# Text Classification Using Transformer Networks (BERT)

Some initialization:

In [12]:

```
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:

In [13]:

```python
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 [14]:

```python
labels = open('/kaggle/input/classes-txt/classes.txt').read().splitlines()

train_df = read_data('/kaggle/input/agnews-pytorch-simple-embed-classif-90/AG_NEWS/tra

test_df = read_data('/kaggle/input/agnews-pytorch-simple-embed-classif-90/AG_NEWS/test

train_df
```

Out[14]:

| | 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 [15]:

```python
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 [16]:

```python
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[16]:

```
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 [17]:

```
from transformers import AutoTokenizer


transformer_name = 'bert-base-cased'

tokenizer = AutoTokenizer.from_pretrained(transformer_name)
```

```
/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` b
y default. This behavior will be deprecated in transformers v4.45, and will be then s
et to `False` by default. For more details check this issue: https://github.com/huggi
ngface/transformers/issues/31884
  warnings.warn(
```

Esta parte del código usa la librería Transformers para cargar el tokenizer del modelo preentrenado de BERT: bert-base-cased. Lo que hace es que el tokenizer convierte el texto en pedacitos (tokens) que el modelo puede entender. Con AutoTokenizer.from_pretrained, se descarga el tokenizer y así se puede preparar el texto para que el modelo lo procese.

In [18]:

```
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[18]:

| | 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, 0, ... | [1, 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, 0, ... | [1, 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, 0, ... | [1, 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, 0, ... | [1, 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, 0, ... | [1, 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, 0, ... | [1, 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, 0, ... | [1, 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, 0, ... | [1, 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, 0, ... | [1, 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, 0, ... | [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ... |

108000 rows × 4 columns

Este código aplica el tokenizer al dataset y lo prepara para entrenar el modelo. La función tokenize toma como entrada ejemplos del dataset y les aplica el tokenizer para evitar que los textos sean demasiados largos. Primero, se aplica la función tokenize a todos los ejemplos en grupo del dataset de entrenamiento. Además, elimina las columnas 'title', 'description' y 'text' del dataset, dejando solo los datos necesarios después de tokenizar. Luego, se hace lo mismo para el conjunto de validación.

Create the transformer model:

In [19]:

```
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/65659a29cf5a079842e61a63d57fa244742

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
        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,
        )
```

Este código define una clase llamada **`BertForSequenceClassification`** que adapta el modelo **BERT** para clasificar texto. Toma un texto como entrada, lo procesa con **BERT** para extraer las representaciones del texto y luego utiliza una capa lineal final para clasificar el texto en cierta categoría.

Primero, en la inicialización, se configura el modelo con:

- Una instancia de `BertModel`, que es la parte que entiende el texto.
- Una capa de *dropout* para evitar que el modelo aprenda patrones que no generalizan bien.
- Una capa lineal que toma las representaciones de BERT y las convierte en las probabilidades de cada clase.

En el método `forward`, el texto pasa por BERT, se toma el primer token (`[CLS]`) como resumen de toda la secuencia, y luego se aplica el *dropout* y la capa lineal para generar las predicciones. Si se incluyen etiquetas (las respuestas correctas), también calcula la amiento.

Finalmente, devuelve un objeto que incluye las predicciones, li aplica), los estados ocultos y las atecíficas.

In [20]:

```
from transformers import AutoConfig

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

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

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

En este código, con `AutoConfig`, se obtiene la configuración del modelo preentrenado especificado por `transformer_name` y se ajusta el número de etiquetas para que coincida con la cantidad de categorías en el problema. Luego, con `BertForSequenceClassification.from_pretrained`, se carga el modelo preentrenado de **Hugging Face**, utilizando la configuraion que se definios.

Create the trainer object and train:

In [21]:

```python
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,
    evaluation_strategy='epoch',
    weight_decay=weight_decay,
)
```

```
/opt/conda/lib/python3.10/site-packages/transformers/training_args.py:1545: FutureWar
ning: `evaluation_strategy` is deprecated and will be removed in version 4.46 of 🤗 T
ransformers. Use `eval_strategy` instead
  warnings.warn(
```

En este código se define el número de épocas, el tamaño del batch y la regularización para evitar el sobreajuste.

Con `TrainingArguments`, se establecen configuraciones clave como:

- **`output_dir`** : La carpeta donde se guardarán los modelos entrenados y los resultados.
- **`num_train_epochs`** : El número de veces que el modelo verá el conjunto completo de entrenamiento (2 épocas en este caso).
- **`per_device_train_batch_size`** y **`per_device_eval_batch_size`** : El tamaño debatchlotes para entrenamiento y evaluación.
- **`evaluation_strategy`** : Se configura para evaluar el modelo al final de cada época.
- **`log_level`** : Solo se mostrarán errores durante el entrenasultados.

In [22]:

