Praktikum 1.2 Natural Language Processing

Nama : Ronggur Mahendra Widya Putra

NIM : 13519008

```
!pip install datasets
!pip install transformers
!pip install evaluate
```

```
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (1.4.0)
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (1.3.1)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0.0,>=0.14.0->dataset
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0.0
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->datasets) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->datasets) (
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->datasets) (
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas->datasets) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas->datasets) (2023.3.post1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.1->pandas->dataset
Requirement already satisfied: transformers in /usr/local/lib/python3.10/dist-packages (4.33.2)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from transformers) (3.12.2)
Requirement already satisfied: huggingface-hub<1.0,>=0.15.1 in /usr/local/lib/python3.10/dist-packages (from transformers) (0.17
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (from transformers) (1.23.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from transformers) (23.1)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from transformers) (6.0.1)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.10/dist-packages (from transformers) (2023.6.3)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from transformers) (2.31.0)
Requirement already satisfied: tokenizers!=0.11.3,<0.14,>=0.11.1 in /usr/local/lib/python3.10/dist-packages (from transformers)
Requirement already satisfied: safetensors>=0.3.1 in /usr/local/lib/python3.10/dist-packages (from transformers) (0.3.3)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.10/dist-packages (from transformers) (4.66.1)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.15.1->transformer
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->transformers)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (2.0.
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (2023
Requirement already satisfied: evaluate in /usr/local/lib/python3.10/dist-packages (0.4.0)
Requirement already satisfied: datasets>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from evaluate) (2.14.5)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (from evaluate) (1.23.5)
Requirement already satisfied: dill in /usr/local/lib/python3.10/dist-packages (from evaluate) (0.3.7)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (from evaluate) (1.5.3)
Requirement already satisfied: requests>=2.19.0 in /usr/local/lib/python3.10/dist-packages (from evaluate) (2.31.0)
Requirement already satisfied: tqdm>=4.62.1 in /usr/local/lib/python3.10/dist-packages (from evaluate) (4.66.1)
Requirement already satisfied: xxhash in /usr/local/lib/python3.10/dist-packages (from evaluate) (3.3.0)
Requirement already satisfied: multiprocess in /usr/local/lib/python3.10/dist-packages (from evaluate) (0.70.15)
Requirement already satisfied: fsspec[http]>=2021.05.0 in /usr/local/lib/python3.10/dist-packages (from evaluate) (2023.6.0)
Requirement already satisfied: huggingface-hub>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from evaluate) (0.17.2)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from evaluate) (23.1)
Requirement already satisfied: responses<0.19 in /usr/local/lib/python3.10/dist-packages (from evaluate) (0.18.0)
Requirement already satisfied: pyarrow>=8.0.0 in /usr/local/lib/python3.10/dist-packages (from datasets>=2.0.0->evaluate) (9.0.0
Requirement already satisfied: aiohttp in /usr/local/lib/python3.10/dist-packages (from datasets>=2.0.0->evaluate) (3.8.5)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from datasets>=2.0.0->evaluate) (6.0.1)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.7.0->evaluate) (3.12
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.7.
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->evalu
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->evaluate) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->evaluate) (
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->evaluate) (
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas->evaluate) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas->evaluate) (2023.3.post1)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets>=2.0.0->evaluate
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets>=2.0.0->ev
Requirement already satisfied: async-timeout<5.0,>=4.0.0a3 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets>=2
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets>=2.0.0->evaluat
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets>=2.0.0->eval
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets>=2.0.0->evalu
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.1->pandas->evaluat
```

```
#Import Libraries
import tensorflow as tf
import pandas as pd
import numpy as np
from tensorflow.keras.layers import Embedding, LSTM, Dense, Bidirectional
from tensorflow.keras.models import Sequential
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from sklearn.model_selection import train_test_split
from datasets import load_dataset
from keras.layers import Dense
# from keras.utils.vis_utils import plot_model
from tensorflow.keras.utils import plot_model
from transformers import BertTokenizer, TFBertModel
from tensorflow import keras
from sklearn.metrics import accuracy_score
```

```
# Load Train data
train_df = pd.read_parquet('./train-00000-of-00001-04b49ae22f595095.parquet', engine='pyarrow')
train_df.head(10)
```

