In [3]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from transformers import DistilBertTokenizer, DistilBertForSequenceClassification
from torch.utils.data import DataLoader, Dataset
from torch.nn.utils.rnn import pad_sequence
import torch
from tqdm import tqdm

# Load the dataset
data = pd.read_csv("bbc.csv")  # Replace "your_dataset.csv" with the actual file path
data = data.dropna()  # Drop rows with missing values

# Split the dataset into training and testing sets
train_data, test_data = train_test_split(data, test_size=0.2, random_state=42)

# Tokenizer
tokenizer = DistilBertTokenizer.from_pretrained("distilbert-base-uncased")

# Encode the text data
train_encodings = tokenizer(train_data["text"].tolist(), truncation=True, padding=True, max_leng
test_encodings = tokenizer(test_data["text"].tolist(), truncation=True, padding=True, max_length

# PyTorch Dataset
class TextDataset(Dataset):
    def __init__(self, encodings, labels):
        self.encodings = encodings
        self.labels = labels

    def __getitem__(self, idx):
        item = {key: torch.tensor(val[idx]) for key, val in self.encodings.items()}
        item["labels"] = torch.tensor(self.labels[idx])
        return item

    def __len__(self):
        return len(self.labels)

train_dataset = TextDataset(train_encodings, train_data["category"].astype("category").cat.codes
test_dataset = TextDataset(test_encodings, test_data["category"].astype("category").cat.codes.to

# DataLoader
train_loader = DataLoader(train_dataset, batch_size=16, shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=16, shuffle=False)

# Model
model = DistilBertForSequenceClassification.from_pretrained("distilbert-base-uncased", num_label

# Optimizer and Loss function
optimizer = torch.optim.AdamW(model.parameters(), lr=5e-5)
loss_fn = torch.nn.CrossEntropyLoss()

# Training
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model.to(device)

num_epochs = 3
for epoch in range(num_epochs):
    model.train()
    for batch in tqdm(train_loader):
        optimizer.zero_grad()
        inputs = {key: val.to(device) for key, val in batch.items() if key != "labels"}
        labels = batch["labels"].to(device)
        outputs = model(**inputs)
        loss = loss_fn(outputs.logits, labels)
        loss.backward()
        optimizer.step()

# Evaluation
model.eval()
all_preds = []
all_labels = []
with torch.no_grad():
    for batch in tqdm(test_loader):
        inputs = {key: val.to(device) for key, val in batch.items() if key != "labels"}
        labels = batch["labels"].to(device)
        outputs = model(**inputs)
        preds = torch.argmax(outputs.logits, dim=1).cpu().numpy()
```

```
        all_preds.extend(preds)
        all_labels.extend(labels.cpu().numpy())

# Calculate accuracy
accuracy = accuracy_score(all_labels, all_preds)
print(f"Accuracy: {accuracy}")
```

```
Some weights of DistilBertForSequenceClassification were not initialized from the model checkpoin
t at distilbert-base-uncased and are newly initialized: ['classifier.bias', 'classifier.weight',
'pre_classifier.bias', 'pre_classifier.weight']
You should probably TRAIN this model on a down-stream task to be able to use it for predictions a
nd inference.
100%|██████████| 112/112 [0
0:07<00:00, 14.56it/s]
100%|██████████| 112/112 [0
0:07<00:00, 15.85it/s]
100%|██████████| 112/112 [0
0:07<00:00, 15.88it/s]
100%|██████████| 28/28 [0
0:00<00:00, 50.44it/s]
Accuracy: 0.9775280898876404
```

In [6]:
```
df_results = pd.DataFrame({
    'True_Labels': all_labels,
    'Predicted_Labels': all_preds
})
df_results
```

Out[6]:

	True_Labels	Predicted_Labels
0	2	2
1	0	0
2	1	1
3	4	4
4	3	3
...	...	...
440	0	0
441	0	0
442	0	0
443	0	0
444	0	0

445 rows × 2 columns