```
import pandas as pd
import torch
from torch.utils.data import Dataset
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from transformers import DistilBertTokenizerFast, DistilBertForSequenceClassifi
```

```
!pip install -U transformers
```

```
Requirement already satisfied: transformers in /usr/local/lib/python3.11/di
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: huggingface-hub<1.0,>=0.30.0 in /usr/local/l
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.11/dis
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.11/dis
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: tokenizers<0.22,>=0.21 in /usr/local/lib/pyt
Requirement already satisfied: safetensors>=0.4.3 in /usr/local/lib/python3
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.11/dist
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.1
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib
Requirement already satisfied: hf-xet<2.0.0,>=1.1.2 in /usr/local/lib/pytho
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/p
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/di
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3
```

```
import os
```

```
!pip install -U transformers datasets
os.environ["WANDB_DISABLED"] = "true"
```

```
Requirement already satisfied: transformers in /usr/local/lib/python3.11/di
Requirement already satisfied: datasets in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: huggingface-hub<1.0,>=0.30.0 in /usr/local/l
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.11/dis
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.11/dis
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: tokenizers<0.22,>=0.21 in /usr/local/lib/pyt
Requirement already satisfied: safetensors>=0.4.3 in /usr/local/lib/python3
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.11/dist
Requirement already satisfied: pyarrow>=15.0.0 in /usr/local/lib/python3.11
Requirement already satisfied: dill<0.3.9,>=0.3.0 in /usr/local/lib/python3
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: xxhash in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: multiprocess<0.70.17 in /usr/local/lib/pytho
Requirement already satisfied: fsspec<=2025.3.0,>=2023.1.0 in /usr/local/li
Requirement already satisfied: aiohttp!=4.0.0a0,!=4.0.0a1 in /usr/local/lib
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib
Requirement already satisfied: hf-xet<2.0.0,>=1.1.2 in /usr/local/lib/pytho
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/p
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/di
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/pyt
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/di
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in /usr/local/lib/py
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.1
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.11/d
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python
Requirement already satisfied: propcache>=0.2.0 in /usr/local/lib/python3.1
Requirement already satisfied: yarl<2.0,>=1.17.0 in /usr/local/lib/python3.
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-p
```

```python
# Define all file names and labels
files = {
    "twitter_parsed_dataset.csv": "twitter",
    "twitter_racism_parsed_dataset.csv": "racism",
    "twitter_sexism_parsed_dataset.csv": "sexism",
    "youtube_parsed_dataset.csv": "youtube"
}

all_dfs = []

for file, label in files.items():
    df = pd.read_csv(file)
    text_col = None
    for col in df.columns:
        if 'text' in col.lower():
            text_col = col
            break
    if not text_col:
        raise ValueError(f"Couldn't find a text column in {file}")
    df = df[[text_col]].dropna().copy()
    df.columns = ['text']
    df['label'] = label
    all_dfs.append(df)

final_df = pd.concat(all_dfs).reset_index(drop=True)
final_df.head()
```

	text	label
0	@halalflaws @biebervalue @greenlinerzjm I read...	twitter
1	@ShreyaBafna3 Now you idiots claim that people...	twitter
2	RT @Mooseoftorment Call me sexist, but when I ...	twitter
3	@g0ssipsquirrelx Wrong, ISIS follows the examp...	twitter
4	#mkr No No No No No No	twitter

Next steps: Generate code with `final_df` | View recommended plots | New interactive sheet

```
# Sample 2000 entries per label to balance the dataset
sample_df = final_df.groupby('label').apply(lambda x: x.sample(n=2000, random_s

# Encode labels into integers
le = LabelEncoder()
sample_df['label_encoded'] = le.fit_transform(sample_df['label'])

# Check results
sample_df[['label', 'label_encoded']].drop_duplicates()
```

```
/tmp/ipython-input-6-3204026891.py:2: DeprecationWarning: DataFrameGroupBy.
  sample_df = final_df.groupby('label').apply(lambda x: x.sample(n=2000, ra
```

