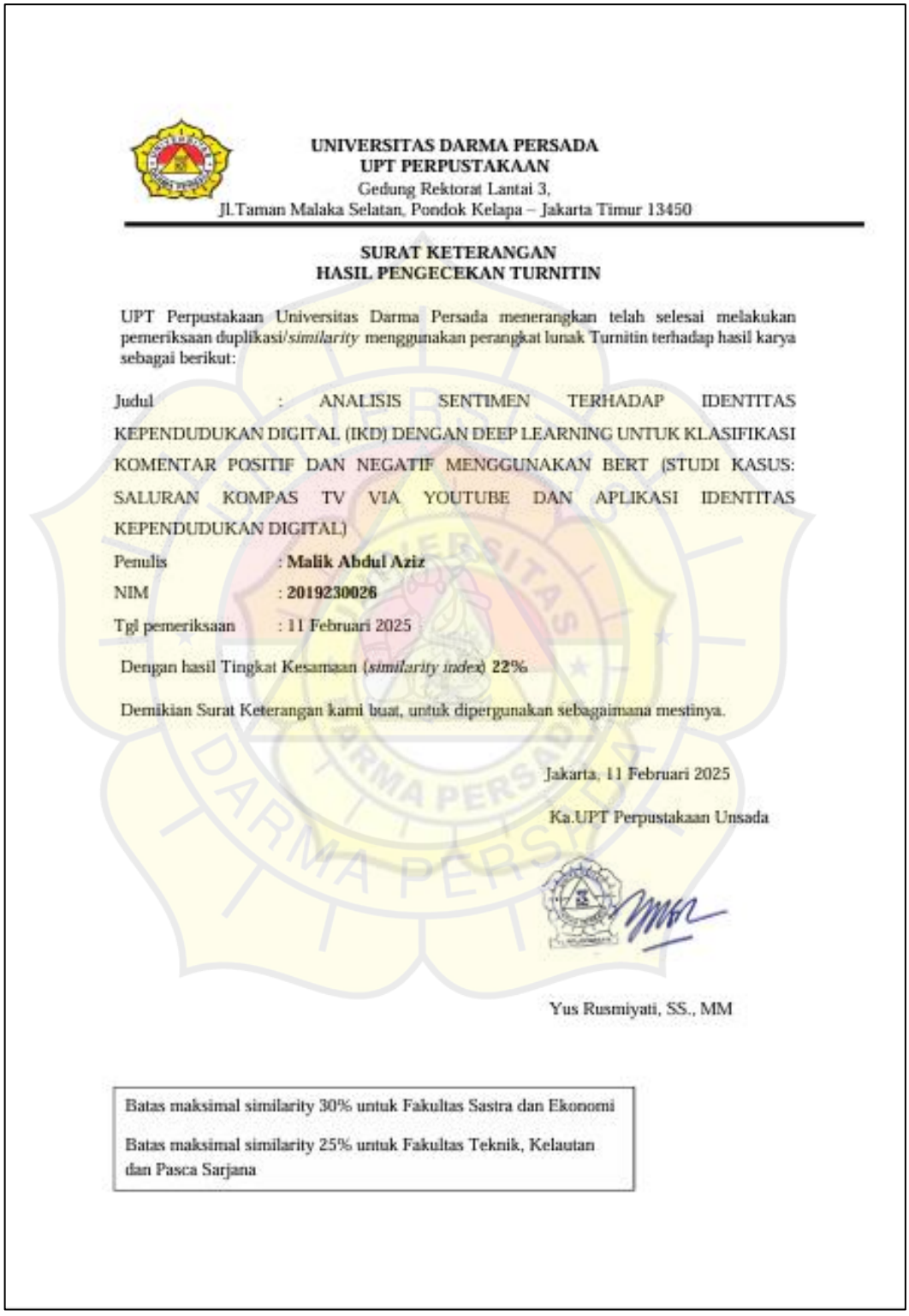
LAMPIRAN

1. Surat Keterangan Hasil Pengecekan Turnitin.



UNIVERSITAS DARMA PERSADA
UPT PERPUSTAKAAN
Gedung Rektorat Lantai 3,
Jl.Taman Malaka Selatan, Pondok Kelapa – Jakarta Timur 13450

