BERT x BERTimbau

[ ]:
```
# !pip install -r requirements.txt
# !pip install --upgrade ipywidgets
# !pip install tf-keras
```

Carregar bibliotecas

[ ]:
```
import numpy as np
from tensorflow.keras.callbacks import EarlyStopping
import tensorflow as tf
tf.autograph.set_verbosity(0)  # Para suprimir os avisos do AutoGraph
from transformers import BertTokenizer, TFBertModel, BertConfig,␣
 ↪TFBertForSequenceClassification
from keras.layers import *
from keras.models import Model, Sequential
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
 ↪f1_score
```

0.1 Carregar base

News of the Brazilian Newspaper

A base de dados consiste em 167.053 exemplos e contém Manchetes, URL do Artigo, Artigo Completo e Categoria. As notícias foram resumidas do Inshorts e coletadas apenas os artigos de notícias do Jornal Folha de São Paulo. O período de tempo varia entre janeiro de 2015 e setembro de 2017.

[ ]:
```
import pandas as pd

# Caminho para o arquivo no Google Drive
caminho_pasta = 'articles.csv'

# Carregar o arquivo CSV usando Pandas
df = pd.read_csv(caminho_pasta, delimiter=',')
```

[ ]:
```
contagem_categorias = df['category'].nunique()
print('Categorias diferentes:', contagem_categorias)
```

```
contagem_subcategorias = df['subcategory'].nunique()
print('Subcategorias diferentes:', contagem_subcategorias)
```

```
Categorias diferentes: 48
Subcategorias diferentes: 293
```

A base possui seis colunas:

- ***title*** (título da notícia)
- ***text*** (texto da notícia)
- ***date*** (data da publicação)
- ***category*** (48 categorias): poder, ilustrada, mercado, mundo, esporte, tec, cotidiano, ambiente, equilibrioesaude, sobretudo, colunas, educacao, tv, banco-de-dados, opiniao, ciencia, paineldoleitor, saopaulo, ilustrissima, seminariosfolha, turismo, empreendedorsocial, serafina, asmais, o-melhor-de-sao-paulo, bbc, comida, musica, folhinha, especial, treinamento, multimidia, cenarios-2017, topofmind, dw, ombudsman, contas-de-casa, mulher, 2016, guia-de-livros-discos-filmes, treinamentocienciaesaude, rfi, vice, bichos, euronews, guia-de-livros-filmes-discos, infograficos, 2015
- ***subcategory*** (293 subcategorias)
- ***link*** (fonte da notícia)

[ ]:
```
df.head()
```

[ ]:
```
                                               title  \
0  Lula diz que está 'lascado', mas que ainda tem…
1  'Decidi ser escrava das mulheres que sofrem', …
2  Três reportagens da Folha ganham Prêmio Petrob…
3  Filme 'Star Wars: Os Últimos Jedi' ganha trail…
4  CBSS inicia acordos com fintechs e quer 30% do…

                                                text        date   category  \
0  Com a possibilidade de uma condenação impedir …  2017-09-10      poder
1  Para Oumou Sangaré, cantora e ativista malines…  2017-09-10  ilustrada
2  Três reportagens da Folha foram vencedoras do …  2017-09-10      poder
3  A Disney divulgou na noite desta segunda-feira…  2017-09-10  ilustrada
4  O CBSS, banco da holding Elopar dos sócios Bra…  2017-09-10    mercado

  subcategory                                               link
0         NaN  http://www1.folha.uol.com.br/poder/2017/10/192…
1         NaN  http://www1.folha.uol.com.br/ilustrada/2017/10…
2         NaN  http://www1.folha.uol.com.br/poder/2017/10/192…
3         NaN  http://www1.folha.uol.com.br/ilustrada/2017/10…
4         NaN  http://www1.folha.uol.com.br/mercado/2017/10/1…
```

1 Pré-processamento de dados

Como o intuito deste notebook é classificar as categorias, desconsiderou-se as colunas ***subcategory***, ***date*** e ***link***. A coluna ***title*** foi concatenada com ***text*** para também ser considerado.

