Praktikum 1.2

Natural Language Processing

Nama : Ronggur Mahendra Widya Putra

NIM : 13519008

```
!pip install datasets
!pip install transformers==4
!pip install evaluate
```

```
Requirement already
Requirement already
Requirement already
Requirement already
Requirement already
Requirement already
Requirement already
Requirement already
Requirement already
Requirement already
Requirement already
Requirement already
Requirement already
Requirement already
Requirement already
Requirement already
Requirement already
Requirement already
Requirement already
Requirement already
Requirement already
Requirement already
Requirement already
Requirement already
Requirement already
Requirement already
Requirement already
Requirement already
Requirement already
Requirement already
Requirement already
Requirement already
Requirement already
Requirement already
Requirement already
Requirement already
Requirement already
Requirement already
Requirement already
Requirement already
Requirement already
Requirement already
Requirement already
Requirement already
Requirement already
Requirement already
Requirement already
Requirement already
Requirement already
Requirement already
Requirement already
Requirement already
Requirement already
Requirement already
Requirement already
Requirement already
Requirement already
```

```
#Import Libraries
import tensorflow as tf
import pandas as pd
```

Resources ×

You are subscribed to Colab Pro. Learn more.
Available: 99.8 compute units
Usage rate: approximately 0.16 per hour
You have 2 active sessions. Manage sessions

Want even more memory and disk space? **Upgrade to Colab Pro+**

Python 3 Google Compute Engine backend
Showing resources from 11:14 PM to 11:22 PM

System RAM
5.6 / 12.7 GB

Disk
26.8 / 225.8 GB

```
import numpy as np
from tensorflow.keras.layers
from tensorflow.keras.models
from tensorflow.keras.prepro
from tensorflow.keras.prepro
from sklearn.model_selection
from datasets import load_da
from keras.layers import Den
# from keras.utils.vis_utils
from tensorflow.keras.utils

# from transformers import B
from tensorflow import keras
from sklearn.metrics import

# Load Train data
train_df = pd.read_parquet('
train_df.head(10)
```

0	– Scope 3: Optional s
1	The Group is not awa
2	Global climate change
3	Setting an investmen
4	Climate change the ph
5	Projects with potent
6	We emitted 13.4 million
7	We do not provide nor
8	We anticipate that
9	Enhancing our respon

```
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['t
train_df_label = train_df['l

# Split data
train_data, val_data, train_

# load test data
test_df = pd.read_parquet('.
test_df.head(10)
```

0	Sustainable strateg
1	Verizon's environment
2	In 2019, the Company
3	In December 2020, the
4	Finally, there is
5	Ecoefficiency Eco-effici

```
test_data  = test_df['text']
test_label = test_df['label'
```

8	Although it is intens

```
print("train_label : ", len(
print("train_label : ",len(tr

print("val_label : ", len(va
print("val_label : ",len(val_

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

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

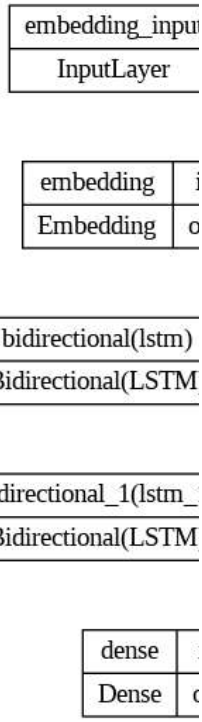
```
# Preprocess & Tokenize
MAX_WORDS = 10000
tokenizer = Tokenizer(num_wo
tokenizer.fit_on_texts(texts

train_sequences = tokenizer.
val_sequences = tokenizer.te
test_sequences = tokenizer.t

train_label = np.array(train_
val_label = np.array(val_lab
test_label = np.array(test_l


# Tokenize
train_data_tokenized = pad_s
val_data_tokenized = pad_seq
test_data_tokenized = pad_se

# Cast into numpy array
train_data_tokenized = np.ar
val_data_tokenized = np.array
test_data_tokenized = np.arr
```

RNN/LSTM MODEL