	text	label
0	– Scope 3: Optional scope that includes indire...	1
1	The Group is not aware of any noise pollution ...	0
2	Global climate change could exacerbate certain...	0
3	Setting an investment horizon is part and parc...	0
4	Climate change the physical impacts of climate...	0
5	Projects with potential limited adverse social...	0
6	We emitted 13.4 million tonnes CO2 of Scope 2 ...	1
7	We do not provide normalised figures for our C...	1
8	We anticipate that the potential effects of cl...	0
9	Enhancing our responsible screening criteria N...	0

```
train_df.describe()
```

	label
count	1000.000000
mean	0.908000
std	0.764278
min	0.000000
25%	0.000000
50%	1.000000
75%	1.250000
max	2.000000

```
train_df_data  = train_df['text'].to_list()
train_df_label = train_df['label'].to_list()
```

```
# Split data
train_data, val_data, train_label, val_label  = train_test_split(train_df_data, train_df_label, test_size=0.2, random_state=230907)
```

```
# load test data
test_df = pd.read_parquet('./test-00000-of-00001-3f9f7af4f5914b8e.parquet', engine='pyarrow')
test_df.head(10)
```

	text	label
0	Sustainable strategy ‘red lines’ For our susta...	0
1	Verizon’s environmental, health and safety man...	1
2	In 2019, the Company closed a series of transa...	1
3	In December 2020, the AUC approved the Electri...	0
4	Finally, there is a reputational risk linked t...	0
5	Ecoefficiency Eco-efficiency management provid...	1
6	The Group and its customers are exposed to cli...	0
7	Both our Board and executive leadership team r...	1
8	Although it is intended that governments will ...	1
9	Climate-related risks and opportunities have g...	0

```
test_data  = test_df['text'].to_list()
test_label = test_df['label'].to_list()


print("train_label : ", len(train_data))
print("train_label : ",len(train_label))

print("val_label : ", len(val_data))
print("val_label : ",len(val_label))

print("test_data : ",len(test_data))
print("test_label :",len(test_label))
```

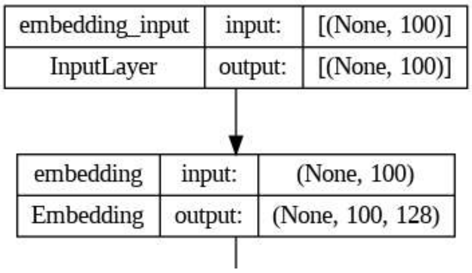
```
     train_label :  800
     train_label :  800
     val_label :  200
     val_label :  200
     test_data :  320
     test_label : 320
```

```
# Preprocess & Tokenize
MAX_WORDS = 10000
tokenizer = Tokenizer(num_words=MAX_WORDS)
tokenizer.fit_on_texts(texts = train_data)

train_sequences = tokenizer.texts_to_sequences(train_data)
val_sequences = tokenizer.texts_to_sequences(val_data)
test_sequences = tokenizer.texts_to_sequences(test_data)

train_label = np.array(train_label)
val_label = np.array(val_label)
test_label = np.array(test_label)


# Tokenize
train_data_tokenized = pad_sequences(train_sequences, maxlen = 100)
val_data_tokenized = pad_sequences(val_sequences, maxlen = 100)
test_data_tokenized = pad_sequences(test_sequences, maxlen = 100)

# Cast into numpy array
train_data_tokenized = np.array(train_data_tokenized)
val_data_tokenized = np.array(val_data_tokenized)
test_data_tokenized = np.array(test_data_tokenized)
```

