Text Classification Using Transformer Networks (BERT)

Some initialization:

In [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:

In [2]:
```
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 [3]:
```
labels = open('/kaggle/input/clases/classes.txt').read().splitlines()
train_df = read_data('https://raw.githubusercontent.com/mhjabreel/CharCnn_Keras/
test_df = read_data('https://raw.githubusercontent.com/mhjabreel/CharCnn_Keras/r
train_df
```

Out[3]:

	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 [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
```

In [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
```

Out[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:

In [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(
```

In [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]
```

Out[7]:

	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

Create the transformer model:

In [8]:
```
from torch import nn
from transformers.modeling_outputs import SequenceClassifierOutput
from transformers.models.bert.modeling_bert import BertModel, BertPreTrainedMode

# https://github.com/huggingface/transformers/blob/65659a29cf5a079842e61a63d57fa

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,
        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 [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:

In [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,
    evaluation_strategy='epoch',
    weight_decay=weight_decay,
)
```

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

In [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)}
```

In [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,
)
```

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

```
wandb: WARNING The `run_name` is currently set to the same value as `TrainingArgu
ments.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/wand
b-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=
0.011112625022229218, max=1.0…
```

Tracking run with wandb version 0.18.3

Run data is saved locally in `/kaggle/working/wandb/run-20241114_185717-9blul3sq`

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

View project at https://wandb.ai/a00834710-tecnol-gico-de-monterrey/huggingface

View run at https://wandb.ai/a00834710-tecnol-gico-de-monterrey/huggingface/runs/9blul3sq

```
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/parallel_apply.py:79: F
utureWarning: `torch.cuda.amp.autocast(args...)` is deprecated. Please use `torc
h.amp.autocast('cuda', args...)` instead.
  with torch.cuda.device(device), torch.cuda.stream(stream), autocast(enabled=aut
ocast_enabled):
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/_functions.py:68: UserW
arning: Was asked to gather along dimension 0, but all input tensors were scalar
s; will instead unsqueeze and return a vector.
  warnings.warn('Was asked to gather along dimension 0, but all '
```

[4500/4500 52:49, Epoch 2/2]

Epoch	Training Loss	Validation Loss	Accuracy
1	0.187500	0.172834	0.941750
2	0.101100	0.163638	0.946917

```
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/parallel_apply.py:79: F
utureWarning: `torch.cuda.amp.autocast(args...)` is deprecated. Please use `torc
h.amp.autocast('cuda', args...)` instead.
  with torch.cuda.device(device), torch.cuda.stream(stream), autocast(enabled=aut
ocast_enabled):
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/_functions.py:68: UserW
arning: Was asked to gather along dimension 0, but all input tensors were scalar
s; 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: F
utureWarning: `torch.cuda.amp.autocast(args...)` is deprecated. Please use `torc
h.amp.autocast('cuda', args...)` instead.
  with torch.cuda.device(device), torch.cuda.stream(stream), autocast(enabled=aut
ocast_enabled):
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/_functions.py:68: UserW
arning: Was asked to gather along dimension 0, but all input tensors were scalar
s; 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: F
utureWarning: `torch.cuda.amp.autocast(args...)` is deprecated. Please use `torc
h.amp.autocast('cuda', args...)` instead.
  with torch.cuda.device(device), torch.cuda.stream(stream), autocast(enabled=aut
ocast_enabled):
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/_functions.py:68: UserW
arning: Was asked to gather along dimension 0, but all input tensors were scalar
s; 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: F
utureWarning: `torch.cuda.amp.autocast(args...)` is deprecated. Please use `torc
h.amp.autocast('cuda', args...)` instead.
  with torch.cuda.device(device), torch.cuda.stream(stream), autocast(enabled=aut
ocast_enabled):
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/_functions.py:68: UserW
arning: Was asked to gather along dimension 0, but all input tensors were scalar
s; 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: F
utureWarning: `torch.cuda.amp.autocast(args...)` is deprecated. Please use `torc
h.amp.autocast('cuda', args...)` instead.
  with torch.cuda.device(device), torch.cuda.stream(stream), autocast(enabled=aut
ocast_enabled):
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/_functions.py:68: UserW
arning: Was asked to gather along dimension 0, but all input tensors were scalar
s; 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: F
utureWarning: `torch.cuda.amp.autocast(args...)` is deprecated. Please use `torc
h.amp.autocast('cuda', args...)` instead.
  with torch.cuda.device(device), torch.cuda.stream(stream), autocast(enabled=aut
ocast_enabled):
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/_functions.py:68: UserW
arning: Was asked to gather along dimension 0, but all input tensors were scalar
s; 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: F
utureWarning: `torch.cuda.amp.autocast(args...)` is deprecated. Please use `torc
h.amp.autocast('cuda', args...)` instead.
  with torch.cuda.device(device), torch.cuda.stream(stream), autocast(enabled=aut
ocast_enabled):
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/_functions.py:68: UserW
```