[ ]:
```
df['text'] = df['title'] + ' ' + df['text']
remove = ['title', 'subcategory', 'link', 'date']
df = df.drop(remove, axis=1)
```

Foram removidos registros com valores nulos.

[ ]:
```
df = df.dropna()
```

[ ]:
```
print(df.shape)
df
```

```
(166288, 2)
```

[ ]:

	text	category
0	Lula diz que está 'lascado', mas que ainda tem…	poder
1	'Decidi ser escrava das mulheres que sofrem', …	ilustrada
2	Três reportagens da Folha ganham Prêmio Petrob…	poder
3	Filme 'Star Wars: Os Últimos Jedi' ganha trail…	ilustrada
4	CBSS inicia acordos com fintechs e quer 30% do…	mercado
…	…	…
167048	Em cenário de crise, tucano Beto Richa assume …	poder
167049	Filho supera senador Renan Calheiros e assume …	poder
167050	Hoje na TV: Tottenham x Chelsea, Campeonato In…	esporte
167051	Kim Jong-un diz estar aberto a se reunir com p…	mundo
167052	Heitor Maia Neto (1928 - 2014) - Pioneiro da a…	cotidiano

[166288 rows x 2 columns]

1.0.1 Ajuste nos dados

Por uma questão de limitação de recursos computacionais, testarei os modelos apenas com as sete categorias mais numerosas, considerando apenas os 100 primeiros registros de cada uma.

[ ]:
```
# Categorias com mais registros
manter = ['poder', 'colunas', 'mercado', 'esporte', 'mundo', 'cotidiano',␣
 ↪'ilustrada']

# Deixar apenas as categorias definidas
df = df[df['category'].isin(manter)]
```

[ ]:
```
# Criando a amostra estratificada
_ , df_amostra = train_test_split(df, test_size=0.05, stratify=df['category'],␣
 ↪random_state=10)
```

[ ]:
```
df_amostra2 = df_amostra.groupby('category').head(100)
```

[ ]:
```
df_amostra2['category'].value_counts()
```

[ ]:
```
category
mundo        100
esporte      100
poder        100
ilustrada    100
cotidiano    100
colunas      100
mercado      100
Name: count, dtype: int64
```

Dividir em treinamento (80%) e teste (20%)

[ ]:
```
# Exemplo de preprocessamento dos dados
X = df_amostra2['text'].tolist()
y = df_amostra2['category'].tolist()

# Dividir em treino e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
```

Verificar o tamanho dos dados divididos

[ ]:
```
print(f'Shape de X_train: {len(X_train)}, {len(X_train[0])}')
print(f'Shape de X_test: {len(X_test)}, {len(X_test[0])}')
print(f'Shape de y_train: {len(y_train)}')
print(f'Shape de y_test: {len(y_test)}')
```

```
Shape de X_train: 560, 1924
Shape de X_test: 140, 3181
Shape de y_train: 560
Shape de y_test: 140
```

1.1 BERT

O BERT é treinado em textos multilínguas e, portanto, pode capturar padrões gerais de linguagem que se aplicam bem a diferentes idiomas, incluindo o português. Ele é pré-treinado em um grande corpus que inclui diversos idiomas, sendo robustos para tarefas multilíngues.

