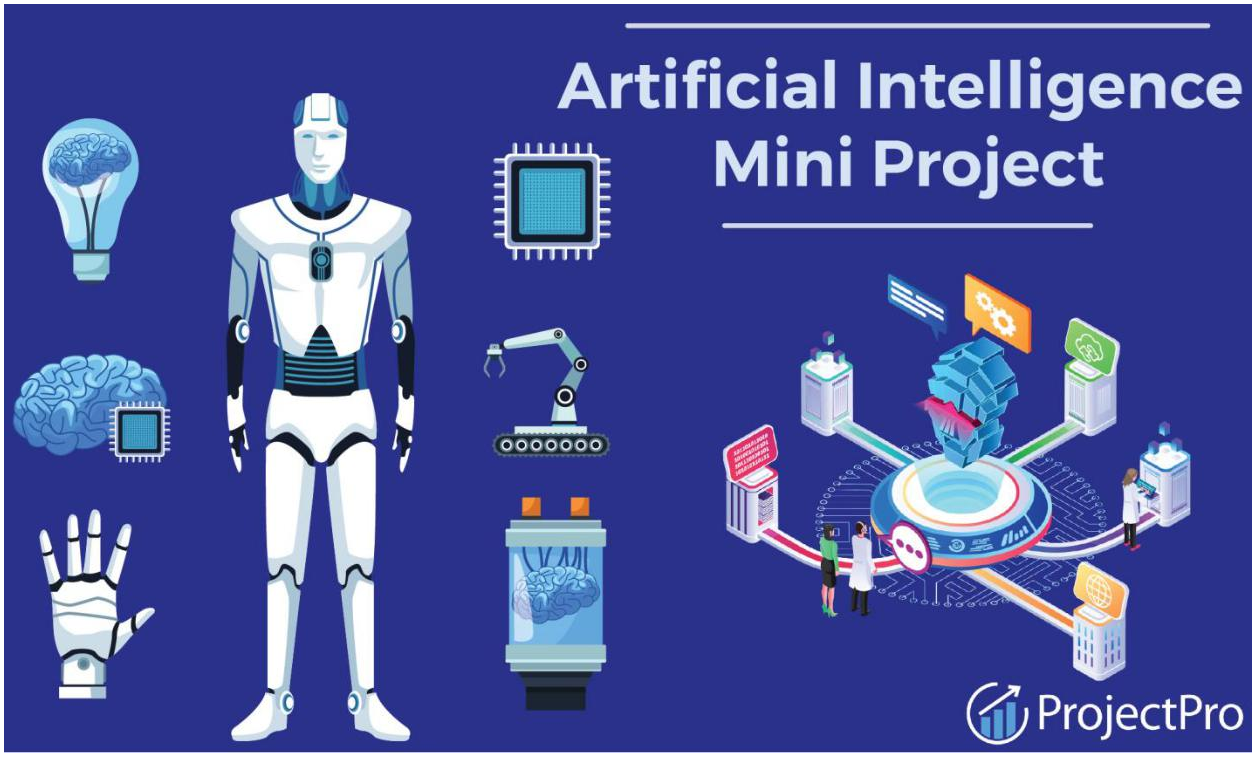
Artificial Intelligence Mini Project



Autores: Vinícius Gabriel Santos Vidal e Danilo Rodrigues Torchio

Project Title: Protótipo de aplicativo para classificação de sentimento

Project Description: Nesse projeto foi desenvolvido um aplicativo para classificação de sentimento de uma frase que o usuário fala para o app.

Steps Involved:

1. **Preparação dos dados:** Para o treinamento do modelo, foi utilizado uma base de dados de tweets já classificados em 3 grupos de sentimento: positivo, negativo e neutro. No tratamento dos dados, foram retirados caracteres especiais e letra maiúsculas foram passadas para minúscula.
2. **Construção do modelo:** Foi desenvolvido um modelo de rede neural baseado em BERT. Também foi utilizado o tokenizador do BERT por ser o um dos mais eficientes para tarefas de classificação atualmente.