	label	label_encoded
0	racism	0
2000	sexism	1
4000	twitter	2
6000	youtube	3

```
# Split into train and validation sets (90/10 split)
from sklearn.model_selection import train_test_split

train_texts, val_texts, train_labels, val_labels = train_test_split(
    sample_df['text'].tolist(),
    sample_df['label_encoded'].tolist(),
    test_size=0.1,
    stratify=sample_df['label_encoded'],
    random_state=42
)
```

```
# Load tokenizer
tokenizer = DistilBertTokenizerFast.from_pretrained('distilbert-base-uncased')

# Tokenize train and validation sets
train_encodings = tokenizer(train_texts, truncation=True, padding=True, max_ler
val_encodings = tokenizer(val_texts, truncation=True, padding=True, max_length=
```

```
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_auth.py:94: U
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings t
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access 
  warnings.warn(
tokenizer_config.json: 100%                     48.0/48.0 [00:00<00:00, 4.96kB/s]
vocab.txt: 100%                                 232k/232k [00:00<00:00, 2.98MB/s]
tokenizer.json: 100%                            466k/466k [00:00<00:00, 25.9MB/s]
config.json: 100%                               483/483 [00:00<00:00, 31.6kB/s]
```

```
!pip install -q datasets
```

```
from datasets import Dataset

# Rebuild train and validation into HF-compatible Dataset format
train_dict = {
    'text': train_texts,
    'label': train_labels
}

val_dict = {
    'text': val_texts,
    'label': val_labels
}

train_dataset = Dataset.from_dict(train_dict)
val_dataset = Dataset.from_dict(val_dict)
```

```
def tokenize_fn(example):
    return tokenizer(example['text'], truncation=True, padding='max_length', ma

# Tokenize datasets
train_dataset = train_dataset.map(tokenize_fn, batched=True)
val_dataset = val_dataset.map(tokenize_fn, batched=True)

# Rename label column for Trainer compatibility
train_dataset = train_dataset.rename_column("label", "labels")
val_dataset = val_dataset.rename_column("label", "labels")

# Set format to PyTorch
train_dataset.set_format(type='torch', columns=['input_ids', 'attention_mask',
val_dataset.set_format(type='torch', columns=['input_ids', 'attention_mask', 'l
```

```
Map: 100%                                  7200/7200 [00:02<00:00, 3456.91 examples/s]

Map: 100%                                    800/800 [00:00<00:00, 1063.56 examples/s]
```

```
from transformers import DistilBertForSequenceClassification

# Number of unique labels (classes)
num_labels = len(set(train_labels))

# Load pretrained DistilBERT model with classification head
model = DistilBertForSequenceClassification.from_pretrained(
    'distilbert-base-uncased',
    num_labels=num_labels
)
```

```
model.safetensors: 100%                         268M/268M [00:05<00:00, 35.8MB/s]
Some weights of DistilBertForSequenceClassification were not initialized fr
You should probably TRAIN this model on a down-stream task to be able to us
```

```
from transformers import TrainingArguments
training_args = TrainingArguments(
    output_dir='./results',
    num_train_epochs=2,
    per_device_train_batch_size=16,
    per_device_eval_batch_size=64,
    warmup_steps=100,
    weight_decay=0.01,
    logging_dir='./logs',
    logging_steps=50
)
```

```
Using the `WANDB_DISABLED` environment variable is deprecated and will be r
```

Start coding or generate with AI.

```
from transformers import DistilBertForSequenceClassification
num_labels = len(set(train_labels))  # or len(set(sample_df['label_encoded']))
model = DistilBertForSequenceClassification.from_pretrained(
    'distilbert-base-uncased',
    num_labels=num_labels
)
```

```
Some weights of DistilBertForSequenceClassification were not initialized fr
You should probably TRAIN this model on a down-stream task to be able to us
```

```
from transformers import Trainer, EvalPrediction
import numpy as np
from sklearn.metrics import accuracy_score
def compute_metrics(p: EvalPrediction):
    preds = np.argmax(p.predictions, axis=1)
    return {"accuracy": accuracy_score(p.label_ids, preds)}

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=val_dataset,
    compute_metrics=compute_metrics
)
trainer.train()
```

[900/900 02:55, Epoch 2/2]

Step	Training Loss
50	1.307200
100	0.924600
150	0.891700
200	0.863800
250	0.901700
300	0.884000
350	0.863300
400	0.858300
450	0.861300
500	0.845500
550	0.793800
600	0.800400
650	0.840100
700	0.841000
750	0.788100
800	0.829700
850	0.773100
900	0.821300

```
TrainOutput(global_step=900, training_loss=0.8716072675916884, metrics=
{'train_runtime': 177.7276, 'train_samples_per_second': 81.023,
'train_steps_per_second': 5.064, 'total_flos': 476899644211200.0,
```

```
# Evaluate model on validation set
trainer.evaluate()
```