[ ]:
```
# Carregar o tokenizer e o modelo BERT
tokenizer = BertTokenizer.from_pretrained('bert-base-multilingual-cased')
bert_model = TFBertModel.from_pretrained('bert-base-multilingual-cased')
# Congelar os pesos do BERT
bert_model.trainable = False
```

```
2024-07-01 19:56:38.163206: I
external/local_xla/xla/stream_executor/cuda/cuda_executor.cc:998] successful
NUMA node read from SysFS had negative value (-1), but there must be at least
one NUMA node, so returning NUMA node zero. See more at
https://github.com/torvalds/linux/blob/v6.0/Documentation/ABI/testing/sysfs-bus-
pci#L344-L355
2024-07-01 19:56:38.163617: W
tensorflow/core/common_runtime/gpu/gpu_device.cc:2251] Cannot dlopen some GPU
libraries. Please make sure the missing libraries mentioned above are installed
properly if you would like to use GPU. Follow the guide at
https://www.tensorflow.org/install/gpu for how to download and setup the
required libraries for your platform.
Skipping registering GPU devices…
2024-07-01 19:56:38.413516: W
external/local_tsl/tsl/framework/cpu_allocator_impl.cc:83] Allocation of
367248384 exceeds 10% of free system memory.
2024-07-01 19:56:38.609040: W
external/local_tsl/tsl/framework/cpu_allocator_impl.cc:83] Allocation of
367248384 exceeds 10% of free system memory.
2024-07-01 19:56:38.651953: W
external/local_tsl/tsl/framework/cpu_allocator_impl.cc:83] Allocation of
367248384 exceeds 10% of free system memory.
2024-07-01 19:56:39.584358: W
external/local_tsl/tsl/framework/cpu_allocator_impl.cc:83] Allocation of
367248384 exceeds 10% of free system memory.
Some weights of the PyTorch model were not used when initializing the TF 2.0
model TFBertModel: ['cls.predictions.transform.LayerNorm.bias',
'cls.seq_relationship.bias', 'cls.predictions.transform.dense.weight',
'cls.predictions.transform.LayerNorm.weight', 'cls.predictions.bias',
'cls.predictions.transform.dense.bias', 'cls.seq_relationship.weight']
- This IS expected if you are initializing TFBertModel from a PyTorch model
trained on another task or with another architecture (e.g. initializing a
TFBertForSequenceClassification model from a BertForPreTraining model).
- This IS NOT expected if you are initializing TFBertModel from a PyTorch model
that you expect to be exactly identical (e.g. initializing a
TFBertForSequenceClassification model from a BertForSequenceClassification
model).
All the weights of TFBertModel were initialized from the PyTorch model.
If your task is similar to the task the model of the checkpoint was trained on,
you can already use TFBertModel for predictions without further training.
```

[ ]:
```
# Tokenizar os textos
max_length = 128
train_encodings = tokenizer(X_train, padding=True, truncation=True,␣
 ↪max_length=max_length, return_tensors='tf')
test_encodings = tokenizer(X_test, padding=True, truncation=True,␣
 ↪max_length=max_length, return_tensors='tf')
```

O modelo BERT para classificação de sequências é carregado para ser capaz de classificar as 7 classes. O modelo é inicializado usando pesos pré-treinados do BERT multilíngua. Os pesos da camada BERT são congelados para não serem retreinados. Os rótulos de classe são então transformados para valores numéricos usando LabelEncoder. Por fim, é necessário compilar o modelo. Foi utilizado o otimizador Adam.

[ ]:
```
# Criando uma MLP como classificador
model = tf.keras.Sequential([
    tf.keras.layers.Dense(256, activation='relu', input_shape=(bert_model.
 ↪config.hidden_size,)),
    tf.keras.layers.Dense(256, activation='relu'),
    tf.keras.layers.Dense(256, activation='relu'),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(7)  # 7 classes de saída
])
```

[ ]:
```
model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_11 (Dense)            (None, 256)               196864

 dense_12 (Dense)            (None, 256)               65792

 dense_13 (Dense)            (None, 256)               65792

 dense_14 (Dense)            (None, 128)               32896

 dense_15 (Dense)            (None, 128)               16512

 dense_16 (Dense)            (None, 7)                 903

=================================================================
Total params: 378759 (1.44 MB)
Trainable params: 378759 (1.44 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_11 (Dense)            (None, 256)               196864

 dense_12 (Dense)            (None, 256)               65792

 dense_13 (Dense)            (None, 256)               65792
```

```
 dense_14 (Dense)            (None, 128)               32896

 dense_15 (Dense)            (None, 128)               16512

 dense_16 (Dense)            (None, 7)                 903

=================================================================
Total params: 378759 (1.44 MB)
Trainable params: 378759 (1.44 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

[ ]:
```python
# Ajustar rótulos usando LabelEncoder
label_encoder = LabelEncoder()
y_train_adjusted = label_encoder.fit_transform(y_train)
y_test_adjusted = label_encoder.transform(y_test)

# Compilar o modelo
model.compile(optimizer='adam',
              loss=tf.keras.losses.
 ↪SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

Treinamento por 30 épocas, com o treinamento por batelada. Onde os pesos são ajustados a cada 4 registros.

