```
!pip install datasets
```

```
Requirement already satisfied: datasets in
/usr/local/lib/python3.11/dist-packages (2.14.4)
Requirement already satisfied: numpy>=1.17 in
/usr/local/lib/python3.11/dist-packages (from datasets) (2.0.2)
Requirement already satisfied: pyarrow>=8.0.0 in
/usr/local/lib/python3.11/dist-packages (from datasets) (18.1.0)
Requirement already satisfied: dill<0.3.8,>=0.3.0 in
/usr/local/lib/python3.11/dist-packages (from datasets) (0.3.7)
Requirement already satisfied: pandas in
/usr/local/lib/python3.11/dist-packages (from datasets) (2.2.2)
Requirement already satisfied: requests>=2.19.0 in
/usr/local/lib/python3.11/dist-packages (from datasets) (2.32.3)
Requirement already satisfied: tqdm>=4.62.1 in
/usr/local/lib/python3.11/dist-packages (from datasets) (4.67.1)
Requirement already satisfied: xxhash in
/usr/local/lib/python3.11/dist-packages (from datasets) (3.5.0)
Requirement already satisfied: multiprocess in
/usr/local/lib/python3.11/dist-packages (from datasets) (0.70.15)
Requirement already satisfied: fsspec>=2021.11.1 in
/usr/local/lib/python3.11/dist-packages (from fsspec[http]>=2021.11.1-
>datasets) (2025.3.2)
Requirement already satisfied: aiohttp in
/usr/local/lib/python3.11/dist-packages (from datasets) (3.11.15)
Requirement already satisfied: huggingface-hub<1.0.0,>=0.14.0 in
/usr/local/lib/python3.11/dist-packages (from datasets) (0.31.4)
Requirement already satisfied: packaging in
/usr/local/lib/python3.11/dist-packages (from datasets) (24.2)
Requirement already satisfied: pyyaml>=5.1 in
/usr/local/lib/python3.11/dist-packages (from datasets) (6.0.2)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in
/usr/local/lib/python3.11/dist-packages (from aiohttp->datasets)
(2.6.1)
Requirement already satisfied: aiosignal>=1.1.2 in
/usr/local/lib/python3.11/dist-packages (from aiohttp->datasets)
(1.3.2)
Requirement already satisfied: attrs>=17.3.0 in
/usr/local/lib/python3.11/dist-packages (from aiohttp->datasets)
(25.3.0)
Requirement already satisfied: frozenlist>=1.1.1 in
/usr/local/lib/python3.11/dist-packages (from aiohttp->datasets)
(1.6.0)
Requirement already satisfied: multidict<7.0,>=4.5 in
/usr/local/lib/python3.11/dist-packages (from aiohttp->datasets)
(6.4.4)
Requirement already satisfied: propcache>=0.2.0 in
/usr/local/lib/python3.11/dist-packages (from aiohttp->datasets)
(0.3.1)
Requirement already satisfied: yarl<2.0,>=1.17.0 in
```

```
/usr/local/lib/python3.11/dist-packages (from aiohttp->datasets)
(1.20.0)
Requirement already satisfied: filelock in
/usr/local/lib/python3.11/dist-packages (from huggingface-
hub<1.0.0,>=0.14.0->datasets) (3.18.0)
Requirement already satisfied: typing-extensions>=3.7.4.3 in
/usr/local/lib/python3.11/dist-packages (from huggingface-
hub<1.0.0,>=0.14.0->datasets) (4.13.2)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.11/dist-packages (from requests>=2.19.0-
>datasets) (3.4.2)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.11/dist-packages (from requests>=2.19.0-
>datasets) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.11/dist-packages (from requests>=2.19.0-
>datasets) (2.4.0)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.11/dist-packages (from requests>=2.19.0-
>datasets) (2025.4.26)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.11/dist-packages (from pandas->datasets)
(2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in
/usr/local/lib/python3.11/dist-packages (from pandas->datasets)
(2025.2)
Requirement already satisfied: tzdata>=2022.7 in
/usr/local/lib/python3.11/dist-packages (from pandas->datasets)
(2025.2)
Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2-
>pandas->datasets) (1.17.0)
```

```
!pip install bertviz
```

```
Collecting bertviz