[13/13 00:02]

```
{'eval_loss': 0.8418189883232117,
 'eval_accuracy': 0.49625,
 'eval_runtime': 2.4966,
 'eval_samples_per_second': 320.439,
 'eval_steps_per_second': 5.207,
 'epoch': 2.0}
```

```
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
label_encoder.fit(final_df['label'])
```

```
▾ LabelEncoder ⓘ ?

LabelEncoder()
```

```
%reset -f
```

```
import pandas as pd
import torch
from torch.utils.data import Dataset
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from transformers import DistilBertTokenizerFast, DistilBertForSequenceClassifi
```

```
!pip install -U transformers
```

```
Requirement already satisfied: transformers in /usr/local/lib/python3.11/di
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: huggingface-hub<1.0,>=0.30.0 in /usr/local/l
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.11/dis
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.11/dis
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: tokenizers<0.22,>=0.21 in /usr/local/lib/pyt
Requirement already satisfied: safetensors>=0.4.3 in /usr/local/lib/python3
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.11/dist
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.1
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib
Requirement already satisfied: hf-xet<2.0.0,>=1.1.2 in /usr/local/lib/pytho
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/p
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/di
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3
```

```
import os
```

```
!pip install -U transformers datasets
os.environ["WANDB_DISABLED"] = "true"
```

```
Requirement already satisfied: transformers in /usr/local/lib/python3.11/di
Requirement already satisfied: datasets in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: huggingface-hub<1.0,>=0.30.0 in /usr/local/l
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.11/dis
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.11/dis
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: tokenizers<0.22,>=0.21 in /usr/local/lib/pyt
Requirement already satisfied: safetensors>=0.4.3 in /usr/local/lib/python3
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.11/dist
Requirement already satisfied: pyarrow>=15.0.0 in /usr/local/lib/python3.11
Requirement already satisfied: dill<0.3.9,>=0.3.0 in /usr/local/lib/python3
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: xxhash in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: multiprocess<0.70.17 in /usr/local/lib/pytho
Requirement already satisfied: fsspec<=2025.3.0,>=2023.1.0 in /usr/local/li
Requirement already satisfied: aiohttp!=4.0.0a0,!=4.0.0a1 in /usr/local/lib
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib
Requirement already satisfied: hf-xet<2.0.0,>=1.1.2 in /usr/local/lib/pytho
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/p
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/di
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/pyt
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/di
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in /usr/local/lib/py
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.1
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.11/d
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python
Requirement already satisfied: propcache>=0.2.0 in /usr/local/lib/python3.1
Requirement already satisfied: yarl<2.0,>=1.17.0 in /usr/local/lib/python3.
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-p
```

```
# Define all file names and labels
files = {
    "toxicity_parsed_dataset.csv": "toxicity",
    "twitter_racism_parsed_dataset.csv": "racism",
    "twitter_sexism_parsed_dataset.csv": "sexism",
    "youtube_parsed_dataset.csv": "youtube"
}

all_dfs = []

for file, label in files.items():
    df = pd.read_csv(file)
    text_col = None
    for col in df.columns:
        if 'text' in col.lower():
            text_col = col
            break
    if not text_col:
        raise ValueError(f"Couldn't find a text column in {file}")
    df = df[[text_col]].dropna().copy()
    df.columns = ['text']
    df['label'] = label
    all_dfs.append(df)

final_df = pd.concat(all_dfs).reset_index(drop=True)
final_df.head()
```