[ ]:
```python
# Treinar o modelo
model.fit(bert_model(train_encodings)[1], y_train_adjusted,
          validation_data=(bert_model(test_encodings)[1], y_test_adjusted),
          epochs=30,
          batch_size=4,
          verbose=1)
```

Avaliando o modelo BERT verificando Acurácia, Precisão, Recall e F1-score.

[ ]:
```python
# Fazer previsões nos dados de teste
y_pred = model.predict(bert_model(test_encodings)[1])
y_pred_classes = np.argmax(y_pred, axis=1)

# Calcular métricas
accuracy = accuracy_score(y_test_adjusted, y_pred_classes)
precision = precision_score(y_test_adjusted, y_pred_classes, average='weighted')
recall = recall_score(y_test_adjusted, y_pred_classes, average='weighted')
f1 = f1_score(y_test_adjusted, y_pred_classes, average='weighted')

print(f'Acurácia: {accuracy}')
print(f'Precisão: {precision}')
```

```
print(f'Recall: {recall}')
print(f'F1-score: {f1}')
```

```
5/5 [==============================] - 0s 1ms/step
5/5 [==============================] - 0s 1ms/step
Acurácia: 0.6571428571428571
Precisão: 0.70306167262689
Recall: 0.6571428571428571
F1-score: 0.6562934691750211
```

1.2 BERTimbau

Uma versão do BERT pré-treinada especificamente em textos em português do Brasil. Isso significa que ele pode capturar nuances específicas do idioma português brasileiro que o BERT genérico pode não capturar tão bem. O BERTimbal foi implementado de maneira similar ao BERT.

[ ]:
```
# Carregar o tokenizer e o modelo BERTimbal
tokenizer = BertTokenizer.from_pretrained('neuralmind/
 ↪bert-base-portuguese-cased')
bert_model = TFBertModel.from_pretrained('neuralmind/
 ↪bert-base-portuguese-cased')

# Congelar os pesos do BERT
bert_model.trainable = False
```

```
tokenizer_config.json:   0%|          | 0.00/43.0 [00:00<?, ?B/s]

vocab.txt:   0%|          | 0.00/210k [00:00<?, ?B/s]

added_tokens.json:   0%|          | 0.00/2.00 [00:00<?, ?B/s]

special_tokens_map.json:   0%|          | 0.00/112 [00:00<?, ?B/s]

config.json:   0%|          | 0.00/647 [00:00<?, ?B/s]

tf_model.h5:   0%|          | 0.00/529M [00:00<?, ?B/s]

Some layers from the model checkpoint at neuralmind/bert-base-portuguese-cased
were not used when initializing TFBertModel: ['mlm___cls']
- This IS expected if you are initializing TFBertModel from the checkpoint of a
model trained on another task or with another architecture (e.g. initializing a
BertForSequenceClassification model from a BertForPreTraining model).
- This IS NOT expected if you are initializing TFBertModel from the checkpoint
of a model that you expect to be exactly identical (initializing a
BertForSequenceClassification model from a BertForSequenceClassification model).
Some layers of TFBertModel were not initialized from the model checkpoint at
neuralmind/bert-base-portuguese-cased and are newly initialized:
['bert/pooler/dense/bias:0', 'bert/pooler/dense/kernel:0']
You should probably TRAIN this model on a down-stream task to be able to use it
for predictions and inference.
Some layers of TFBertModel were not initialized from the model checkpoint at
```

```
neuralmind/bert-base-portuguese-cased and are newly initialized:
['bert/pooler/dense/bias:0', 'bert/pooler/dense/kernel:0']
You should probably TRAIN this model on a down-stream task to be able to use it
for predictions and inference.
```