  Downloading bertviz-1.4.0-py3-none-any.whl.metadata (19 kB)
Requirement already satisfied: transformers>=2.0 in
/usr/local/lib/python3.11/dist-packages (from bertviz) (4.52.2)
Requirement already satisfied: torch>=1.0 in
/usr/local/lib/python3.11/dist-packages (from bertviz) (2.6.0+cu124)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-
packages (from bertviz) (4.67.1)
Collecting boto3 (from bertviz)
  Downloading boto3-1.38.25-py3-none-any.whl.metadata (6.6 kB)
Requirement already satisfied: requests in
/usr/local/lib/python3.11/dist-packages (from bertviz) (2.32.3)
Requirement already satisfied: regex in
/usr/local/lib/python3.11/dist-packages (from bertviz) (2024.11.6)
Requirement already satisfied: sentencepiece in
```

```
/usr/local/lib/python3.11/dist-packages (from bertviz) (0.2.0)
Requirement already satisfied: filelock in
/usr/local/lib/python3.11/dist-packages (from torch>=1.0->bertviz)
(3.18.0)
Requirement already satisfied: typing-extensions>=4.10.0 in
/usr/local/lib/python3.11/dist-packages (from torch>=1.0->bertviz)
(4.13.2)
Requirement already satisfied: networkx in
/usr/local/lib/python3.11/dist-packages (from torch>=1.0->bertviz)
(3.4.2)
Requirement already satisfied: jinja2 in
/usr/local/lib/python3.11/dist-packages (from torch>=1.0->bertviz)
(3.1.6)
Requirement already satisfied: fsspec in
/usr/local/lib/python3.11/dist-packages (from torch>=1.0->bertviz)
(2025.3.2)
Collecting nvidia-cuda-nvrtc-cu12==12.4.127 (from torch>=1.0->bertviz)
  Downloading nvidia_cuda_nvrtc_cu12-12.4.127-py3-none-
manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cuda-runtime-cu12==12.4.127 (from torch>=1.0-
>bertviz)
  Downloading nvidia_cuda_runtime_cu12-12.4.127-py3-none-
manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cuda-cupti-cu12==12.4.127 (from torch>=1.0->bertviz)
  Downloading nvidia_cuda_cupti_cu12-12.4.127-py3-none-
manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cudnn-cu12==9.1.0.70 (from torch>=1.0->bertviz)
  Downloading nvidia_cudnn_cu12-9.1.0.70-py3-none-
manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cublas-cu12==12.4.5.8 (from torch>=1.0->bertviz)
  Downloading nvidia_cublas_cu12-12.4.5.8-py3-none-
manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cufft-cu12==11.2.1.3 (from torch>=1.0->bertviz)
  Downloading nvidia_cufft_cu12-11.2.1.3-py3-none-
manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-curand-cu12==10.3.5.147 (from torch>=1.0->bertviz)
  Downloading nvidia_curand_cu12-10.3.5.147-py3-none-
manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cusolver-cu12==11.6.1.9 (from torch>=1.0->bertviz)
  Downloading nvidia_cusolver_cu12-11.6.1.9-py3-none-
manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cusparse-cu12==12.3.1.170 (from torch>=1.0->bertviz)
  Downloading nvidia_cusparse_cu12-12.3.1.170-py3-none-
manylinux2014_x86_64.whl.metadata (1.6 kB)
Requirement already satisfied: nvidia-cusparselt-cu12==0.6.2 in
/usr/local/lib/python3.11/dist-packages (from torch>=1.0->bertviz)
(0.6.2)
Requirement already satisfied: nvidia-nccl-cu12==2.21.5 in
/usr/local/lib/python3.11/dist-packages (from torch>=1.0->bertviz)
```

```
(2.21.5)
Requirement already satisfied: nvidia-nvtx-cu12==12.4.127 in
/usr/local/lib/python3.11/dist-packages (from torch>=1.0->bertviz)
(12.4.127)
Collecting nvidia-nvjitlink-cu12==12.4.127 (from torch>=1.0->bertviz)
  Downloading nvidia_nvjitlink_cu12-12.4.127-py3-none-
manylinux2014_x86_64.whl.metadata (1.5 kB)
Requirement already satisfied: triton==3.2.0 in
/usr/local/lib/python3.11/dist-packages (from torch>=1.0->bertviz)
(3.2.0)
Requirement already satisfied: sympy==1.13.1 in
/usr/local/lib/python3.11/dist-packages (from torch>=1.0->bertviz)
(1.13.1)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in