	text	label
0	This: :One can make an analogy in mathematical...	toxicity
1	` :Clarification for you (and Zundark's righ...	toxicity
2	Elected or Electoral? JHK	toxicity
3	`This is such a fun entry. Devotchka I once...	toxicity
4	Please relate the ozone hole to increases in c...	toxicity

```python
from sklearn.utils import resample
import pandas as pd

# Get the smallest class count
min_count = 1970

# Balanced list
balanced = []

# Loop through each label and resample
for label in final_df["label"].unique():
    label_df = final_df[final_df["label"] == label]
    if len(label_df) > min_count:
        resampled = resample(label_df, replace=False, n_samples=min_count, ranc
    else:
        resampled = resample(label_df, replace=True, n_samples=min_count, randc
    balanced.append(resampled)

# Concatenate all balanced samples
balanced_df = pd.concat(balanced).sample(frac=1, random_state=42).reset_index(c

# Check new distribution
print("New Label Distribution:")
print(balanced_df["label"].value_counts())
```

```
New Label Distribution:
label
toxicity    1970
racism      1970
youtube     1970
sexism      1970
Name: count, dtype: int64
```

```python
# New label encoding
label_map = {'toxicity': 0, 'racism': 1, 'youtube': 2, 'sexism': 3}
balanced_df['label_encoded'] = balanced_df['label'].map(label_map)
```

```python
import re

def clean(text):
    text = re.sub(r"http\S+", "", text)
    text = re.sub(r"[^A-Za-z\s]", "", text)
    text = re.sub(r"\s+", " ", text).strip()
    return text.lower()

balanced_df["clean_text"] = balanced_df["text"].apply(clean)
```

```
# Split into train and validation sets (90/10 split)
from sklearn.model_selection import train_test_split

train_texts, val_texts, train_labels, val_labels = train_test_split(
    balanced_df['text'].tolist(),
    balanced_df['label_encoded'].tolist(),
    test_size=0.1,
    stratify=balanced_df['label_encoded'],
    random_state=42
)
```

```
# Load tokenizer
tokenizer = DistilBertTokenizerFast.from_pretrained('distilbert-base-uncased')

# Tokenize train and validation sets
train_encodings = tokenizer(train_texts, truncation=True, padding=True, max_len
val_encodings = tokenizer(val_texts, truncation=True, padding=True, max_length=
```

```
!pip install -q datasets
```

```
from datasets import Dataset

# Rebuild train and validation into HF-compatible Dataset format
train_dict = {
    'text': train_texts,
    'label': train_labels
}

val_dict = {
    'text': val_texts,
    'label': val_labels
}

train_dataset = Dataset.from_dict(train_dict)
val_dataset = Dataset.from_dict(val_dict)
```

```
def tokenize_fn(example):
    return tokenizer(example['text'], truncation=True, padding='max_length', ma

# Tokenize datasets
train_dataset = train_dataset.map(tokenize_fn, batched=True)
val_dataset = val_dataset.map(tokenize_fn, batched=True)

# Rename label column for Trainer compatibility
train_dataset = train_dataset.rename_column("label", "labels")
val_dataset = val_dataset.rename_column("label", "labels")

# Set format to PyTorch
train_dataset.set_format(type='torch', columns=['input_ids', 'attention_mask',
val_dataset.set_format(type='torch', columns=['input_ids', 'attention_mask', 'l
```

```
Map: 100%                                    7092/7092 [00:07<00:00, 1001.17 examples/s]

Map: 100%                                      788/788 [00:00<00:00, 1085.10 examples/s]
```

```
from transformers import DistilBertForSequenceClassification# Number of unique
```

```
from transformers import TrainingArguments
training_args = TrainingArguments(
    output_dir='./results',
    num_train_epochs=4,
    per_device_train_batch_size=16,
    per_device_eval_batch_size=32,
    warmup_steps=100,
    weight_decay=0.01,
    logging_dir='./logs',
    logging_steps=10,
    save_strategy="no"
)
```

```
Using the `WANDB_DISABLED` environment variable is deprecated and will be r
```

```
from sklearn.metrics import accuracy_score, f1_score

def compute_metrics(p):
    preds = p.predictions.argmax(axis=1)
    labels = p.label_ids
    return {
        "accuracy": accuracy_score(labels, preds),
        "macro_f1": f1_score(labels, preds, average='macro')
    }

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


trainer.train()
```

```
[1776/1776 05:03, Epoch 4/4]

Step  Training Loss
  10       1.385100
  20       1.364300
  30       1.327100
  40       1.223100
  50       0.994100
  60       0.809600
  70       0.646800
  80       0.663200
  90       0.529100
 100       0.487500
 110       0.541300
 120       0.418200
 130       0.528400
 140       0.559200
 150       0.568900
```

```
160    0.464700
170    0.503300
180    0.413600
190    0.444100
200    0.472500
210    0.485800
220    0.476300
230    0.487500
240    0.530000
250    0.494500
260    0.457700
270    0.458100
280    0.548000
290    0.442800
300    0.464800
310    0.391400
320    0.385600
330    0.403600
340    0.375400
350    0.387600
360    0.339500
370    0.433800
380    0.444800
390    0.357400
400    0.343300
410    0.469400
420    0.415500
430    0.429200
440    0.426900
450    0.430200
```