SURAT KETERANGAN
HASIL PENGECEKAN TURNITIN

UPT Perpustakaan Universitas Darma Persada menerangkan telah selesai melakukan pemeriksaan duplikasi/*similarity* menggunakan perangkat lunak Turnitin terhadap hasil karya sebagai berikut:

Judul : ANALISIS SENTIMEN TERHADAP IDENTITAS KEPENDUDUKAN DIGITAL (IKD) DENGAN DEEP LEARNING UNTUK KLASIFIKASI KOMENTAR POSITIF DAN NEGATIF MENGGUNAKAN BERT (STUDI KASUS: SALURAN KOMPAS TV VIA YOUTUBE DAN APLIKASI IDENTITAS KEPENDUDUKAN DIGITAL)

Penulis : **Malik Abdul Aziz**

NIM : **2019230026**

Tgl pemeriksaan : 11 Februari 2025

Dengan hasil Tingkat Kesamaan (*similarity index*) **22%**

Demikian Surat Keterangan kami buat, untuk dipergunakan sebagaimana mestinya.

Jakarta, 11 Februari 2025

Ka.UPT Perpustakaan Unsada

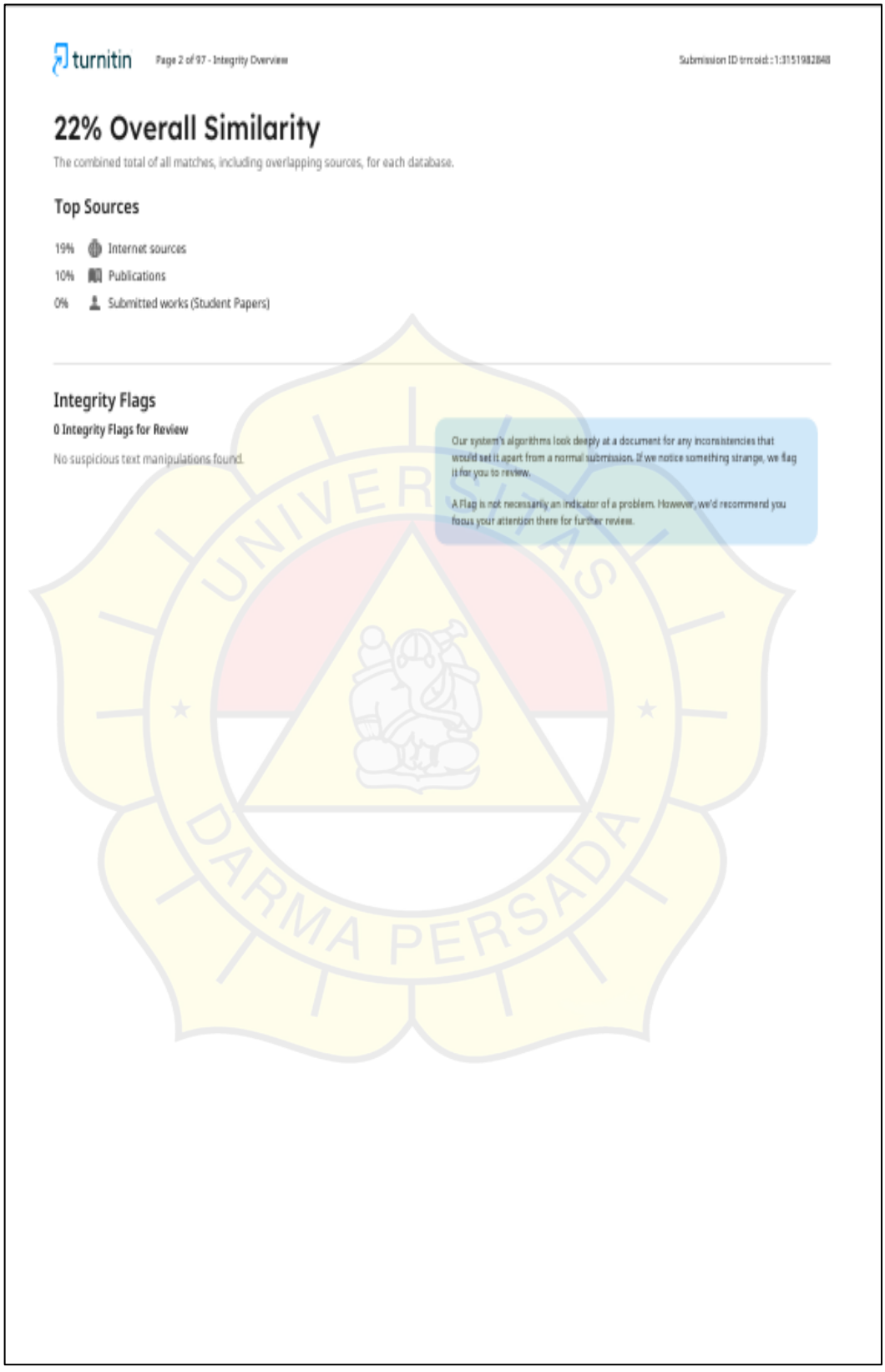
Yus Rusmiyati, SS., MM

Batas maksimal similarity 30% untuk Fakultas Sastra dan Ekonomi

Batas maksimal similarity 25% untuk Fakultas Teknik, Kelautan dan Pasca Sarjana

Lampiran 1 Surat Keterangan Hasil Pengecekan Turnitin

2. Hasil Turnitin



turnitin Page 2 of 97 - Integrity Overview Submission ID trn:oid:::1:3151982848

22% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Top Sources

19% Internet sources
10% Publications
0% Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Lampiran 2 Hasil Turnitin

3. Kode Program Pengambilan Data Komentar di Youtube

```
import pandas as pd
from googleapiclient.discovery import build
```

```
def video_comments(video_id):
  # empty list for storing reply
  replies = []

  # creating youtube resource object
  youtube = build('youtube', 'v3', developerKey=api_key)

  # retrieve youtube video results
  video_response =
youtube.commentThreads().list(part='snippet,replies',
videoId=video_id).execute()

  # iterate video response
  while video_response:

    # extracting required info
    # from each result object
    for item in video_response['items']:

      # Extracting comments ()