/usr/local/lib/python3.11/dist-packages (from sympy==1.13.1-
>torch>=1.0->bertviz) (1.3.0)
Requirement already satisfied: huggingface-hub<1.0,>=0.30.0 in
/usr/local/lib/python3.11/dist-packages (from transformers>=2.0-
>bertviz) (0.31.4)
Requirement already satisfied: numpy>=1.17 in
/usr/local/lib/python3.11/dist-packages (from transformers>=2.0-
>bertviz) (2.0.2)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.11/dist-packages (from transformers>=2.0-
>bertviz) (24.2)
Requirement already satisfied: pyyaml>=5.1 in
/usr/local/lib/python3.11/dist-packages (from transformers>=2.0-
>bertviz) (6.0.2)
Requirement already satisfied: tokenizers<0.22,>=0.21 in
/usr/local/lib/python3.11/dist-packages (from transformers>=2.0-
>bertviz) (0.21.1)
Requirement already satisfied: safetensors>=0.4.3 in
/usr/local/lib/python3.11/dist-packages (from transformers>=2.0-
>bertviz) (0.5.3)
Collecting botocore<1.39.0,>=1.38.25 (from boto3->bertviz)
  Downloading botocore-1.38.25-py3-none-any.whl.metadata (5.7 kB)
Collecting jmespath<2.0.0,>=0.7.1 (from boto3->bertviz)
  Downloading jmespath-1.0.1-py3-none-any.whl.metadata (7.6 kB)
Collecting s3transfer<0.14.0,>=0.13.0 (from boto3->bertviz)
  Downloading s3transfer-0.13.0-py3-none-any.whl.metadata (1.7 kB)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.11/dist-packages (from requests->bertviz)
(3.4.2)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.11/dist-packages (from requests->bertviz)
(3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.11/dist-packages (from requests->bertviz)
(2.4.0)
```

```
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.11/dist-packages (from requests->bertviz)
(2025.4.26)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in
/usr/local/lib/python3.11/dist-packages (from
botocore<1.39.0,>=1.38.25->boto3->bertviz) (2.9.0.post0)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.11/dist-packages (from jinja2->torch>=1.0-
>bertviz) (3.0.2)
Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.11/dist-packages (from python-
dateutil<3.0.0,>=2.1->botocore<1.39.0,>=1.38.25->boto3->bertviz)
(1.17.0)
Downloading bertviz-1.4.0-py3-none-any.whl (157 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 157.6/157.6 kB 12.8 MB/s eta
0:00:00
anylinux2014_x86_64.whl (363.4 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 363.4/363.4 MB 4.6 MB/s eta
0:00:00
anylinux2014_x86_64.whl (13.8 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.8/13.8 MB 47.5 MB/s eta
0:00:00
anylinux2014_x86_64.whl (24.6 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 24.6/24.6 MB 40.9 MB/s eta
0:00:00
e_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (883 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 883.7/883.7 kB 48.7 MB/s eta
0:00:00
anylinux2014_x86_64.whl (664.8 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 664.8/664.8 MB 796.3 kB/s eta
0:00:00
anylinux2014_x86_64.whl (211.5 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 211.5/211.5 MB 5.5 MB/s eta
0:00:00
anylinux2014_x86_64.whl (56.3 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 56.3/56.3 MB 14.5 MB/s eta
0:00:00
anylinux2014_x86_64.whl (127.9 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 127.9/127.9 MB 7.8 MB/s eta
0:00:00
anylinux2014_x86_64.whl (207.5 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 207.5/207.5 MB 6.4 MB/s eta
0:00:00
anylinux2014_x86_64.whl (21.1 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 21.1/21.1 MB 39.4 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 139.9/139.9 kB 13.2 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.6/13.6 MB 61.7 MB/s eta
```

```
0:00:00
espath-1.0.1-py3-none-any.whl (20 kB)
Downloading s3transfer-0.13.0-py3-none-any.whl (85 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 85.2/85.2 kB 7.2 MB/s eta