```
arning: Was asked to gather along dimension 0, but all input tensors were scalar
s; 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: F
utureWarning: `torch.cuda.amp.autocast(args...)` is deprecated. Please use `torc
h.amp.autocast('cuda', args...)` instead.
  with torch.cuda.device(device), torch.cuda.stream(stream), autocast(enabled=aut
ocast_enabled):
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/_functions.py:68: UserW
arning: Was asked to gather along dimension 0, but all input tensors were scalar
s; 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: F
utureWarning: `torch.cuda.amp.autocast(args...)` is deprecated. Please use `torc
h.amp.autocast('cuda', args...)` instead.
  with torch.cuda.device(device), torch.cuda.stream(stream), autocast(enabled=aut
ocast_enabled):
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/_functions.py:68: UserW
arning: Was asked to gather along dimension 0, but all input tensors were scalar
s; 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: F
utureWarning: `torch.cuda.amp.autocast(args...)` is deprecated. Please use `torc
h.amp.autocast('cuda', args...)` instead.
  with torch.cuda.device(device), torch.cuda.stream(stream), autocast(enabled=aut
ocast_enabled):
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/_functions.py:68: UserW
arning: Was asked to gather along dimension 0, but all input tensors were scalar
s; will instead unsqueeze and return a vector.
  warnings.warn('Was asked to gather along dimension 0, but all '
```

Out[13]:
```
TrainOutput(global_step=4500, training_loss=0.16159833441840277, metrics={'trai
n_runtime': 3450.3436, 'train_samples_per_second': 62.602, 'train_steps_per_sec
ond': 1.304, 'total_flos': 1.5600315493990656e+16, 'train_loss': 0.161598334418
40277, 'epoch': 2.0})
```

Evaluate on the test partition:

In [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]
```

Out[14]:

	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

In [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 '
```

Out[15]:
```
PredictionOutput(predictions=array([[ 0.24020289, -3.984492  ,  4.6809845 , -1.
0258693 ],
       [ 0.22886965, -3.4685717 , -3.6386886 ,  6.016186  ],
       [ 1.3360738 , -3.2784834 , -4.329989  ,  5.341624  ],
       ...,
       [-1.6345811 ,  7.7355776 , -2.2592394 , -3.5820985 ],
       [-0.72235173, -3.4981773 ,  5.6760693 , -2.0300438 ],
       [-3.5574112 , -3.8130646 ,  4.21603   ,  2.0997121 ]],
      dtype=float32), label_ids=array([2, 3, 3, ..., 1, 2, 2]), metrics={'test_
loss': 0.16880503296852112, 'test_accuracy': 0.9484210526315789, 'test_runtim
e': 36.3347, 'test_samples_per_second': 209.166, 'test_steps_per_second': 4.37
6})
```

In [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       0.96      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
```

BERT supera a un clasificador de regresión logística con embeddings GloVe porque captura el contexto bidireccional de las palabras, es adaptable a tareas específicas mediante ajuste fino, modela relaciones complejas entre palabras con su arquitectura de transformadores, y se preentrena en grandes corpus para entender mejor el lenguaje. Además, su arquitectura profunda es más poderosa y efectiva para problemas complejos en comparación con los modelos lineales y embeddings estáticos.