      published =
item['snippet']['topLevelComment']['snippet']['publishedAt']
      user =
item['snippet']['topLevelComment']['snippet']['authorDisplayNa
me']

      # Extracting comments
      comment =
item['snippet']['topLevelComment']['snippet']['textDisplay']
      likeCount =
item['snippet']['topLevelComment']['snippet']['likeCount']

      replies.append([published, user, comment, likeCount])

      # counting number of reply of comment
      replycount = item['snippet']['totalReplyCount']

      # if reply is there
      if replycount>0:
        # iterate through all reply
        for reply in item['replies']['comments']:

          # Extract reply
          published = reply['snippet']['publishedAt']
          user = reply['snippet']['authorDisplayName']
          repl = reply['snippet']['textDisplay']
          likeCount = reply['snippet']['likeCount']

          # Store reply is list
          #replies.append(reply)
          replies.append([published, user, repl, likeCount])
```

```
      # print comment with list of reply
      #print(comment, replies, end = '\n\n')

      # empty reply list
      #replies = []

    # Again repeat
    if 'nextPageToken' in video_response:
      video_response = youtube.commentThreads().list(
          part = 'snippet,replies',
          pageToken = video_response['nextPageToken'],
          videoId = video_id
        ).execute()
    else:
      break
  #endwhile
  return replies
```

```
# isikan dengan api key Anda
api_key = " "

# Enter video id
# contoh url video =
https://www.youtube.com/watch?v=5tucmKjOGi8
video_id = "u5pI1fxseII" #isikan dengan kode / ID video

# Call function
comments = video_comments(video_id)

comments
```

```
df = pd.DataFrame(comments, columns=['publishedAt',
'authorDisplayName', 'textDisplay', 'likeCount'])
df
```

```
df.to_csv('Data Percobaan IKD YT.csv', index=False)
df.to_excel('Data Percobaan IKD YT.xlsx', index=False)
```