0:00:00
e-cu12, nvidia-cuda-nvrtc-cu12, nvidia-cuda-cupti-cu12, nvidia-cublas-
cu12, jmespath, nvidia-cusparse-cu12, nvidia-cudnn-cu12, botocore,
s3transfer, nvidia-cusolver-cu12, boto3, bertviz
  Attempting uninstall: nvidia-nvjitlink-cu12
    Found existing installation: nvidia-nvjitlink-cu12 12.5.82
    Uninstalling nvidia-nvjitlink-cu12-12.5.82:
      Successfully uninstalled nvidia-nvjitlink-cu12-12.5.82
  Attempting uninstall: nvidia-curand-cu12
    Found existing installation: nvidia-curand-cu12 10.3.6.82
    Uninstalling nvidia-curand-cu12-10.3.6.82:
      Successfully uninstalled nvidia-curand-cu12-10.3.6.82
  Attempting uninstall: nvidia-cufft-cu12
    Found existing installation: nvidia-cufft-cu12 11.2.3.61
    Uninstalling nvidia-cufft-cu12-11.2.3.61:
      Successfully uninstalled nvidia-cufft-cu12-11.2.3.61
  Attempting uninstall: nvidia-cuda-runtime-cu12
    Found existing installation: nvidia-cuda-runtime-cu12 12.5.82
    Uninstalling nvidia-cuda-runtime-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-runtime-cu12-12.5.82
  Attempting uninstall: nvidia-cuda-nvrtc-cu12
    Found existing installation: nvidia-cuda-nvrtc-cu12 12.5.82
    Uninstalling nvidia-cuda-nvrtc-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-nvrtc-cu12-12.5.82
  Attempting uninstall: nvidia-cuda-cupti-cu12
    Found existing installation: nvidia-cuda-cupti-cu12 12.5.82
    Uninstalling nvidia-cuda-cupti-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-cupti-cu12-12.5.82
  Attempting uninstall: nvidia-cublas-cu12
    Found existing installation: nvidia-cublas-cu12 12.5.3.2
    Uninstalling nvidia-cublas-cu12-12.5.3.2:
      Successfully uninstalled nvidia-cublas-cu12-12.5.3.2
  Attempting uninstall: nvidia-cusparse-cu12
    Found existing installation: nvidia-cusparse-cu12 12.5.1.3
    Uninstalling nvidia-cusparse-cu12-12.5.1.3:
      Successfully uninstalled nvidia-cusparse-cu12-12.5.1.3
  Attempting uninstall: nvidia-cudnn-cu12
    Found existing installation: nvidia-cudnn-cu12 9.3.0.75
    Uninstalling nvidia-cudnn-cu12-9.3.0.75:
      Successfully uninstalled nvidia-cudnn-cu12-9.3.0.75
  Attempting uninstall: nvidia-cusolver-cu12
    Found existing installation: nvidia-cusolver-cu12 11.6.3.83
    Uninstalling nvidia-cusolver-cu12-11.6.3.83:
      Successfully uninstalled nvidia-cusolver-cu12-11.6.3.83
Successfully installed bertviz-1.4.0 boto3-1.38.25 botocore-1.38.25
```

```
jmespath-1.0.1 nvidia-cublas-cu12-12.4.5.8 nvidia-cuda-cupti-cu12-
12.4.127 nvidia-cuda-nvrtc-cu12-12.4.127 nvidia-cuda-runtime-cu12-
12.4.127 nvidia-cudnn-cu12-9.1.0.70 nvidia-cufft-cu12-11.2.1.3 nvidia-
curand-cu12-10.3.5.147 nvidia-cusolver-cu12-11.6.1.9 nvidia-cusparse-
cu12-12.3.1.170 nvidia-nvjitlink-cu12-12.4.127 s3transfer-0.13.0
```

```
!pip install lime
```

```
Collecting lime
  Downloading lime-0.2.0.1.tar.gz (275 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0.0/275.7 kB ? eta -:--:--
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 275.7/275.7 kB 21.0 MB/s eta
0:00:00
etadata (setup.py) ... ent already satisfied: matplotlib in
/usr/local/lib/python3.11/dist-packages (from lime) (3.10.0)
Requirement already satisfied: numpy in
/usr/local/lib/python3.11/dist-packages (from lime) (2.0.2)
Requirement already satisfied: scipy in
/usr/local/lib/python3.11/dist-packages (from lime) (1.15.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-
packages (from lime) (4.67.1)
Requirement already satisfied: scikit-learn>=0.18 in
/usr/local/lib/python3.11/dist-packages (from lime) (1.6.1)
Requirement already satisfied: scikit-image>=0.12 in
/usr/local/lib/python3.11/dist-packages (from lime) (0.25.2)
Requirement already satisfied: networkx>=3.0 in