450	0.430200
460	0.369700
470	0.514500
480	0.384200
490	0.357500
500	0.388300
510	0.346100
520	0.334900
530	0.459600
540	0.311800
550	0.403900
560	0.359700
570	0.441800
580	0.374200
590	0.330600
600	0.364700
610	0.456600
620	0.341200
630	0.393100
640	0.305000
650	0.384800
660	0.315300
670	0.370200
680	0.438600
690	0.398600
700	0.332300
710	0.366400
720	0.340000
730	0.311000
740	0.407600

750	0.402300
760	0.335500
770	0.414600
780	0.369000
790	0.360500
800	0.357800
810	0.418000
820	0.299000
830	0.304600
840	0.330100
850	0.399200
860	0.284600
870	0.354400
880	0.324400
890	0.349500
900	0.295200
910	0.340800
920	0.262300
930	0.328200
940	0.291300
950	0.279500
960	0.279400
970	0.319400
980	0.393900
990	0.288400
1000	0.278900
1010	0.314600
1020	0.316000
1030	0.294200
1040	0.368900

1050	0.263600
1060	0.342300
1070	0.306200
1080	0.243600
1090	0.274900
1100	0.260500
1110	0.333600
1120	0.316500
1130	0.315200
1140	0.301100
1150	0.341300
1160	0.294800
1170	0.255600
1180	0.278500
1190	0.369800
1200	0.267800
1210	0.367100
1220	0.358900
1230	0.302500
1240	0.314200
1250	0.299300
1260	0.257600
1270	0.323100
1280	0.277600
1290	0.258200
1300	0.249000
1310	0.393500
1320	0.276900
1330	0.300600

```
1340    0.227100
1350    0.216200
1360    0.187100
1370    0.284300
1380    0.195600
1390    0.214200
1400    0.282100
1410    0.240400
1420    0.260700
1430    0.215900
1440    0.226400
1450    0.237300
1460    0.273100
1470    0.229000
1480    0.203000
1490    0.238300
1500    0.294900
1510    0.265400
1520    0.226600
1530    0.187200
1540    0.231000
1550    0.248700
1560    0.181200
1570    0.314200
1580    0.246600
1590    0.193900
1600    0.237500
1610    0.344900
1620    0.239900
1630    0.178000
```

```
1640    0.224700
1650    0.247200
1660    0.225000
1670    0.183300
1680    0.206900
1690    0.203400
1700    0.200900
1710    0.233900
1720    0.204400
1730    0.195600
1740    0.203000
1750    0.241100
1760    0.259200
1770    0.215400
```

```
TrainOutput(global_step=1776, training_loss=0.3667051704885723, metrics=
{'train_runtime': 303.3996, 'train_samples_per_second': 93.5,
'train_steps_per_second': 5.854, 'total_flos': 939492299096064.0
```

```
# Evaluate model on validation set
trainer.evaluate()
```

```
[25/25 00:02]
{'eval_loss': 0.6470827460289001,
 'eval_accuracy': 0.7373096446700508,
 'eval_macro_f1': 0.7378934462747743,
 'eval_runtime': 2.8971,
 'eval_samples_per_second': 271.999,
 'eval_steps_per_second': 8.629,
 'epoch': 4.0}
```

```
from transformers import DistilBertTokenizer
tokenizer = DistilBertTokenizer.from_pretrained('distilbert-base-uncased')