Lampiran 3 *Source Code* Pengambilan Data Komentar di Youtube

4. Kode Program Transformer GPLAY

```
import random
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline
```

```
import torch

if torch.cuda.is_available():
```

```
  device = torch.device('cuda')

  print('there are %d GPU(s) available.' %
torch.cuda.device_count())

  print('we will use the GPU: ',
torch.cuda.get_device_name(0))

else:
  print("No GPU available, using the CPU instead")
  device = torch.device("cpu")
```

```
%tensorflow_version 2.x

import tensorflow as tf
print(tf.__version__)
```

```
# Install library transformers https://huggingface.co/
!pip -q install transformers
```

```
import transformers
print(transformers.__version__)
```

```
data = pd.read_csv('Sentimen_IKD_YT.csv')
data.head()
```

```
data.drop('publishedAt', axis=1, inplace=True)
data.drop('authorDisplayName', axis=1, inplace=True)
data.drop('likeCount', axis=1, inplace=True)

data.head()
```

```
data['sentimen'].value_counts()
print(data.sentimen.value_counts())
print("TOTAL:" ,data.sentimen.value_counts().sum())
```

```
# mengubah data dari string menjadi integer
from sklearn.preprocessing import LabelEncoder

lb = LabelEncoder()
data['sentimen'] = lb.fit_transform(data['sentimen'])
data.head()
```

```
# copy data
raw_data = data.copy()
```

```
import re

def text_preprocessing (text):
  # Ubah menjadi lower case
  text = text.lower()
  # Hapus URL
  text = re.sub(r'http\S+', '', text)
  # Hapus tanda baca
  text = re.sub(r'[^\w\s]', '', text)
  # Hapus angka
  text = re.sub(r'\d+', '', text)
```

```
  # Hapus emoji
  text = re.sub(r'[^\x00-\x7F]+', '', text)  # Menghapus
karakter non-ASCII (termasuk emoji)
  return text
```

```
data["text_preprocessing"] =
data["content"].apply(text_preprocessing)
data["text_preprocessing"]
```

```
!pip install PySastrawi
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory

factory = StemmerFactory()
stemmer = factory.create_stemmer()
```

```
data["stemmer"] =
data["text_preprocessing"].apply(stemmer.stem)
data["stemmer"]
```

```
kbba_dictionary =
pd.read_csv('https://raw.githubusercontent.com/azizpatuha/kamu
s/refs/heads/main/kbba.txt',
                                delimiter='\t', names=['slang',
'formal'], header=None, encoding='utf-8')

slang_dict = dict(zip(kbba_dictionary['slang'],
kbba_dictionary['formal']))
kbba_dictionary
```

```
def convert_slangword(text):
    words = text.split()

    normalized_words = [slang_dict[word] if word in slang_dict
else word for word in words]
    normalized_text = ' '.join(normalized_words)
    return normalized_text
```

```
data["slang"] = data["stemmer"].apply(convert_slangword)
data["slang"]
```

```
import nltk
from nltk.corpus import stopwords

# Download stopwords Bahasa Indonesia from NLTK
nltk.download('stopwords')
indonesian_stopwords = stopwords.words('indonesian')

indonesian_stopwords.remove('tidak')

# Cache stopwords list outside the function to avoid reloading
every time
cached_stopwords = indonesian_stopwords
```

```
def remove_stopwords(text):
    words = text.split()
    # Hapus stopwords
    filtered_words = [word for word in words if word not in
cached_stopwords]
```

```
    normalized_text = ' '.join(filtered_words)
    return normalized_text
```

```
data["stopword_text"] = data["slang"].apply(remove_stopwords)
data["stopword_text"]
```

```
data.head()
```

```
# merubah nama kolom
data = data.rename (columns={'stopword_text': 'comment'})
data.head()
```

```
data.sentimen.unique()
```

```
data.info()
```

```
data = data.dropna (subset=['content'])
data = data.dropna (subset=['sentimen'])
data = data.dropna (subset=['text_preprocessing'])
data = data.dropna (subset=['stemmer'])
data = data.dropna (subset=['slang'])
data = data.dropna (subset=['comment'])

data.info()
```

```
data.drop('content', axis=1, inplace=True)
data.drop('text_preprocessing', axis=1, inplace=True)
data.drop('stemmer', axis=1, inplace=True)
data.drop('slang', axis=1, inplace=True)

data.head()
```

```
#copy data
raw_data = data.copy()
