 Score  : 0.9025
```

Get Encode of Predict Data

```
def get_encode(x):
  x = token.texts_to_sequences(x)
  x = pad_sequences(x,maxlen = max_kata, padding = "post")
  return x
```

Get Sentiment Classesof Predict Data

```
def get_sentiment_classes(x):
  x = get_encode(x)
  predict_x=model.predict(x)
  classes_x=np.argmax(predict_x,axis=1)
  sentiment_classes = ['tidak kasar','kasar']
  print('kata tersebut mengandung konotasi',sentiment_classes[classes_x[0]])
```

Predict Data 1

```
# untuk melakukan prediksi kata yang tidak kasar
get_sentiment_classes(['ibu peri hari ini cantik banget ya'])
```

```
kata tersebut mengandung konotasi tidak kasar
```

Predict Data 2

```
# untuk melakukan prediksi kata yang kasar
get_sentiment_classes(['bangsat cok raimu koyok asu'])
```

```
kata tersebut mengandung konotasi kasar
```

✓ 0s completed at 10:32 PM