/usr/local/lib/python3.11/dist-packages (from scikit-image>=0.12-
>lime) (3.4.2)
Requirement already satisfied: pillow>=10.1 in
/usr/local/lib/python3.11/dist-packages (from scikit-image>=0.12-
>lime) (11.2.1)
Requirement already satisfied: imageio!=2.35.0,>=2.33 in
/usr/local/lib/python3.11/dist-packages (from scikit-image>=0.12-
>lime) (2.37.0)
Requirement already satisfied: tifffile>=2022.8.12 in
/usr/local/lib/python3.11/dist-packages (from scikit-image>=0.12-
>lime) (2025.5.21)
Requirement already satisfied: packaging>=21 in
/usr/local/lib/python3.11/dist-packages (from scikit-image>=0.12-
>lime) (24.2)
Requirement already satisfied: lazy-loader>=0.4 in
/usr/local/lib/python3.11/dist-packages (from scikit-image>=0.12-
>lime) (0.4)
Requirement already satisfied: joblib>=1.2.0 in
/usr/local/lib/python3.11/dist-packages (from scikit-learn>=0.18-
>lime) (1.5.0)
Requirement already satisfied: threadpoolctl>=3.1.0 in
/usr/local/lib/python3.11/dist-packages (from scikit-learn>=0.18-
>lime) (3.6.0)
Requirement already satisfied: contourpy>=1.0.1 in
```

```
/usr/local/lib/python3.11/dist-packages (from matplotlib->lime)
(1.3.2)
Requirement already satisfied: cycler>=0.10 in
/usr/local/lib/python3.11/dist-packages (from matplotlib->lime)
(0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.11/dist-packages (from matplotlib->lime)
(4.58.0)
Requirement already satisfied: kiwisolver>=1.3.1 in
/usr/local/lib/python3.11/dist-packages (from matplotlib->lime)
(1.4.8)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.11/dist-packages (from matplotlib->lime)
(3.2.3)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.11/dist-packages (from matplotlib->lime)
(2.9.0.post0)
Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.7-
>matplotlib->lime) (1.17.0)
Building wheels for collected packages: lime
  Building wheel for lime (setup.py) ... e: filename=lime-0.2.0.1-py3-
none-any.whl size=283834
sha256=27596adcb1f4caeb058ab255043909cdb6575fdd92f8ab87fd8b929f87ec34c
1
  Stored in directory:
/root/.cache/pip/wheels/85/fa/a3/9c2d44c9f3cd77cf4e533b58900b2bf4487f2
a17e8ec212a3d
Successfully built lime
Installing collected packages: lime
Successfully installed lime-0.2.0.1
```

```python
# Standard libraries
import os
import re
import string
import copy
import json
import random
import zipfile
import requests
import urllib.request
from pathlib import Path
from collections import Counter, defaultdict

# Data manipulation and visualization
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```python
from tqdm import tqdm
from IPython.display import display, HTML
from wordcloud import WordCloud

# Text processing
"""import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
nltk.download('punkt')
nltk.download('punkt_tab')
nltk.download('wordnet')
nltk.download('stopwords')"""

# Machine learning metrics
from sklearn.metrics import (
    precision_score,
    recall_score,
    f1_score,
    accuracy_score,
    confusion_matrix,
    precision_recall_curve,
    classification_report,
    auc
)
from sklearn.metrics import PrecisionRecallDisplay

# PyTorch
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader, TensorDataset

# Hugging Face Transformers
from transformers import (
    AutoModelForSequenceClassification,
    TFAutoModelForSequenceClassification,
    AutoTokenizer,
    TrainingArguments,
    DataCollatorWithPadding,
    Trainer
)
from scipy.special import softmax

# Datasets
from datasets import Dataset

import warnings
warnings.filterwarnings("ignore", category=FutureWarning)
```

```python
json_files = ["test.json", "training.json", "validation.json"]
dataframes = {}
for file_name in json_files:
    with open(file_name, "r") as file:
        data = json.load(file)
