notebook1-nlp-bert

November 22, 2024

1 Text Classification Using Transformer Networks (BERT)

Nombre: Rodolfo Jesús Cruz Rebollar

Matrícula: A01368326

Grupo: 101

Some initialization:

[1]:
```
import random
import torch
import numpy as np
import pandas as pd
from tqdm.notebook import tqdm

# enable tqdm in pandas
tqdm.pandas()

# set to True to use the gpu (if there is one available)
use_gpu = True

# select device
device = torch.device('cuda' if use_gpu and torch.cuda.is_available() else␣
 ↪'cpu')
print(f'device: {device.type}')

# random seed
seed = 1122

# set random seed
if seed is not None:
    print(f'random seed: {seed}')
    random.seed(seed)
    np.random.seed(seed)
    torch.manual_seed(seed)
```

```
device: cuda
random seed: 1122
```

Read the train/dev/test datasets and create a HuggingFace `Dataset` object:

[2]:
```
def read_data(filename):
    # read csv file
    df = pd.read_csv(filename, header=None).iloc[1:, ]
    # add column names
    df.columns = ['label', 'title', 'description']
    df['label'] = df['label'].astype("int")
    # make labels zero-based
    df['label'] -= 1
    # concatenate title and description, and remove backslashes
    df['text'] = df['title'] + " " + df['description']
    df['text'] = df['text'].str.replace('\\', ' ', regex=False)
    return df
```

[3]:
```
labels = open('/kaggle/input/datos-modelo-bert/classes.txt').read().splitlines()
train_df = read_data('/kaggle/input/datos-modelo-bert/train.csv')
test_df = read_data('/kaggle/input/datos-modelo-bert/test.csv')
train_df
```

[3]:
```
        label                                              title  \
1           2  Wall St. Bears Claw Back Into the Black (Reuters)
2           2  Carlyle Looks Toward Commercial Aerospace (Reu…
3           2    Oil and Economy Cloud Stocks' Outlook (Reuters)
4           2  Iraq Halts Oil Exports from Main Southern Pipe…
5           2  Oil prices soar to all-time record, posing new…
…       …                                                  …
119996      0  Pakistan's Musharraf Says Won't Quit as Army C…
119997      1                  Renteria signing a top-shelf deal
119998      1                    Saban not going to Dolphins yet
119999      1                                Today's NFL games
120000      1                       Nets get Carter from Raptors

                                              description  \
1       Reuters - Short-sellers, Wall Street's dwindli…
2       Reuters - Private investment firm Carlyle Grou…
3       Reuters - Soaring crude prices plus worries\ab…
4       Reuters - Authorities have halted oil export\f…
5       AFP - Tearaway world oil prices, toppling reco…
…                                                     …
119996   KARACHI (Reuters) - Pakistani President Perve…
119997  Red Sox general manager Theo Epstein acknowled…
119998  The Miami Dolphins will put their courtship of…
119999  PITTSBURGH at NY GIANTS Time: 1:30 p.m. Line: …
120000  INDIANAPOLIS -- All-Star Vince Carter was trad…

                                                     text
```

```
1       Wall St. Bears Claw Back Into the Black (Reute…
2       Carlyle Looks Toward Commercial Aerospace (Reu…
3       Oil and Economy Cloud Stocks' Outlook (Reuters…
4       Iraq Halts Oil Exports from Main Southern Pipe…
5       Oil prices soar to all-time record, posing new…
…                                                     …
119996  Pakistan's Musharraf Says Won't Quit as Army C…
119997  Renteria signing a top-shelf deal Red Sox gene…
119998  Saban not going to Dolphins yet The Miami Dolp…
119999  Today's NFL games PITTSBURGH at NY GIANTS Time…
120000  Nets get Carter from Raptors INDIANAPOLIS -- A…

[120000 rows x 4 columns]
```