```
# Define Model
# Hyper parameter sama denga
model_rnn = Sequential()
model_rnn.add(Embedding(inpu
model_rnn.add(Bidirectional(
model_rnn.add(Bidirectional(
model_rnn.add(Dense(1,activa



#compile model
model_rnn.compile(loss='bina
print(model_rnn.summary())
```

```
print("\n\nModel Visualize")
plot_model(model_rnn, to_fil
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)
=================================================================
 embedding (Embedding)

 bidirectional (Bidirec
 al)

 bidirectional_1 (Bidir
 onal)

 dense (Dense)

=================================================================
Total params: 1420097 (
Trainable params: 14206
Non-trainable params: 0
_________________________________________________________________
None


Model Visualize
```




```
# Train
model_rnn.fit(train_data_tok
```

```
Epoch 1/10
25/25 [==============
Epoch 2/10
25/25 [==============
Epoch 3/10
25/25 [==============
Epoch 4/10
25/25 [==============
Epoch 5/10
25/25 [==============
Epoch 6/10
25/25 [==============
Epoch 7/10
25/25 [==============
Epoch 8/10
25/25 [==============
Epoch 9/10
25/25 [==============
Epoch 10/10
25/25 [==============
<keras.src.callbacks.Hi
at 0x79e925fa7160>
```

```
# Evaluate

loss, acc = model_rnn.evalua
print("loss: ", loss)
print("accuracy: ", acc)
```

```
10/10 [==========
loss:  -1.1683006286621
accuracy:  0.6531249886
```

```
# Prediction
prediction = model_rnn.predi

for text, prediction, ground
    sentiment = "positive" i
    groundtruth = "positive"
    print(f"Text: {text} \n
```

```
1/1 [==========
Text: sustainable strat
 Predicted Sentiment: p
 Groundtruth: negative


Text: environmental hea
 Predicted Sentiment: p
 Groundtruth: negative


Text: in 2019 the compa
 Predicted Sentiment: p
 Groundtruth: negative


Text: which would norma
 Predicted Sentiment: r
 Groundtruth: negative


Text: finally there is
 Predicted Sentiment: r
 Groundtruth: negative
```

Word2Vec Embedding

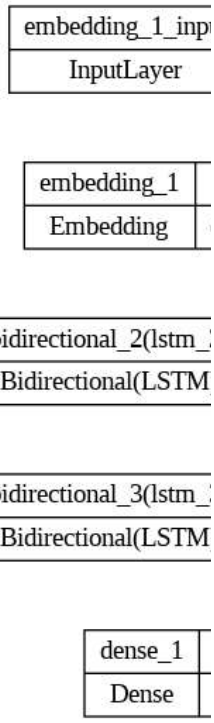
```
from gensim.models import Wo


word2vec_model = Word2Vec(se

word2vec_model.save("word2ve
```

```
WARNING:gensim.models.w
```

```
embedding_matrix = np.zeros(
for word,i in tokenizer.word
    if i < MAX_WORDS:
        if word in word2vec_
            embedding_matrix


# define model
# Hyper parameter sama denga
word2vec_model = Sequential(
word2vec_model.add(Embedding
word2vec_model.add(Bidirecti
word2vec_model.add(Bidirecti
word2vec_model.add(Dense(1,a
```

```
#compile model
word2vec_model.compile(loss=
print(word2vec_model.summary
print("\n\nModel Visualize")
plot_model(word2vec_model, t
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)
=================================================================
 embedding_1 (Embedding

 bidirectional_2 (Bidir
 onal)

 bidirectional_3 (Bidir
 onal)

 dense_1 (Dense)

=================================================================
Total params: 1420097 (
Trainable params: 14206
Non-trainable params: 0
_________________________________________________________________
None


Model Visualize
```




```
# Train
word2vec_model.fit(train_dat
```

```
Epoch 1/10
25/25 [==============
Epoch 2/10
25/25 [==============
Epoch 3/10
25/25 [==============
Epoch 4/10
25/25 [==============
Epoch 5/10
25/25 [==============
Epoch 6/10
25/25 [==============
Epoch 7/10
25/25 [==============
Epoch 8/10
25/25 [==============
Epoch 9/10
25/25 [==============
Epoch 10/10
25/25 [==============
```