```

```
from sklearn.model_selection import train_test_split

#Pembagian Data Pelatihan dan Pengujian
df_train, df_test = train_test_split(data, test_size=0.2)
#Pembagian Data Validasi dan Pengujian
df_val, df_test = train_test_split(df_test, test_size=0.5)

df_train.shape, df_test.shape, df_val.shape
```

```
# Menentukan pre trained model yang akan digunakan untuk fine
tuning
PRE_TRAINED_MODEL = 'indobenchmark/indobert-base-p2' #
https://huggingface.co/indobenchmark/indobert-base-p2
```

```
# Import Model BERT
from transformers import BertTokenizer

#Load tokenizer dari pre-trained model BERT
bert_tokenizer =
BertTokenizer.from_pretrained(PRE_TRAINED_MODEL)
```

```
# Lihat vocabulary dari pre-trained model yang telah di load
sebelumnya
```

```
vocabulary = bert_tokenizer.get_vocab()

print('Panjang Vocabulary:', len(vocabulary))
print(vocabulary)
```

```
# Contoh tokenisasi
id_data = 4  #mengambil data ke-4 pada dataframe

print('Kalimat:', raw_data['comment'][id_data])
print('BERT_Tokenizer:',
bert_tokenizer.tokenize(raw_data['comment'][id_data]))
```

```
# Input formalling untuk BERT, sehingga dapat menggunakan
fungsi "encode_plus'
bert_input = bert_tokenizer.encode_plus(
    data.comment[id_data],          # Sample kalimat
    add_special_tokens = True,      # Tambahkan token [CLS]
pada awal kalimat dan token [SEP] pada akhir kalimat
    padding = 'max_length',         # Tambahkan padding ke max
length menggunakan token [PAD] jika kalimat kurang dari
                                    # max length
    truncation = 'longest_first',   # Trucate jika kalimat
lebih dari seluruh kalimat
    max_length = 350,               # Menentukan max_length
dari seluruh kalimat
    return_allenLion_mask=True,     # Mengembalikan nilai
attention mask
    return_Loken_Lype_ids=True,     # Mengembalikan nilai
token type id (Segment Embedding)
)

#Fungsi 'encode plus' mengembalikan 3 nilai: input ids, token
type ids, attention mask
bert_input.keys()
```

```
# Menampilkan informasi yang berbeda mengenai data teks yang
diolah dengan model BERT yang sudah di format dan di
tokenisasi
print('Kalimat\t\t:', raw_data.comment[id_data])
print('Tokenizer\t:', bert_tokenizer.convert_ids_to_tokens
(bert_input['input_ids']))
print('Input IDs\t:', bert_input['input_ids'])
print('Attention Mask\t:', bert_input['attention_mask'])
print('Token Type IDs\t:', bert_input['token_type_ids'])
```

```
# Ada banyak cara untuk menentukan max_length
# Intuisinya adalah kita tidak ingin menolong kalimat, atau
terlalu banyak menambahkan padding (komputasi lebih lama)
# Contoh ini, max_lenght ditentukan dari distribusi token pada
dataset

token_lens = []

for txt in data.comment:
  tokens = bert_tokenizer.encode(txt)
  token_lens.append(len(tokens))

sns.histplot(token_lens, kde=True, stat='density',
```

```
linewidth=0)
plt.xlim([0, 350]);
plt.xlabel('Token Count');
```

```
# Dapat dilihat dari grafik diatas, sebagian besar kalimat
tampaknya kurang 150 token # Maka, tentukan max_length adalah
150
MAX_LEN = 150
```

```
# Buat fungsi untuk menggabungkan langkah tokensasi,
menambahkan special tokens untuk keseluruhan data sebagai
input formatting ke model BERT
def convert_example_to_feature (sentence):
  return bert_tokenizer.encode_plus(
    sentence,
    add_special_tokens = True,
    padding='max_length',
    truncation = 'longest_first',
    max_length= MAX_LEN,
    return_attention_mask=True,
    return_token_type_ids=True
  )
```

```
# Buat fungsi untuk memetakan input hasil input formatting
agar sesuai dengan model BERT
def map_example_to_dict (input_ids, attention_masks,
token_type_ids, label):
  return {
    'input_ids': input_ids,
    'attention_mask': attention_masks,
    'token_type_ids': token_type_ids
  }, label
```

```
# Buat fungsi iterasi pada setiap kalimat pada keseluruhan
def encode(data):
  input_ids_list = [] # Initialize input_ids_list here
  attention_mask_list = [] # Initialize attention_mask_list
here
  token_type_ids_list = [] # Initialize token_type_ids_list
here
  label_list = []

