Text Classification Using Transformer Networks (BERT)

Some initialization:

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

In [43]:
```
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
```

In [44]:
```
def read_data(filename):
    # read csv file
    df = pd.read_csv(filename, header=None)
    # add column names
    df.columns = ['label', 'title', 'description']
    # 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
```

In [45]:
```
labels = open('/kaggle/input/ag-news/classes.txt').read().splitlines()
train_df = read_data('/kaggle/input/ag-news/train.csv')
test_df = read_data('/kaggle/input/ag-news/test.csv')
train_df
```

Out[45]:

	label	title	description	text
0	2	Wall St. Bears Claw Back Into the Black (Reuters)	Reuters - Short-sellers, Wall Street's dwindli...	Wall St. Bears Claw Back Into the Black (Reute...
1	2	Carlyle Looks Toward Commercial Aerospace (Reu...	Reuters - Private investment firm Carlyle Grou...	Carlyle Looks Toward Commercial Aerospace (Reu...
2	2	Oil and Economy Cloud Stocks' Outlook (Reuters)	Reuters - Soaring crude prices plus worries\ab...	Oil and Economy Cloud Stocks' Outlook (Reuters...
3	2	Iraq Halts Oil Exports from Main Southern Pipe...	Reuters - Authorities have halted oil export\f...	Iraq Halts Oil Exports from Main Southern Pipe...
4	2	Oil prices soar to all-time record, posing new...	AFP - Tearaway world oil prices, toppling reco...	Oil prices soar to all-time record, posing new...
...	...	...	...	...
119995	0	Pakistan's Musharraf Says Won't Quit as Army C...	KARACHI (Reuters) - Pakistani President Perve...	Pakistan's Musharraf Says Won't Quit as Army C...
119996	1	Renteria signing a top-shelf deal	Red Sox general manager Theo Epstein acknowled...	Renteria signing a top-shelf deal Red Sox gene...
119997	1	Saban not going to Dolphins yet	The Miami Dolphins will put their courtship of...	Saban not going to Dolphins yet The Miami Dolp...
119998	1	Today's NFL games	PITTSBURGH at NY GIANTS Time: 1:30 p.m. Line: ...	Today's NFL games PITTSBURGH at NY GIANTS Time...
119999	1	Nets get Carter from Raptors	INDIANAPOLIS -- All-Star Vince Carter was trad...	Nets get Carter from Raptors INDIANAPOLIS -- A...

120000 rows × 4 columns

In [46]:
```
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
```

In [47]:
```
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
```

Out[47]:
```
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:

In [48]:
```
from transformers import AutoTokenizer

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

In [49]:
```
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]
```

Out[49]:

	label	input_ids	token_type_ids	attention_mask
0	2	[101, 16752, 13335, 1186, 2101, 6690, 9717, 11...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
1	1	[101, 145, 11680, 17308, 9741, 2428, 150, 1469...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
2	2	[101, 1418, 14099, 27086, 1494, 1114, 4031, 11...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
3	1	[101, 2404, 117, 6734, 1996, 118, 1565, 5465, ...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
4	3	[101, 142, 10044, 27302, 4317, 1584, 3273, 111...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
...	...	...	...	...
107995	1	[101, 4922, 2274, 1654, 1112, 10503, 1505, 112...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
107996	3	[101, 10605, 24632, 11252, 21285, 10221, 118, ...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
107997	2	[101, 13832, 3484, 11300, 4060, 5058, 112, 188...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
107998	3	[101, 142, 13675, 3756, 5795, 2445, 1104, 109,...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
107999	2	[101, 157, 16450, 1658, 5302, 185, 7776, 11006...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...