[4]:
```
from sklearn.model_selection import train_test_split

train_df, eval_df = train_test_split(train_df, train_size=0.9)
train_df.reset_index(inplace=True, drop=True)
eval_df.reset_index(inplace=True, drop=True)

print(f'train rows: {len(train_df.index):,}')
print(f'eval rows: {len(eval_df.index):,}')
print(f'test rows: {len(test_df.index):,}')
```

```
train rows: 108,000
eval rows: 12,000
test rows: 7,600
```

[5]:
```
from datasets import Dataset, DatasetDict

ds = DatasetDict()
ds['train'] = Dataset.from_pandas(train_df)
ds['validation'] = Dataset.from_pandas(eval_df)
ds['test'] = Dataset.from_pandas(test_df)
ds
```

[5]:
```
DatasetDict({
    train: Dataset({
        features: ['label', 'title', 'description', 'text'],
        num_rows: 108000
    })
    validation: Dataset({
        features: ['label', 'title', 'description', 'text'],
        num_rows: 12000
    })
    test: Dataset({
        features: ['label', 'title', 'description', 'text'],
        num_rows: 7600
```

```
    })
})
```

Tokenize the texts:

[6]:
```
from transformers import AutoTokenizer

transformer_name = 'bert-base-cased'
tokenizer = AutoTokenizer.from_pretrained(transformer_name)
```

```
tokenizer_config.json:   0%|          | 0.00/49.0 [00:00<?, ?B/s]
```

```
config.json:   0%|          | 0.00/570 [00:00<?, ?B/s]
```

```
vocab.txt:   0%|          | 0.00/213k [00:00<?, ?B/s]
```

```
tokenizer.json:   0%|          | 0.00/436k [00:00<?, ?B/s]
```

```
/opt/conda/lib/python3.10/site-
packages/transformers/tokenization_utils_base.py:1617: FutureWarning:
`clean_up_tokenization_spaces` was not set. It will be set to `True` by default.
This behavior will be deprecated in transformers v4.45, and will be then set to
`False` by default. For more details check this issue:
https://github.com/huggingface/transformers/issues/31884
  warnings.warn(
```

[7]:
```
def tokenize(examples):
    return tokenizer(examples['text'], truncation=True)

train_ds = ds['train'].map(
    tokenize, batched=True,
    remove_columns=['title', 'description', 'text'],
)
eval_ds = ds['validation'].map(
    tokenize,
    batched=True,
    remove_columns=['title', 'description', 'text'],
)
train_ds.to_pandas()
```

```
Map:   0%|          | 0/108000 [00:00<?, ? examples/s]
```

```
Map:   0%|          | 0/12000 [00:00<?, ? examples/s]
```

[7]:

	label	input_ids	\
0	2	[101, 16752, 13335, 1186, 2101, 6690, 9717, 11…	
1	1	[101, 145, 11680, 17308, 9741, 2428, 150, 1469…	
2	2	[101, 1418, 14099, 27086, 1494, 1114, 4031, 11…	
3	1	[101, 2404, 117, 6734, 1996, 118, 1565, 5465, …	
4	3	[101, 142, 10044, 27302, 4317, 1584, 3273, 111…	
…	…	…	

```
107995      1  [101, 4922, 2274, 1654, 1112, 10503, 1505, 112…
107996      3  [101, 10605, 24632, 11252, 21285, 10221, 118, …
107997      2  [101, 13832, 3484, 11300, 4060, 5058, 112, 188…
107998      3  [101, 142, 13675, 3756, 5795, 2445, 1104, 109,…
107999      2  [101, 157, 16450, 1658, 5302, 185, 7776, 11006…

                                              token_type_ids  \
0       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
1       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
2       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
3       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
4       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
…                                                     …
107995  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
107996  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
107997  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
107998  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
107999  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …

                                              attention_mask
0       [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, …
1       [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, …
2       [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, …
3       [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, …
4       [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, …
…                                                     …
107995  [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, …
107996  [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, …
107997  [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, …
107998  [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, …
107999  [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, …

[108000 rows x 4 columns]
```

Create the transformer model:

[8]:
```python
from torch import nn
from transformers.modeling_outputs import SequenceClassifierOutput
from transformers.models.bert.modeling_bert import BertModel,␣
 ↪BertPreTrainedModel

# https://github.com/huggingface/transformers/blob/
 ↪65659a29cf5a079842e61a63d57fa24474288998/src/transformers/models/bert/
 ↪modeling_bert.py#L1486

class BertForSequenceClassification(BertPreTrainedModel):
    def __init__(self, config):
```

```
        super().__init__(config)
        self.num_labels = config.num_labels
        self.bert = BertModel(config)
        self.dropout = nn.Dropout(config.hidden_dropout_prob)
        self.classifier = nn.Linear(config.hidden_size, config.num_labels)
        self.init_weights()

    def forward(self, input_ids=None, attention_mask=None, token_type_ids=None,␣
↪labels=None, **kwargs):
        outputs = self.bert(
            input_ids,
            attention_mask=attention_mask,
            token_type_ids=token_type_ids,
            **kwargs,
        )
        cls_outputs = outputs.last_hidden_state[:, 0, :]
        cls_outputs = self.dropout(cls_outputs)
        logits = self.classifier(cls_outputs)
        loss = None
        if labels is not None:
            loss_fn = nn.CrossEntropyLoss()
            loss = loss_fn(logits, labels)
        return SequenceClassifierOutput(
            loss=loss,
            logits=logits,
            hidden_states=outputs.hidden_states,
            attentions=outputs.attentions,
        )
```

[9]:
```
from transformers import AutoConfig

config = AutoConfig.from_pretrained(
    transformer_name,
    num_labels=len(labels),
)

model = (
    BertForSequenceClassification
    .from_pretrained(transformer_name, config=config)
)
```

```
model.safetensors:   0%|          | 0.00/436M [00:00<?, ?B/s]
```

```
Some weights of BertForSequenceClassification were not initialized from the
model checkpoint at bert-base-cased and are newly initialized:
['classifier.bias', 'classifier.weight']
You should probably TRAIN this model on a down-stream task to be able to use it
for predictions and inference.
```

Create the trainer object and train:

[10]:
```
from transformers import TrainingArguments

num_epochs = 2
batch_size = 24
weight_decay = 0.01
model_name = f'{transformer_name}-sequence-classification'

training_args = TrainingArguments(
    output_dir=model_name,
    log_level='error',
    num_train_epochs=num_epochs,
    per_device_train_batch_size=batch_size,
    per_device_eval_batch_size=batch_size,
    eval_strategy='epoch',
    weight_decay=weight_decay,
)
```

[11]:
```
from sklearn.metrics import accuracy_score

def compute_metrics(eval_pred):
    y_true = eval_pred.label_ids
    y_pred = np.argmax(eval_pred.predictions, axis=-1)
    return {'accuracy': accuracy_score(y_true, y_pred)}
```

[12]:
```
from transformers import Trainer

trainer = Trainer(
    model=model,
    args=training_args,
    compute_metrics=compute_metrics,
    train_dataset=train_ds,
    eval_dataset=eval_ds,
    tokenizer=tokenizer,
)
```