  # Iterate directly over DataFrame rows using iterrows()
  for _, row in data.iterrows():
    label = row['sentimen'] # Assuming 'sentimen' is your
label column
    sentence = row['comment'] # Assuming 'comment' is your
text column

    bert_input = convert_example_to_feature(sentence)

    input_ids_list.append(bert_input['input_ids'])
    token_type_ids_list.append(bert_input['token_type_ids'])
    attention_mask_list.append(bert_input['attention_mask'])
    label_list.append(label)

  return tf.data.Dataset.from_tensor_slices((input_ids_list,
attention_mask_list, token_type_ids_list,
```

```
label_list)).map(map_example_to_dict)
```

```
# Tentukan nilai hyperparameter untuk fine tuning
EPOCHS = 5
BATCH_SIZE = 16
LEARNING_RATE = 5e-5
```

```
# Lakukan input formatting menggunakan fungsi sebelumnya pada
data keseluruhan data
train_encoded = encode(df_train). batch (BATCH_SIZE)
test_encoded = encode(df_test). batch (BATCH_SIZE)
val_encoded = encode(df_val). batch (BATCH_SIZE)
```

```
from transformers import TFBertForSequenceClassification #
Changed to correct class name

# Load model
bert_model =
TFBertForSequenceClassification.from_pretrained(PRE_TRAINED_MO
DEL, num_labels=2)
```

```
# Tentukan optimizer dengan learning rate tertentu
optimizer = tf.keras.optimizers.Adam
(learning_rate=LEARNING_RATE)

# Karena tidak menggunakan one-hot vectors, sehingga loss-
function dapat menggunakan sparse categorical cross entropy
loss = tf.keras.losses.SparseCategoricalCrossentropy
(from_logits=True)

# Metrics untuk evaluasi
metric =
tf.keras.metrics.SparseCategoricalAccuracy('accuracy')

# Compile model
bert_model.compile(optimizer=optimizer, loss=loss,
metrics=[metric])
```

```
%%time
bert_history = bert_model.fit(train_encoded, epochs=EPOCHS,
batch_size=BATCH_SIZE, validation_data=val_encoded)
```

```
# Buat fungsi untuk plotting hasil training
def plot_graph(history, string):
  plt.plot(history.history[string])
  plt.plot(history.history['val_'+string])
  plt.xlabel('Epochs')
  plt.ylabel(string)
  plt.legend([string, 'val '+string])
  plt.show()
```

```
# Menghasilkan plot grafik dari performa model BERT selama
proses pelatihan dan validasi
plot_graph(bert_history, 'accuracy')
plot_graph (bert_history, 'loss')
```

```
# Mencetak informasi tentang performa model BERT pada setiap
epoch selama proses pelatihan dan validasi
```

```
print("\nEpoch No. Train Accuracy Train Loss Val Accuracy val
Loss")
for i in range(EPOCHS):
  print('{:8d} {:12f} \t {:10f} \t {:12f} \t
{:10f}'.format(i+1,
                                                    bert_histo
ry.history['accuracy'][i],
                                                    bert_histo
ry.history['loss'][i],
                                                    bert_histo
ry.history['val_accuracy'][i],
                                                    bert_histo
ry.history['val_loss'][i]) )
```

```
# Mengevaluasi performa model BERT pada data uji yang telah
diubah bentuk yang dapat di proses oleh model
score = bert_model.evaluate(test_encoded)
print('Test Loss:', score[0])
print('Test Accuracy:', score [1])
```

```
# Melakukan prediksi menggunakan model BERT pada data uji yang
telah dibuah menjadi bentuk yang dapat diproses oleh model
predicted_raw = bert_model.predict(test_encoded)
```

```
y_pred = np.argmax(predicted_raw['logits'], axis=1)
y_true = np.array(df_test['sentimen'])
```

```
# Menghitung akurasi dari model BERT pada data Juji dengan
membandingkan label kelas yang sebenarnya # dengan label kelas
yang di prediksi oleh model
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix,
classification_report
accuracy_score (y_true, y_pred)
```

```
# Menghasilkan matriks confusion dari model BERT pada data uji
confusion_matrix(y_true, y_pred)
```

```
# Menghasilkan laporan klasifikasi dari model BERT pada data
uji
print(classification_report (y_true, y_pred))
```

```
bert_model.save_pretrained('/content/drive/My Drive/Colab
Notebooks/bert_model_yt')
```

Lampiran 4 *Source Code* Pemodelan Data Menggunakan *Transformer BERT*