108000 rows × 4 columns

Create the transformer model:

In [50]:
```
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_b

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,
        )
```

In [51]:
```
from transformers import AutoConfig

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

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

Create the trainer object and train:

In [52]:
```
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,
)
```

In [53]:
```
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)}
```

In [54]:
```
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,
)
```

In [55]:
```
trainer.train()
```

```
/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 '
{'loss': 0.2915, 'grad_norm': 5.123822212219238, 'learning_rate': 4.4444444444444447e-05, 'epoch': 0.2222222222222222}
/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 '
{'loss': 0.2121, 'grad_norm': 3.1903417110443115, 'learning_rate': 3.888888888888889e-05, 'epoch': 0.4444444444444444}
/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 '
{'loss': 0.1916, 'grad_norm': 2.8626973628997803, 'learning_rate': 3.3333333333333335e-05, 'epoch': 0.6666666666666666}
```

```
/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 '
{'loss': 0.1876, 'grad_norm': 2.7386739253997803, 'learning_rate': 2.777777777777778e-05, 'epoch': 0.8888888888888888}
/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 '
{'eval_loss': 0.17288987338542938, 'eval_accuracy': 0.9396666666666667, 'eval_runtime': 57.592, 'eval_samples_per_second': 208.362, 'eval_steps_per_second': 4.341, 'epoch': 1.0}
/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 '
{'loss': 0.1439, 'grad_norm': 1.6634832620620728, 'learning_rate': 2.2222222222222223e-05, 'epoch': 1.1111111111111112}
/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 '
{'loss': 0.1121, 'grad_norm': 1.9883543252944946, 'learning_rate': 1.6666666666666667e-05, 'epoch': 1.3333333333333333}
/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 '
{'loss': 0.1118, 'grad_norm': 1.0829625129699707, 'learning_rate': 1.1111111111111112e-05, 'epoch': 1.5555555555555556}
/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 '
{'loss': 0.1045, 'grad_norm': 2.0572495460510254, 'learning_rate': 5.555555555555556e-06, 'epoch': 1.7777777777777777}
/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 '
{'loss': 0.1022, 'grad_norm': 1.9468250274658203, 'learning_rate': 0.0, 'epoch': 2.0}
/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 '
{'eval_loss': 0.16231119632720947, 'eval_accuracy': 0.9463333333333334, 'eval_runtime': 57.8449, 'eval_samples_per_second': 207.451, 'eval_steps_per_second': 4.322, 'epoch': 2.0}
{'train_runtime': 3117.1194, 'train_samples_per_second': 69.295, 'train_steps_per_second': 1.444, 'train_loss': 0.16191654290093316, 'epoch': 2.0}
```

Out[55]:
```
TrainOutput(global_step=4500, training_loss=0.16191654290093316, metrics={'train_runtime': 3117.1194, 'train_samples_per_second': 69.295, 'train_steps_per_second': 1.444, 'train_loss': 0.16191654290093316, 'epoch': 2.0})
```

Evaluate on the test partition:

In [56]:
```
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]
```

Out[56]:

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

7600 rows × 4 columns

In [57]:
```
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...)` i
s 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 '
```

Out[57]:
```
PredictionOutput(predictions=array([[ 0.11802129, -4.3101683 ,  5.182914  , -1.5071326 ],
       [-0.35843354, -3.5217607 , -3.2178686 ,  6.085772  ],
       [ 0.8099433 , -3.1893542 , -3.8001406 ,  5.2195983 ],
       ...,
       [-1.149165  ,  7.0233674 , -2.742669  , -3.5430453 ],
       [-1.420962  , -3.531547  ,  6.073373  , -1.7579337 ],
       [-3.2170436 , -4.254717  ,  3.9358306 ,  2.0419703 ]],
      dtype=float32), label_ids=array([2, 3, 3, ..., 1, 2, 2]), metrics={'test_loss': 0.1675737500190735, 'test_accuracy': 0.94815
78947368421, 'test_runtime': 35.1418, 'test_samples_per_second': 216.266, 'test_steps_per_second': 4.525})
```

In [58]:
```
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       0.96      0.96      0.96      1900
      Sports       0.99      0.99      0.99      1900
    Business       0.93      0.91      0.92      1900
    Sci/Tech       0.91      0.94      0.93      1900

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

El pipeline carga y preprocesa los datos de texto importados para clasificación, configura el entorno y divide los datos en los conjuntos de entrenamiento, validación y prueba. Luego convierte estos datos al formato compatible con la biblioteca "Datasets" para utilizarlos con los modelos de BERT.