        dataframes[file_name] = pd.DataFrame(data)


# Load the original datasets from the specified JSON files into
separate DataFrames
original_train_df = dataframes['training.json']
original_validation_df = dataframes['validation.json']
original_test_df = dataframes['test.json']

def determine_majority(response_list):
    """
    Counts the occurrences of "YES" and "NO" in the input list and
returns:
      - 1 if "YES" is the majority,
      - 0 if "NO" is the majority,
      - 2 if there is a tie.
    """
    yes_count = response_list.count("YES")
    no_count = response_list.count("NO")
    if yes_count > no_count:
        return 1
    elif no_count > yes_count:
        return 0
    else:
        return 2


def transform_df(df):
    """
    - Transposes the DataFrame.
    - Adds a column `hard_label_task1` based on the majority label in
`labels_task1`.
    - Filters rows where `lang` is 'en' and excludes rows where
`hard_label_task1` equals 2(tie).
    - Selects specific columns for the final output.
    """
    df = df.T
    df['hard_label_task1'] =
df['labels_task1'].apply(determine_majority)
    df = df[df['lang'] == 'en']
    df = df[df['hard_label_task1'] != 2]
    df = df[['id_EXIST', 'lang', 'tweet', 'hard_label_task1']]
    return df
```

```
# Apply the `transform_df` function to preprocess the training,
validation, and test DataFrames
original_train_df = transform_df(original_train_df)
original_validation_df = transform_df(original_validation_df)
original_test_df = transform_df(original_test_df)

original_train_df
```

```
{"summary":"{\n  \"name\": \"original_train_df\",\n  \"rows\": 2870,\n
\"fields\": [\n    {\n      \"column\": \"id_EXIST\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 2870,\n        \"samples\": [\n
\"200504\",\n          \"202694\",\n          \"200852\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"lang\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 1,\n
\"samples\": [\n          \"en\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"tweet\",\n      \"properties\": {\
n        \"dtype\": \"string\",\n        \"num_unique_values\": 2870,\
n        \"samples\": [\n          \"Call me sexist but it just feels
wrong that women are reffing the NBA like go ref the WNBA\\ud83d\\
ude2c\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"hard_label_task1\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"original_train_df"}
```

```
# Create copies of the original datasets to avoid modifying them
directly
train_df = original_train_df.copy()
validation_df = original_validation_df.copy()
test_df = original_test_df.copy()

def clean_tweet(tweet):
    """
    Cleans a tweet by removing unwanted characters, URLs, hashtags,
mentions
    """
    # Remove non-ASCII characters
    tweet = re.sub(r'[^\x00-\x7F]+', '', tweet)
    # Remove hashtags
    tweet = re.sub(r'#\w+', '', tweet)
    # Remove mentions
    tweet = re.sub(r'@\w+', '', tweet)
    # Remove URLs
    tweet = re.sub(r'http\S+|www\S+', '', tweet)
    # Remove non-alphanumeric characters (except spaces)
    tweet = re.sub(r'[^a-zA-Z0-9\s]', '', tweet)
    # Remove specific quote characters
    tweet = tweet.replace('“', '').replace('”', '').replace('‘',
'').replace('’', '')
    # Convert to lowercase
    cleaned_tweet = tweet.lower()

    return cleaned_tweet

# Apply the `clean_tweet` function to the tweet column in the
training, validation, and test datasets
train_df['tweet'] = train_df['tweet'].apply(clean_tweet)
validation_df['tweet'] = validation_df['tweet'].apply(clean_tweet)
test_df['tweet'] = test_df['tweet'].apply(clean_tweet)

test_df.head()
```

```
{"summary":"{\n  \"name\": \"test_df\",\n  \"rows\": 286,\n
\"fields\": [\n    {\n      \"column\": \"id_EXIST\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 286,\n        \"samples\": [\n