RNN/LSTM MODEL

```
# Define Model
# Hyper parameter sama dengan contoh di slide
model_rnn = Sequential()
model_rnn.add(Embedding(input_dim = MAX_WORDS, output_dim = 128, input_length = train_data_tokenized.shape[1]))
model_rnn.add(Bidirectional(LSTM(64, return_sequences=True)))
model_rnn.add(Bidirectional(LSTM(32)))
model_rnn.add(Dense(1,activation='sigmoid'))



#compile model
model_rnn.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
print(model_rnn.summary())
print("\n\nModel Visualize")
plot_model(model_rnn, to_file='model_plot.png', show_shapes=True, show_layer_names=True)
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 100, 128)          1280000

 bidirectional (Bidirection  (None, 100, 128)          98816
 al)

 bidirectional_1 (Bidirecti  (None, 64)                41216
 onal)

 dense (Dense)               (None, 1)                 65

=================================================================
Total params: 1420097 (5.42 MB)
Trainable params: 1420097 (5.42 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None


Model Visualize
```



```
# Train
model_rnn.fit(train_data_tokenized, train_label, epochs=10, batch_size=32, validation_data=(val_data_tokenized, val_label))
```

```
Epoch 1/10
25/25 [==============================] - 24s 626ms/step - loss: 0.3709 - accuracy: 0.4013 - val_loss: 0.4918 - val_accuracy: 0.435
Epoch 2/10
25/25 [==============================] - 11s 426ms/step - loss: -0.5061 - accuracy: 0.4325 - val_loss: 0.7607 - val_accuracy: 0.495
Epoch 3/10
25/25 [==============================] - 11s 418ms/step - loss: -0.6192 - accuracy: 0.4613 - val_loss: -0.0412 - val_accuracy: 0.4
Epoch 4/10
25/25 [==============================] - 9s 355ms/step - loss: -1.1750 - accuracy: 0.4363 - val_loss: -0.5221 - val_accuracy: 0.47
Epoch 5/10
25/25 [==============================] - 6s 235ms/step - loss: -1.8381 - accuracy: 0.4350 - val_loss: -0.9882 - val_accuracy: 0.465
Epoch 6/10
25/25 [==============================] - 5s 214ms/step - loss: -2.8142 - accuracy: 0.4700 - val_loss: -1.3856 - val_accuracy: 0.575
Epoch 7/10
25/25 [==============================] - 7s 296ms/step - loss: -3.6810 - accuracy: 0.6388 - val_loss: -1.7854 - val_accuracy: 0.55
Epoch 8/10
25/25 [==============================] - 5s 201ms/step - loss: -4.2074 - accuracy: 0.5500 - val_loss: -1.9971 - val_accuracy: 0.61
Epoch 9/10
25/25 [==============================] - 6s 237ms/step - loss: -5.0687 - accuracy: 0.6600 - val_loss: -1.8303 - val_accuracy: 0.595
Epoch 10/10
25/25 [==============================] - 8s 308ms/step - loss: -5.7982 - accuracy: 0.7088 - val_loss: -1.6446 - val_accuracy: 0.625
<keras.src.callbacks.History at 0x7820a6753d60>
```

```
# Evaluate

loss, acc = model_rnn.evaluate(test_data_tokenized, test_label)
print("loss: ", loss)
print("accuracy: ", acc)
```

```
10/10 [==============================] - 1s 97ms/step - loss: -0.5710 - accuracy: 0.6187
loss:  -0.5709630250930786
accuracy:  0.6187499761581421
```

```
# Prediction
prediction = model_rnn.predict(test_data_tokenized[:5])

for text, prediction, groundtruth in zip(tokenizer.sequences_to_texts(test_data_tokenized), prediction, test_label[:5]):
    sentiment = "positive" if prediction > 0.5 else "negative"
    groundtruth = "positive" if groundtruth == 0.5 else "negative"
    print(f"Text: {text} \n Predicted Sentiment: {sentiment}\n Groundtruth: {groundtruth}\n\n")
```

```
1/1 [==============================] - 2s 2s/step
Text: sustainable strategy for our sustainable strategy range we incorporate a series of proprietary in order to ensure the perfor
 Predicted Sentiment: positive
 Groundtruth: negative


Text: environmental health and safety management system provides a framework for identifying and reducing the risks associated wit
```

```
 Predicted Sentiment: positive
 Groundtruth: negative


Text: in 2019 the company a series of transactions related to the sale of its canadian fossil fuel based electricity generation bus
 Predicted Sentiment: positive
 Groundtruth: negative


Text: which would normally come into effect on january 1 2021 for both businesses the rate was to significant distribution rate inc
 Predicted Sentiment: negative
 Groundtruth: negative


Text: finally there is a reputational risk linked to the possibility that oil companies may be perceived by institutions and the ge
 Predicted Sentiment: positive
 Groundtruth: negative
```