```python
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)}
```

Este código define una función llamada **`compute_metrics`** , que se encarga de calcular el accuracy de las predicciones realizadas. Toma como entrada `eval_pred` , un objeto que contiene:

- **`label_ids`** : Las etiquetas reales del conjunto de evaluación.
- **`predictions`** : Las predicciones d modelo.e.

Dentro de la función, se usa `np.argmax` para convertir las probabilidades en predicciones finales (seleccionando la clase con mayor probabilidad). Luego, compara estas predicciones con las etiquetas reales usando `accuracy_score` de `scikit-learn` , que calcula el porcentajeaciertos.

ación.

In [23]:

```
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,

)
```

Este código crea un objeto Trainer, para entrenar y evaluar modelos de manera fácil y eficiente. Aquí se integran todos los elementos necesarios para que el entrenamiento funcione correctamente:

- model: El modelo que se va a entrenar, en este caso, BERT adaptado para clasificación de secuencias.
- args: Los argumentos del entrenamiento definidos previamente en training_args, como el número de épocas, el tamaño de los lotes, y cómo manejar la evaluación.
- compute_metrics: Una función que calcula métricas durante la evaluación. Aquí se utiliza compute_metrics para calcular la precisión (accuracy).
- train_dataset y eval_dataset: Los datos de entrenamiento (train_ds) y evaluación (eval_ds) que el modelo utilizará para aprender y validarse.
- tokenizer: El tokenizer que transforma el texto en tokens, asegurando que los datos se procesen correctamente antes de entrar al modelo.

Con este Trainer, puedes entrenar el modelo, evaluarlo y ajustar sus parámetros automáticamente, simplificando mucho el proceso y evitando errores manuales.

In [24]:

```
trainer.train()
```

```
wandb: WARNING The `run_name` is currently set to the same value as `TrainingArgument
s.output_dir`. If this was not intended, please specify a different run name by setti
ng the `TrainingArguments.run_name` parameter.
wandb: Using wandb-core as the SDK backend. Please refer to https://wandb.me/wandb-co
re for more information.
wandb: Logging into wandb.ai. (Learn how to deploy a W&B server locally: https://wand
b.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=0.01
1112841188888625, max=1.0…
```

Tracking run with wandb version 0.18.3

Run data is saved locally in `/kaggle/working/wandb/run-20241124_071032-60mbfb0g`

Syncing run **bert-base-cased-sequence-classification** to Weights & Biases (docs)

View project at https://wandb.ai/mansoor35/huggingface

View run at https://wandb.ai/mansoor35/huggingface/runs/60mbfb0g

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

[4500/4500 55:38, Epoch 2/2]

| Epoch | Training Loss | Validation Loss | Accuracy |
|---:|---:|---:|---:|
| 1 | 0.189400 | 0.170879 | 0.941833 |
| 2 | 0.102300 | 0.163020 | 0.946250 |

```
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/parallel_apply.py:79: Futur
eWarning: `torch.cuda.amp.autocast(args...)` is deprecated. Please use `torch.amp.aut
ocast('cuda', args...)` instead.
  with torch.cuda.device(device), torch.cuda.stream(stream), autocast(enabled=autocas
t_enabled):
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/_functions.py:68: UserWarni
ng: Was asked to gather along dimension 0, but all input tensors were scalars; will i
nstead 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: Futur
eWarning: `torch.cuda.amp.autocast(args...)` is deprecated. Please use `torch.amp.aut
ocast('cuda', args...)` instead.
  with torch.cuda.device(device), torch.cuda.stream(stream), autocast(enabled=autocas
t_enabled):
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/_functions.py:68: UserWarni
ng: Was asked to gather along dimension 0, but all input tensors were scalars; will i
nstead 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: Futur
eWarning: `torch.cuda.amp.autocast(args...)` is deprecated. Please use `torch.amp.aut
ocast('cuda', args...)` instead.
  with torch.cuda.device(device), torch.cuda.stream(stream), autocast(enabled=autocas
t_enabled):
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/_functions.py:68: UserWarni
ng: Was asked to gather along dimension 0, but all input tensors were scalars; will i
nstead 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: Futur
eWarning: `torch.cuda.amp.autocast(args...)` is deprecated. Please use `torch.amp.aut
ocast('cuda', args...)` instead.
  with torch.cuda.device(device), torch.cuda.stream(stream), autocast(enabled=autocas
t_enabled):
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/_functions.py:68: UserWarni
ng: Was asked to gather along dimension 0, but all input tensors were scalars; will i
nstead 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: Futur
eWarning: `torch.cuda.amp.autocast(args...)` is deprecated. Please use `torch.amp.aut
ocast('cuda', args...)` instead.
  with torch.cuda.device(device), torch.cuda.stream(stream), autocast(enabled=autocas
t_enabled):
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/_functions.py:68: UserWarni
ng: Was asked to gather along dimension 0, but all input tensors were scalars; will i
nstead 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: Futur
eWarning: `torch.cuda.amp.autocast(args...)` is deprecated. Please use `torch.amp.aut
ocast('cuda', args...)` instead.
  with torch.cuda.device(device), torch.cuda.stream(stream), autocast(enabled=autocas
t_enabled):
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/_functions.py:68: UserWarni
ng: Was asked to gather along dimension 0, but all input tensors were scalars; will i
nstead 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: Futur
eWarning: `torch.cuda.amp.autocast(args...)` is deprecated. Please use `torch.amp.aut
ocast('cuda', args...)` instead.
  with torch.cuda.device(device), torch.cuda.stream(stream), autocast(enabled=autocas
t_enabled):
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/_functions.py:68: UserWarni
```

