```python
from sklearn.datasets import fetch_20newsgroups
from sklearn.model_selection import train_test_split

# load dataset
newsgroups = fetch_20newsgroups(subset='all', remove=('headers', 'footers', 'quotes'))

texts = newsgroups.data
labels = newsgroups.target
label_names = newsgroups.target_names

# split into train and test
train_texts, test_texts, train_labels, test_labels = train_test_split(
    texts, labels, test_size=0.2, random_state=42)
```

```python
from transformers import AutoTokenizer

# load tokenizer
tokenizer = AutoTokenizer.from_pretrained("distilbert-base-uncased")

# tokenize text data
train_encodings = tokenizer(train_texts, truncation=True, padding=True, max_length=512)
test_encodings = tokenizer(test_texts, truncation=True, padding=True, max_length=512)
```

```python
import torch

class NewsGroupDataset(torch.utils.data.Dataset):
    def __init__(self, encodings, labels):
        self.encodings = encodings
        self.labels = labels

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

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

train_dataset = NewsGroupDataset(train_encodings, train_labels)
test_dataset = NewsGroupDataset(test_encodings, test_labels)
```

```python
from transformers import AutoModelForSequenceClassification

model = AutoModelForSequenceClassification.from_pretrained(
    "distilbert-base-uncased", num_labels=20)
```

```
WARNING:torchao.kernel.intmm:Warning: Detected no triton, on systems without Triton certain kernels will not work
model.safetensors: 100%                                    268M/268M [00:02<00:00, 219MB/s]
Some weights of DistilBertForSequenceClassification were not initialized from the model checkpoint at distilbert-base-uncase
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
```

```python
from transformers import TrainingArguments, Trainer
import numpy as np
from sklearn.metrics import accuracy_score

def compute_metrics(p):
    preds = np.argmax(p.predictions, axis=1)
    return {"accuracy": accuracy_score(p.label_ids, preds)}

training_args = TrainingArguments(
    output_dir="./results",
    do_train=True,
    do_eval=True,
    learning_rate=2e-5,
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    num_train_epochs=3,
    weight_decay=0.01,
    logging_dir="./logs",
    logging_steps=100,
    save_strategy="epoch",
    eval_steps=500
)
```

```
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=test_dataset,
    compute_metrics=compute_metrics
)
```

```
text = "The government passed a new law affecting international trade."

inputs = tokenizer(text, return_tensors="pt", truncation=True, padding=True, max_length=512)
outputs = model(**inputs)
predicted_class = outputs.logits.argmax().item()

print(f"Predicted Topic: {label_names[predicted_class]}")
```

```
Predicted Topic: talk.politics.misc
```

```
Start coding or generate with AI.
```