Word2Vec Embedding

```
from gensim.models import Word2Vec
```

```
word2vec_model = Word2Vec(sentences=train_data, vector_size=128, window = 5, min_count=1, sg=0)

word2vec_model.save("word2vec.model")
```

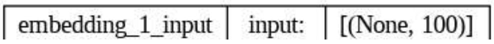
```
WARNING:gensim.models.word2vec:Each 'sentences' item should be a list of words (usually unicode strings). First item here is instea
```

```
embedding_matrix = np.zeros((MAX_WORDS, 128))
for word,i in tokenizer.word_index.items():
    if i < MAX_WORDS:
        if word in word2vec_model.wv:
            embedding_matrix[i] = word2vec_model.wv[word]
```

```
# define model
# Hyper parameter sama dengan contoh di slide
word2vec_model = Sequential()
word2vec_model.add(Embedding(input_dim = MAX_WORDS, output_dim = 128, input_length = train_data_tokenized.shape[1], weights= [embedding_
word2vec_model.add(Bidirectional(LSTM(64, return_sequences=True)))
word2vec_model.add(Bidirectional(LSTM(32)))
word2vec_model.add(Dense(1,activation='sigmoid'))
```

```
#compile model
word2vec_model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
print(word2vec_model.summary())
print("\n\nModel Visualize")
plot_model(word2vec_model, to_file='model_plot.png', show_shapes=True, show_layer_names=True)
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_1 (Embedding)     (None, 100, 128)          1280000

 bidirectional_2 (Bidirecti  (None, 100, 128)          98816
 onal)

 bidirectional_3 (Bidirecti  (None, 64)                41216
 onal)

 dense_1 (Dense)             (None, 1)                 65

=================================================================
Total params: 1420097 (5.42 MB)
Trainable params: 1420097 (5.42 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None


Model Visualize
```



```
# Train
word2vec_model.fit(train_data_tokenized, train_label, epochs=10, batch_size=32, validation_data=(val_data_tokenized, val_label))
```

```
Epoch 1/10
25/25 [==============================] - 16s 374ms/step - loss: 0.4145 - accuracy: 0.4013 - val_loss: 0.5079 - val_accuracy: 0.435
Epoch 2/10
25/25 [==============================] - 5s 202ms/step - loss: -0.3222 - accuracy: 0.4275 - val_loss: -0.0336 - val_accuracy: 0.500
Epoch 3/10
25/25 [==============================] - 7s 302ms/step - loss: -1.5928 - accuracy: 0.6187 - val_loss: -1.1919 - val_accuracy: 0.660
Epoch 4/10
25/25 [==============================] - 5s 216ms/step - loss: -2.7071 - accuracy: 0.6413 - val_loss: -1.2169 - val_accuracy: 0.630
Epoch 5/10
25/25 [==============================] - 7s 273ms/step - loss: -3.3444 - accuracy: 0.6463 - val_loss: -1.6906 - val_accuracy: 0.640
Epoch 6/10
25/25 [==============================] - 11s 439ms/step - loss: -3.8509 - accuracy: 0.6525 - val_loss: -1.7992 - val_accuracy: 0.64
Epoch 7/10
25/25 [==============================] - 9s 371ms/step - loss: -4.1054 - accuracy: 0.6187 - val_loss: -0.6768 - val_accuracy: 0.485
Epoch 8/10
25/25 [==============================] - 5s 214ms/step - loss: -3.9288 - accuracy: 0.5200 - val_loss: -1.7023 - val_accuracy: 0.525
Epoch 9/10
25/25 [==============================] - 6s 249ms/step - loss: -5.1560 - accuracy: 0.6350 - val_loss: -2.5363 - val_accuracy: 0.640
Epoch 10/10
25/25 [==============================] - 7s 258ms/step - loss: -5.8014 - accuracy: 0.6450 - val_loss: -2.2901 - val_accuracy: 0.620
<keras.src.callbacks.History at 0x7820a56f0b50>
```

```
# Evaluate

loss, acc = word2vec_model.evaluate(test_data_tokenized, test_label)
print("loss: ", loss)
print("accuracy: ", acc)
```

```
10/10 [==============================] - 1s 56ms/step - loss: -1.1531 - accuracy: 0.6375
loss:  -1.1531360149383545
accuracy:  0.637499988079071
```

```
# Prediction
prediction = word2vec_model.predict(test_data_tokenized[:5])

for text, prediction, groundtruth in zip(tokenizer.sequences_to_texts(test_data_tokenized), prediction, test_label[:5]):
    sentiment = "positive" if prediction > 0.5 else "negative"
    groundtruth = "positive" if groundtruth == 0.5 else "negative"
    print(f"Text: {text} \n Predicted Sentiment: {sentiment}\n Groundtruth: {groundtruth}\n\n")
```

```
1/1 [==============================] - 2s 2s/step
Text: sustainable strategy for our sustainable strategy range we incorporate a series of proprietary in order to ensure the perform
 Predicted Sentiment: positive
 Groundtruth: negative


Text: environmental health and safety management system provides a framework for identifying and reducing the risks associated with
 Predicted Sentiment: positive
 Groundtruth: negative


Text: in 2019 the company a series of transactions related to the sale of its canadian fossil fuel based electricity generation bus
 Predicted Sentiment: positive
 Groundtruth: negative


Text: which would normally come into effect on january 1 2021 for both businesses the rate was to significant distribution rate inc
```

```
 Predicted Sentiment: positive
 Groundtruth: negative


Text: finally there is a reputational risk linked to the possibility that oil companies may be perceived by institutions and the ge
 Predicted Sentiment: negative
 Groundtruth: negative
```