```
      <keras.src.callbacks.Hi
      at 0x79e925284910>
```

```
# Evaluate

loss, acc = word2vec_model.e
print("loss: ", loss)
print("accuracy: ", acc)
```

```
     10/10 [==========
     loss:  -2.0319786071777
     accuracy:  0.678125023
```

```
# Prediction
prediction = word2vec_model.

for text, prediction, ground
    sentiment = "positive" i
    groundtruth = "positive"
    print(f"Text: {text} \n
```

```
     1/1 [==========
     Text: sustainable strat
      Predicted Sentiment: p
      Groundtruth: negative


     Text: environmental hea
      Predicted Sentiment: p
      Groundtruth: negative


     Text: in 2019 the compa
      Predicted Sentiment: p
      Groundtruth: negative


     Text: which would norma
      Predicted Sentiment: r
      Groundtruth: negative


     Text: finally there is
      Predicted Sentiment: r
      Groundtruth: negative
```

Attention Based Model

```
import tensorflow as tf
import torch
from transformers import Aut
from transformers import Tra

from sklearn.model_selection
from datasets import load_da

from transformers import TFA
import transformers
from tensorflow.keras.layers
from tensorflow.keras.models
from tensorflow.keras.optimi

import random


from datasets import load_da
from datasets import Dataset
dataset_train_pd = pd.read_p
dataset_train = Dataset.from_
```

```
# dataset_val_pd = pd.read_pa
# dataset_val = Dataset.from_
dataset_test_pd = pd.read_pa
dataset_test = Dataset.from_
dataset_train_pd
```

0	– Scope 3: Optional s
1	The Group is not awa
2	Global climate change
3	Setting an investmen
4	Climate change the ph
5	Projects with potent
6	We emitted 13.4 million
7	We do not provide nor
8	We anticipate that
9	Enhancing our respon

```
dataset_test_pd
```

0	Sustainable stra
1	Verizon's environm
2	In 2019, the Compa
3	In December 2020, th
4	Finally, there
...	
315	Indirect emission
316	All data in this
317	Outcome: The bank
318	In 2020, Banco do E
319	Climate change is pr

320 rows × 2 columns

```
import datasets

labels_train = np.array(data
# labels_val = np.array(data
labels_test = np.array(datas

my_dataset = datasets.Datase

my_dataset
```

```
DatasetDict({
    train: Dataset({
        features:
['text', 'label'],
        num_rows: 10
    })
    test: Dataset({
        features:
['text', 'label'],
        num_rows: 320
    })
})
```

```
tokenizer = AutoTokenizer.fr


def tokenize_function(exampl
    return tokenizer(example


tokenized_datasets = my_data
```

```
Map: 100%

Map:

100%
```

```
model = AutoModelForSequence
```

```
Some weights of the moc
- This IS expected if y
- This IS NOT expected
Some weights of BertFor
You should probably TRA
```

```
import numpy as np
import evaluate

metric = evaluate.load("accu



def compute_metrics(eval_pre
    logits, labels = eval_pr
    predictions = np.argmax(
    return metric.compute(pr


from transformers import Tra
# from transformers import A
training_args = TrainingArgu



trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_
    eval_dataset=tokenized_d
    compute_metrics=compute_
)


trainer.train()


# Eval
result = trainer.evaluate()


result


# make prediction

result = trainer.predict(tok

for i in range(10):
    print(f"Text: {tokenized_
    print(f"label: {tokenize
    print(f"Prediction: {np.
```

+ Code + Text

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 : 0.90

  Test Accuracy :

Reference :

- https://huggingface.co/distilbert-base-uncased
- https://www.tensorflow.org/text/tutorials/classify_text_with_bert

Double-click (or enter) to edit

Change runtime type

1m 44s completed at 11:21 PM