Pipeline de Clasificación de Texto con Redes Neuronales Basadas en Transformadores

Carga y Preprocesamiento de Datos:

- Se utiliza un conjunto de datos con columnas de texto (como `title`, `description`, `text`) y etiquetas (`label`).
- Se aplica un mapeo para tokenizar los textos utilizando un tokenizador preentrenado, generando las secuencias `input_ids`, `token_type_ids` y `attention_mask`.
- Las columnas originales de texto se eliminan tras la tokenización, dejando únicamente las características necesarias para el modelo.

Configuración del Modelo:

- Se emplea un modelo pree**BERT** (o similar) ajustado para tareas de clasificación secuencial.
- El modelo se configura con un número de etiquetas igual al de las clases presentes en los datos.

Entrenamiento del Modelo:

- Se define un objeto `Trainer` de **Hugging Face**, que encapsula el modelo, el optimizador y los parámetros de entrenamiento.
- El entrenamiento se realiza utilizando un conjunto de entrenamiento con validación cruzada, supervisando métricas como la pérdida de entrenamiento y validación.
- Durante el proceso, se emplean herramientas como **Weights & Biases (WandB)** para rastrear el progreso del entrenamiento, incluyendo métricas como precisión, pérdida y velocidad de procesamiento.

Evaluación del Modelo:

- El modelo entrenado se evalúa sobre un conjunto de prueba, donde se aplican las mismas transformaciones de tokenización utilizadas durante el entrenamiento.
- El resultado incluye predicciones de probabilidad para cada clase, etiquetas verdaderas y métricas de desempeño (precisión, recall, F1-score).

Análisis de Desempeño:

- Se genera un reporte de clasificación utilizando métricas como **precisión**, **recall** y **F1-score** para cada clase.
- El desempeño global se presenta a través de medidas como el **promedio ponderado** y el **macro promedio**.

Visualización y Seguimiento:

- **WandB** rastrea el desempeño del modelo y visualiza las métricas clave, como la pérdida y precisión durante las épocas de entrenamiento.
- También se proporciona un enlace a los reportes de *asta la evaluación del modelo.

Comparación entre BERT y Regresión Logística con Embeddings GloVe en Clasificación de Texto

La comparación entre los dos enfoques, un clasificador de regresión logística con embeddings GloVe y la implementación basada en PyTorch de BERT, demuestra que el modelo basado en BERT supera al clasificador de regresión logística con embeddings GloVe en la tarea de clasificación de texto.

Desempeño del modelo basado en BERT

El modelo ajustado con BERT (implementado con transformers de HuggingFace) logra:

- **Precisión general**: 95% en el conjunto de prueba.
- **Métricas F1**: cercanas al 0.95 en promedio.

BERT utiliza embeddings contextuales y ajusta su arquitectura de transformador bidireccional para modelar relaciones complejas y específicas del problema, lo que mejora su capacidad de generalización.

Desempeño del clasificador de regresión logística con GloVe

El modelo de regresión logística implementado en PyTorch obtiene métricas de desempeño más bajas:

- **Métricas F1**: varían entre 0.81 y 0.94 según las clases.
- Desempeño más bajoen cl escomo "Words" y "Business"

El uso de embeddings GloVe proporciona representaciones estáticas, lo que limita la capacidad del modelo para captar el contexto dependiente de palabras en oraciones.

Conclusión

La superioridad de **BERT** radica en:

1. **Preentrenamiento**: en grandes corpus de datos, lo que mejora la comprensión del lenguaje.
2. **Embeddings contextuales**: que modelan relaciones complejas entre palabras.
3. **Ajuste fino**: adaptabilidad para tareas específicas.
4. **Arquitectura**: transformador bidireccional más poderoso frente a relaciones lineales de GloVe.

Podemos decir que para tareas complejas de clasificación de texto, un enfoque basado en transformadores como **BERT** es más efectivo y adaptable que modelos lineales combinados con embeddings preentrenados como **GloVe**.

Por lo tanto, queda validado que el mejor desempeño en esta tarea se obtiene utilizando la implementación basada en **BERT**.