[13]:
```
trainer.train()
```

```
wandb: WARNING The `run_name` is currently set to the same
value as `TrainingArguments.output_dir`. If this was not intended, please
specify a different run name by setting the `TrainingArguments.run_name`
parameter.
wandb: Using wandb-core as the SDK backend. Please refer to
https://wandb.me/wandb-core for more information.
wandb: Logging into wandb.ai. (Learn how to deploy a W&B server
locally: https://wandb.me/wandb-server)
wandb: You can find your API key in your browser here:
```

```
https://wandb.ai/authorize
wandb: Paste an API key from your profile and hit enter, or press
ctrl+c to quit:

  ........

wandb: Appending key for api.wandb.ai to your netrc file:
/root/.netrc

VBox(children=(Label(value='Waiting for wandb.init()…\r'), FloatProgress(value=0.
  ↪011112715411112124, max=1.0…

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/parallel_apply.py:79:
FutureWarning: `torch.cuda.amp.autocast(args…)` is deprecated. Please use
`torch.amp.autocast('cuda', args…)` instead.
  with torch.cuda.device(device), torch.cuda.stream(stream),
autocast(enabled=autocast_enabled):
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/_functions.py:68:
UserWarning: Was asked to gather along dimension 0, but all input tensors were
scalars; will instead unsqueeze and return a vector.
  warnings.warn('Was asked to gather along dimension 0, but all '

<IPython.core.display.HTML object>

/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/parallel_apply.py:79:
FutureWarning: `torch.cuda.amp.autocast(args…)` is deprecated. Please use
`torch.amp.autocast('cuda', args…)` instead.
  with torch.cuda.device(device), torch.cuda.stream(stream),
autocast(enabled=autocast_enabled):
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/_functions.py:68:
UserWarning: Was asked to gather along dimension 0, but all input tensors were
scalars; will instead unsqueeze and return a vector.
  warnings.warn('Was asked to gather along dimension 0, but all '
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/parallel_apply.py:79:
FutureWarning: `torch.cuda.amp.autocast(args…)` is deprecated. Please use
`torch.amp.autocast('cuda', args…)` instead.
  with torch.cuda.device(device), torch.cuda.stream(stream),
autocast(enabled=autocast_enabled):
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/_functions.py:68:
UserWarning: Was asked to gather along dimension 0, but all input tensors were
scalars; will instead unsqueeze and return a vector.
  warnings.warn('Was asked to gather along dimension 0, but all '
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/parallel_apply.py:79:
```

```
FutureWarning: `torch.cuda.amp.autocast(args…)` is deprecated. Please use
`torch.amp.autocast('cuda', args…)` instead.
  with torch.cuda.device(device), torch.cuda.stream(stream),
autocast(enabled=autocast_enabled):
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/_functions.py:68:
UserWarning: Was asked to gather along dimension 0, but all input tensors were
scalars; will instead unsqueeze and return a vector.
  warnings.warn('Was asked to gather along dimension 0, but all '
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/parallel_apply.py:79:
FutureWarning: `torch.cuda.amp.autocast(args…)` is deprecated. Please use
`torch.amp.autocast('cuda', args…)` instead.
  with torch.cuda.device(device), torch.cuda.stream(stream),
autocast(enabled=autocast_enabled):
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/_functions.py:68:
UserWarning: Was asked to gather along dimension 0, but all input tensors were
scalars; will instead unsqueeze and return a vector.
  warnings.warn('Was asked to gather along dimension 0, but all '
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/parallel_apply.py:79:
FutureWarning: `torch.cuda.amp.autocast(args…)` is deprecated. Please use
`torch.amp.autocast('cuda', args…)` instead.
  with torch.cuda.device(device), torch.cuda.stream(stream),
autocast(enabled=autocast_enabled):
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/_functions.py:68:
UserWarning: Was asked to gather along dimension 0, but all input tensors were
scalars; will instead unsqueeze and return a vector.
  warnings.warn('Was asked to gather along dimension 0, but all '
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/parallel_apply.py:79:
FutureWarning: `torch.cuda.amp.autocast(args…)` is deprecated. Please use
`torch.amp.autocast('cuda', args…)` instead.
  with torch.cuda.device(device), torch.cuda.stream(stream),
autocast(enabled=autocast_enabled):
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/_functions.py:68:
UserWarning: Was asked to gather along dimension 0, but all input tensors were
scalars; will instead unsqueeze and return a vector.
  warnings.warn('Was asked to gather along dimension 0, but all '
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/parallel_apply.py:79:
FutureWarning: `torch.cuda.amp.autocast(args…)` is deprecated. Please use
`torch.amp.autocast('cuda', args…)` instead.
  with torch.cuda.device(device), torch.cuda.stream(stream),
autocast(enabled=autocast_enabled):
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/_functions.py:68:
UserWarning: Was asked to gather along dimension 0, but all input tensors were
scalars; will instead unsqueeze and return a vector.
  warnings.warn('Was asked to gather along dimension 0, but all '
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/parallel_apply.py:79:
FutureWarning: `torch.cuda.amp.autocast(args…)` is deprecated. Please use
`torch.amp.autocast('cuda', args…)` instead.
  with torch.cuda.device(device), torch.cuda.stream(stream),
```

```
autocast(enabled=autocast_enabled):
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/_functions.py:68:
UserWarning: Was asked to gather along dimension 0, but all input tensors were
scalars; will instead unsqueeze and return a vector.
  warnings.warn('Was asked to gather along dimension 0, but all '
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/parallel_apply.py:79:
FutureWarning: `torch.cuda.amp.autocast(args…)` is deprecated. Please use
`torch.amp.autocast('cuda', args…)` instead.
  with torch.cuda.device(device), torch.cuda.stream(stream),
autocast(enabled=autocast_enabled):
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/_functions.py:68:
UserWarning: Was asked to gather along dimension 0, but all input tensors were
scalars; will instead unsqueeze and return a vector.
  warnings.warn('Was asked to gather along dimension 0, but all '
```