[ ]:
```
# Tokenizar os textos
max_length = 128
train_encodings = tokenizer(X_train, padding=True, truncation=True,␣
 ↪max_length=max_length, return_tensors='tf')
test_encodings = tokenizer(X_test, padding=True, truncation=True,␣
 ↪max_length=max_length, return_tensors='tf')
```

[ ]:
```
# Criando uma MLP como classificador
model = tf.keras.Sequential([
    tf.keras.layers.Dense(256, activation='relu', input_shape=(bert_model.
 ↪config.hidden_size,)),
    tf.keras.layers.Dense(256, activation='relu'),
    tf.keras.layers.Dense(256, activation='relu'),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(7)  # 7 classes de saída
])
```

```
/home/jdfv/anaconda3/envs/narnia/lib/python3.9/site-
packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential models,
prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

[ ]:
```
# Ajustar rótulos usando LabelEncoder
label_encoder = LabelEncoder()
y_train_adjusted = label_encoder.fit_transform(y_train)
y_test_adjusted = label_encoder.transform(y_test)

# Compilar o modelo
model.compile(optimizer='adam',
              loss=tf.keras.losses.
 ↪SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

Também foi feito o treinamento da MLP por 30

[ ]:
```
# Treinar o modelo
model.fit(bert_model(train_encodings)[1], y_train_adjusted,
          validation_data=(bert_model(test_encodings)[1], y_test_adjusted),
          epochs=30,
          batch_size=4,
          verbose=1)
```

```
[ ]: # Fazer previsões nos dados de teste
y_pred = model.predict(bert_model(test_encodings)[1])
y_pred_classes = np.argmax(y_pred, axis=1)

# Calcular métricas
accuracy = accuracy_score(y_test_adjusted, y_pred_classes)
precision = precision_score(y_test_adjusted, y_pred_classes, average='weighted')
recall = recall_score(y_test_adjusted, y_pred_classes, average='weighted')
f1 = f1_score(y_test_adjusted, y_pred_classes, average='weighted')

print(f'Acurácia: {accuracy}')
print(f'Precisão: {precision}')
print(f'Recall: {recall}')
print(f'F1-score: {f1}')
```

```
5/5             0s 10ms/step
Acurácia: 0.8285714285714286
Precisão: 0.8401987352188568
Recall: 0.8285714285714286
F1-score: 0.8286168369548234
```

1.3 Desclaimer

As performances do modelos poderiam ter sido ainda melhores se alguns fatores fossem considerados:

- Usar mais dados da base original. Foi utilizado apenas uma quantidade mínima dos dados (100 registros de cada uma das 7 classes consideradas). A base original tem mais de 160mil textos!!!
- Testar com classificadores mais complexos. Foi usado apenas uma MLP (nem tão profunda/complexa assim)
- Treinar por mais épocas e avaliar diferentes taxas de aprendizagem, funções de ativação e tamanho de batch...

Como o intuito era apenas comparar BERT e BERTimbal em textos em português, optou-se por seguir com os resultados alcançados, apesar dessas limitações de memória/processamento.

1.4 Comparação

Modelo	Acurácia	Precisão	Recall	F1-score
BERT	0.657	0.703	0.657	0.656
BERTimbal	0.829	0.840	0.829	0.829

Os valores indicam uma melhoria significativa em todas as métricas ao utilizar o BERTimbal em comparação com o BERT. Os modelos BERT específicos para determinadas línguas, como o BERTimbal para português, tendem a superar os modelos BERT multilínguas, especialmente quando aplicados a tarefas da determinada língua.

Isso se dá porque o BERTimbal é treinado exclusivamente no idioma português, o que permite capturar melhor as nuances linguísticas únicas ao português. Em contraste, modelos BERT multilíngues são treinados em vários idiomas, o que pode diluir a representação de características linguísticas.