val_encodings = tokenizer(
    val_texts,
    padding="max_length",
    truncation=True,
    max_length=128,
    return_tensors="pt"
)
```

```
import torch

val_labels_tensor = torch.tensor(val_labels)
```

```
from torch.utils.data import Dataset

class ValDataset(Dataset):
    def _init_(self, encodings, labels):
        self.encodings = encodings
        self.labels = labels

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

    def _getitem_(self, idx):
        return {
            'input_ids': self.encodings['input_ids'][idx],
            'attention_mask': self.encodings['attention_mask'][idx],
            'label': self.labels[idx]
        }
```

```
from torch.utils.data import DataLoader

val_loader = DataLoader(val_dataset, batch_size=32)
```

```
import torch

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print("Using device:", device)
```

```
Using device: cuda
```

```
all_preds = []
all_labels = []

model.to(device)
model.eval()

with torch.no_grad():
    for batch in val_loader:
        input_ids = batch['input_ids'].to(device)
        attention_mask = batch['attention_mask'].to(device)


        label_key = 'label' if 'label' in batch else 'labels'
        labels = batch[label_key].to(device)

        outputs = model(input_ids=input_ids, attention_mask=attention_mask)
        logits = outputs.logits
        preds = torch.argmax(logits, dim=1)

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

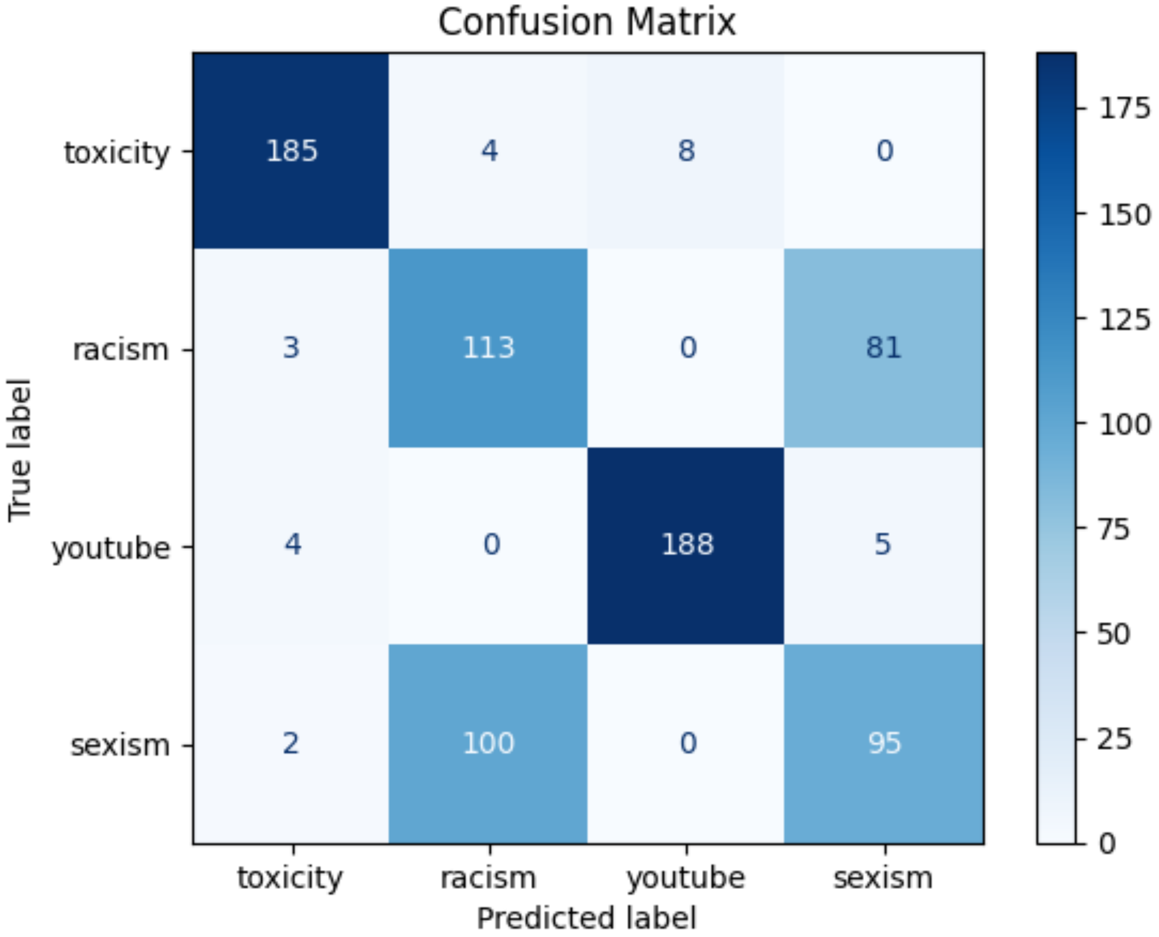
```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

# Define class names (edit if needed)
label_names = ['toxicity', 'racism', 'youtube', 'sexism']

# Generate the confusion matrix
cm = confusion_matrix(all_labels, all_preds)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=label_names)

# Plot
plt.figure(figsize=(8, 6))
disp.plot(cmap='Blues', values_format='d')
plt.title("Confusion Matrix")
plt.show()
```

```
<Figure size 800x600 with 0 Axes>
```



```
from sklearn.metrics import classification_report

print("Classification Report:")
print(classification_report(all_labels, all_preds, target_names=label_names))
```

```
Classification Report:
              precision    recall  f1-score   support

    toxicity       0.95      0.94      0.95       197
      racism       0.52      0.57      0.55       197
     youtube       0.96      0.95      0.96       197
      sexism       0.52      0.48      0.50       197

    accuracy                           0.74       788
   macro avg       0.74      0.74      0.74       788
weighted avg       0.74      0.74      0.74       788
```