[13]:
```
TrainOutput(global_step=4500, training_loss=0.16151543935139975,
metrics={'train_runtime': 3367.0008, 'train_samples_per_second': 64.152,
'train_steps_per_second': 1.337, 'total_flos': 1.5600266159781888e+16,
'train_loss': 0.16151543935139975, 'epoch': 2.0})
```

Evaluate on the test partition:

[14]:
```
test_ds = ds['test'].map(
    tokenize,
    batched=True,
    remove_columns=['title', 'description', 'text'],
)
test_ds.to_pandas()
```

```
Map:   0%|          | 0/7600 [00:00<?, ? examples/s]
```

[14]:

	label	input_ids	\
0	2	[101, 11284, 1116, 1111, 157, 151, 12966, 1170…	
1	3	[101, 1109, 6398, 1110, 1212, 131, 2307, 7219,…	
2	3	[101, 148, 1183, 119, 1881, 16387, 1116, 4468,…	
3	3	[101, 11689, 15906, 6115, 12056, 1116, 1370, 2…	
4	3	[101, 11917, 8914, 119, 19294, 4206, 1106, 215…	
…	…	…	
7595	0	[101, 5596, 1103, 1362, 5284, 5200, 3234, 1384…	
7596	1	[101, 159, 7874, 1110, 2709, 1114, 13875, 1556…	
7597	1	[101, 16247, 2972, 9178, 2409, 4271, 140, 1418…	
7598	2	[101, 126, 1104, 1893, 8167, 10721, 4420, 1107…	
7599	2	[101, 142, 2064, 4164, 3370, 1154, 13519, 1116…	

	token_type_ids	\
0	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …	
1	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …	
2	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …	
3	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …	

```
4     [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
…                                                   …
7595  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
7596  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
7597  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
7598  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
7599  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …

                                          attention_mask
0     [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, …
1     [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, …
2     [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, …
3     [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, …
4     [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, …
…                                                   …
7595  [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, …
7596  [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, …
7597  [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, …
7598  [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, …
7599  [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, …

[7600 rows x 4 columns]
```

[15]:
```
output = trainer.predict(test_ds)
output
```

```
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/parallel_apply.py:79:
FutureWarning: `torch.cuda.amp.autocast(args…)` is deprecated. Please use
`torch.amp.autocast('cuda', args…)` instead.
  with torch.cuda.device(device), torch.cuda.stream(stream),
autocast(enabled=autocast_enabled):
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/_functions.py:68:
UserWarning: Was asked to gather along dimension 0, but all input tensors were
scalars; will instead unsqueeze and return a vector.
  warnings.warn('Was asked to gather along dimension 0, but all '

<IPython.core.display.HTML object>
```