```
ng: Was asked to gather along dimension 0, but all input tensors were scalars; will i
nstead 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: Futur
eWarning: `torch.cuda.amp.autocast(args...)` is deprecated. Please use `torch.amp.aut
ocast('cuda', args...)` instead.
  with torch.cuda.device(device), torch.cuda.stream(stream), autocast(enabled=autocas
t_enabled):
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/_functions.py:68: UserWarni
ng: Was asked to gather along dimension 0, but all input tensors were scalars; will i
nstead 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: Futur
eWarning: `torch.cuda.amp.autocast(args...)` is deprecated. Please use `torch.amp.aut
ocast('cuda', args...)` instead.
  with torch.cuda.device(device), torch.cuda.stream(stream), autocast(enabled=autocas
t_enabled):
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/_functions.py:68: UserWarni
ng: Was asked to gather along dimension 0, but all input tensors were scalars; will i
nstead 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: Futur
eWarning: `torch.cuda.amp.autocast(args...)` is deprecated. Please use `torch.amp.aut
ocast('cuda', args...)` instead.
  with torch.cuda.device(device), torch.cuda.stream(stream), autocast(enabled=autocas
t_enabled):
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/_functions.py:68: UserWarni
ng: Was asked to gather along dimension 0, but all input tensors were scalars; will i
nstead unsqueeze and return a vector.
  warnings.warn('Was asked to gather along dimension 0, but all '
```

Out[24]:

```
TrainOutput(global_step=4500, training_loss=0.16224037170410155, metrics={'train_runt
ime': 3348.6705, 'train_samples_per_second': 64.503, 'train_steps_per_second': 1.344,
'total_flos': 1.5600315493990656e+16, 'train_loss': 0.16224037170410155, 'epoch': 2.
0})
```

En esta parte del código se entrena el modelo.

Evaluate on the test partition:

In [25]:

```
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[25]:

| | 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, 0, ... | [1, 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, 0, ... | [1, 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, 0, ... | [1, 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, 0, ... | [1, 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, 0, ... | [1, 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, 0, ... | [1, 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, 0, ... | [1, 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, 0, ... | [1, 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, 0, ... | [1, 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, 0, ... | [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ... |

7600 rows × 4 columns

Se aplica el mismo proceso al conjunto de prueba.

In [26]:

```
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 '
```

Out[26]:
```
PredictionOutput(predictions=array([[ 0.36316976, -4.0587187 ,  4.708022  , -1.248963
2 ],
       [ 0.23322392, -3.313729  , -3.7166765 ,  5.9260798 ],
       [ 0.5275277 , -3.135042  , -3.806155  ,  5.569708  ],
       ...,
       [-1.4069046 ,  7.2616873 , -2.2133346 , -3.5286772 ],
       [-0.6832599 , -3.6432836 ,  6.008366  , -2.3655388 ],
       [-3.0042052 , -3.8196225 ,  3.4429522 ,  2.3862102 ]],
      dtype=float32), label_ids=array([2, 3, 3, ..., 1, 2, 2]), metrics={'test_loss':
0.17101630568504333, 'test_accuracy': 0.95, 'test_runtime': 37.9558, 'test_samples_pe
r_second': 200.233, 'test_steps_per_second': 4.189})
```

Se predicen los resultados.

In [27]:
```
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      0.96      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
```

Se muestra el desempeño del modelo.