3. **Treinamento do modelo:** Para o treinamento foi utilizado 80% para treinamento e 20% para validação. O otimizador escolhido foi o Adam, a função de perda foi utilizada a *Categorical Cross-Entropy.*
4. **Ajuste fino do modelo:** Para ajustar melhor o modelo, foram testados valores de *decay* e *learning rate* diferentes. Para o modelo final, foi escolhido o que teve melhor performance.
5. **Implantação do modelo:** O modelo final foi preparado para receber qualquer frase e classificá-la.

Código do Projeto:

#Importing Necessary Libraries

```
import pandas as pd
import numpy as np
from tqdm.auto import tqdm
import tensorflow as tf
from transformers import BertTokenizer
from nltk.corpus import stopwords
from contextlib import redirect_stdout
import keras
import matplotlib.pyplot as plt
from sklearn.datasets import make_classification
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from transformers import TFBertModel
```

#Functions

```
def SentimentDatasetMapFunction(input_ids, attn_masks, labels):
    return {
        'input_ids': input_ids,
        'attention_mask': attn_masks
    }, labels


def generate_training_data(df, ids, masks, tokenizer):
    for i, text in tqdm(enumerate(df['tweet_text'])):
        tokenized_text = tokenizer.encode_plus(
            text,
            max_length=256,
            truncation=True,
            padding='max_length',
            add_special_tokens=True,
            return_tensors='tf'
        )
        ids[i, :] = tokenized_text.input_ids
        masks[i, :] = tokenized_text.attention_mask
    return ids, masks
```

#Importing the Dataset

```
df = pd.read_csv("endereco do dataset")
df ["id"] = df.index + 1
```

#Shuffle the Dataset

```
df = df.sample(frac = 1)
```

#Data Preprocessing

```
df['tweet_text'] = df['tweet_text'].str.replace(':', '')
df['tweet_text'] = df['tweet_text'].str.replace(')', '')
df['tweet_text'] = df['tweet_text'].str.replace('(', '')

tokenizer = BertTokenizer.from_pretrained('bert-base-cased')

model = TFBertModel.from_pretrained('bert-base-cased')


X_input_ids, X_attn_masks = generate_training_data(df, X_input_ids, X_attn_masks,

tokenizer)

labels = np.zeros((len(df), 3))

labels[np.arange(len(df)), df['labels'].values.tolist()] = 1 # one-hot encoded target

tensor

dataset = tf.data.Dataset.from_tensor_slices((X_input_ids, X_attn_masks, labels))

dataset = dataset.map(SentimentDatasetMapFunction)

dataset = dataset.shuffle(10000).batch(16, drop_remainder=True)

p = 0.8

df_size = int((len(df)//16)*p)

df_dataset = dataset.take(df_size)

val_dataset = dataset.skip(df_size)
```

#Model

```
input_ids = tf.keras.Input(shape=(256,), name='input_ids', dtype='int32')

attn_masks = tf.keras.Input(shape=(256,), name='attention_mask', dtype='int32')

bert_embds = model.bert(input_ids, attention_mask=attn_masks)[1]

intermediate_layer = tf.keras.layers.Dense(param['dense'], activation='relu',

name='intermediate_layer')(bert_embds)

output_layer = tf.keras.layers.Dense(3, activation='softmax',

name='output_layer')(intermediate_layer)
```

```
rr_model = tf.keras.Model(inputs=[input_ids, attn_masks], outputs=output_layer)
    rr_model.summary()

optim = tf.keras.optimizers.legacy.Adam(learning_rate=param['learning_rate'],
decay=1e-6)
    loss_func = tf.keras.losses.CategoricalCrossentropy()
    acc = tf.keras.metrics.CategoricalAccuracy('accuracy')

    rr_model.compile(optimizer=optim, loss=loss_func, metrics=[acc])
```

#Train Model

```
hist = rr_model.fit(

    df_dataset,

    validation_data=val_dataset,

    epochs=2

    )
```

#Test Model with a test dataset

```
out =[]
for i in test['tweet_text']:
    pred = model.predict(prepare_data(i, tokenizer))
    out.append(np.argmax(pred).tolist())

y_true = test["labels"].tolist()

y_pred = out

metrics = classification_report(y_true, y_pred)
```