[15]:
```
PredictionOutput(predictions=array([[ 0.18817817, -4.099455  ,  4.81954   ,
-1.13671   ],
       [-0.12040745, -3.5072982 , -3.5106635 ,  6.0831623 ],
       [-0.07475641, -3.3091686 , -3.8080387 ,  5.5467186 ],
       …,
       [-1.2274623 ,  7.376894  , -2.4648151 , -3.291837  ],
       [-0.06878442, -3.5864434 ,  5.744611  , -2.30391   ],
       [-3.3749778 , -3.9146588 ,  3.7081194 ,  2.250793  ]],
      dtype=float32), label_ids=array([2, 3, 3, …, 1, 2, 2]),
metrics={'test_loss': 0.17090027034282684, 'test_accuracy': 0.9472368421052632,
```

```
'test_runtime': 38.7559, 'test_samples_per_second': 196.099,
'test_steps_per_second': 4.103})
```

[16]:
```
from sklearn.metrics import classification_report

y_true = output.label_ids
y_pred = np.argmax(output.predictions, axis=-1)
target_names = labels
print(classification_report(y_true, y_pred, target_names=target_names))
```

```
              precision    recall  f1-score   support

       World       0.97      0.95      0.96      1900
      Sports       0.99      0.99      0.99      1900
    Business       0.92      0.91      0.92      1900
    Sci/Tech       0.91      0.93      0.92      1900

    accuracy                           0.95      7600
   macro avg       0.95      0.95      0.95      7600
weighted avg       0.95      0.95      0.95      7600
```

2 Explicación del pipeline ejecutado por el código

1. **Establecer la configuración inicial del entorno de ejecución:**
   - Se establece o configura el tipo de dispositivo con el que se trabajará, ya sea CPU o GPU, lo cual dependerá de cuál de los dispositivos se encuentre disponible.
   - Establecer una semilla de carácter aleatorio con el propósito de que los resultados del modelo sean reproducibles (que puedan ser replicados en futuras ejecuciones del modelo).
2. **Preparación de los datos a modelar:**
   - Los datos a modelar son importados desde archivos en formato csv y son preprocesados para remover caractéres carentes de significado semántico.
   - Se lleva a cabo una combinación de títulos y descripciones en un único texto para elevar el grado de eficiencia del procesamiento del mismo.
   - Realizar una partición de los datos en dataset de entrenamiento, validación y prueba.
3. **Generación de conjuntos de datos (datasets):**
   - Se realiza la transformación de los datos procesados en estructuras con cualidades especiales denominadas `Dataset` y `DatasetDict`, mismas que son de utilidad para operar de forma eficiente con modelos de `HuggingFace`.
4. **Tokenización:**
   - Utilización de un modelo preentrenado para transformar textos en valores numéricos comprensibles para el modelo.
   - El proceso de tokenización es aplicado a los textos pertenecientes a los datasets de entrenamiento, validación y prueba.
5. **Definir el modelo:**
   - Se emplea un modelo BERT preentrenado, mismo que se adapta para la realización de tareas de clasificación, mediante la agregación de una capa adicional para la predicción

de las labels o etiquetas.

6. **Configurar el proceso de entrenamiento:**
   - Se lleva a cabo la selección de valores para los parámetros del modelo, tales como la cantidad de épocas de entrenamiento, tamaño de lotes, además de la forma de evaluación del desempeño del modelo en el transcurso de su fase de entrenamiento.
7. **Entrenamiento del modelo:**
   - El modelo anteriormente creado es entrenado con el conjunto de datos de entrenamiento, además también se realizan evaluaciones periódicas a dicho modelo usando el dataset de validación con el objetivo principal de ajustar y optimizar su rendimiento.
8. **Evaluación final del modelo:**
   - El modelo ya entrenado ahora es puesto a prueba, empleando el conjunto de datos de prueba para obtener los valores de las métricas de desempeño del modelo, tales como el recall, precision, entre otras métricas.
   - Se lleva a cabo la generación de predicciones para el análisis de la forma en que el modelo realiza la clasificación de los datos.
9. **Obtención de resultados finales y análisis:**
   - Preparación de reportes que incluyan los valores correspondientes a las diferentes métricas de desempeño del modelo, mostrando la precisión del mismo tanto de manera general como por cada categoría.