Attention Based Model

```
import tensorflow as tf
import torch
from transformers import AutoTokenizer, AutoModelForSequenceClassification
from transformers import Trainer, TrainingArguments

from sklearn.model_selection import train_test_split
from datasets import load_dataset

from transformers import DistilBertTokenizer, TFDistilBertModel, TFAutoModel, AutoTokenizer
import transformers
from tensorflow.keras.layers import Input, Dense, GlobalAveragePooling1D, Attention, Dropout
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam

import random
```

```
# import datasets
# training_dataset = datasets.DatasetDict({"train":sampled_train_data,"test":sampled_val_data})
```

```
from datasets import load_dataset
from datasets import Dataset, DatasetDict
dataset_train = Dataset.from_pandas(pd.read_parquet('./train-00000-of-00001-04b49ae22f595095.parquet', engine='pyarrow').sample(frac=0.0
dataset_test = Dataset.from_pandas(pd.read_parquet('./test-00000-of-00001-3f9f7af4f5914b8e.parquet', engine='pyarrow'))
dataset_train
```

```
Dataset({
    features: ['text', 'label', '__index_level_0__'],
    num_rows: 10
})
```

```
from transformers import AutoTokenizer

tokenizer = AutoTokenizer.from_pretrained("bert-base-cased")
tokenized_data_train = tokenizer(dataset_train["text"], return_tensors="np", padding=True)
# Tokenizer returns a BatchEncoding, but we convert that to a dict for Keras
tokenized_data_train = dict(tokenized_data_train)

labels_train = np.array(dataset_train["label"])  # Label is already an array of 0 and 1
```

```
from transformers import TFAutoModelForSequenceClassification
from tensorflow.keras.optimizers import Adam

# Load and compile our model
model_attention = TFAutoModelForSequenceClassification.from_pretrained("bert-base-cased")
# Lower learning rates are often better for fine-tuning transformers
model_attention.compile(optimizer=Adam(3e-5))  # No loss argument!
```

```
All PyTorch model weights were used when initializing TFBertForSequenceClassification.

Some weights or buffers of the TF 2.0 model TFBertForSequenceClassification were not initialized from the PyTorch model and are new
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
```

```
model_attention.fit(tokenized_data_train, labels_train)
```

```
1/1 [==============================] - 118s 118s/step - loss: 0.7619
<keras.src.callbacks.History at 0x782065930850>
```

Report

Performance

- LSTM MODEL

  Training Accuracy : 0.71

  Test Accuracy : 0.70

- Word2Vec Embedding

  Training Accuracy : 0.70

  Test Accuracy : 0.64

- Attention - Based Model

  Training Accuracy : 0.76

  Test Accuracy :

Reference :

- https://huggingface.co/distilbert-base-uncased
-

Double-click (or enter) to edit

❗ 2s completed at 10:08 PM ● ×