\"400187\",\n          \"400471\",\n          \"400331\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"lang\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 1,\n
\"samples\": [\n          \"en\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"tweet\",\n      \"properties\": {\
n        \"dtype\": \"string\",\n        \"num_unique_values\": 286,\n
\"samples\": [\n          \"  sex as in gender harassment is what they
are inferring\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"hard_label_task1\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          1\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"test_df"}
```

Transformer

```
# Define the task type (e.g., 'hate' for hate speech classification)
task = 'hate'
# Set the model name based on the task type
MODEL = f"cardiffnlp/twitter-roberta-base-{task}"
# Load the tokenizer for the specified model
tokenizer = AutoTokenizer.from_pretrained(MODEL)
```

```python
# Load the pre-trained transformer model for sequence classification
transformer_model =
AutoModelForSequenceClassification.from_pretrained(MODEL)#,
output_attentions=True)
# Save the model to the local directory for future use
#transformer_model.save_pretrained(MODEL)
```

```
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/
_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your
settings tab (https://huggingface.co/settings/tokens), set it as
secret in your Google Colab and restart your session.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to
access public models or datasets.
  warnings.warn(
```

{"model_id":"3866351aac5044dda340ec73c7fe260b","version_major":2,"vers
ion_minor":0}

{"model_id":"02faca21e06a4bb99818322d2d6535ed","version_major":2,"vers
ion_minor":0}

{"model_id":"aaca394414cd4ebb896796ab8c4d4b06","version_major":2,"vers
ion_minor":0}

{"model_id":"5d86caf83872499ebb86692d3fa93599","version_major":2,"vers
ion_minor":0}

{"model_id":"e3c83515a9dd4120a7a140ddc71e4547","version_major":2,"vers
ion_minor":0}

```python
def preprocess_text(texts):
    # Use the tokenizer to process the 'tweet' column and apply
truncation to handle long texts
    return tokenizer(texts['tweet'], truncation=True, padding=True)

# Convert the training, validation, and test dataframes into
HuggingFace Dataset objects
train_data = Dataset.from_pandas(train_df)
validation_data = Dataset.from_pandas(validation_df)
test_data = Dataset.from_pandas(test_df)

# Apply the preprocessing function to the dataset, using the 'batched'
option to process in batches
train_data = train_data.map(preprocess_text, batched=True)
validation_data = validation_data.map(preprocess_text, batched=True)
test_data = test_data.map(preprocess_text, batched=True)
```

```python
# Show the processed training data
train_data
```

```
{"model_id":"c04f183cb4c24898a31450e142fb98d0","version_major":2,"version_minor":0}
```

```
Asking to truncate to max_length but no maximum length is provided and the model has no predefined maximum length. Default to no truncation.
```

```
{"model_id":"81b0fd798cc141768e8877ce6ddcc706","version_major":2,"version_minor":0}
```

```
{"model_id":"93340426ba99439589c6aa44ea20f92f","version_major":2,"version_minor":0}
```

```
{"model_id":"7f24fd892a3a4390af0c262820473c20","version_major":2,"version_minor":0}
```

```
Dataset({
    features: ['id_EXIST', 'lang', 'tweet', 'hard_label_task1', '__index_level_0__', 'input_ids', 'attention_mask'],
    num_rows: 2870
})
```

```python
# Rename the label columns to match with transformer default
train_data = train_data.rename_column('hard_label_task1', 'label')
validation_data = validation_data.rename_column('hard_label_task1', 'label')
test_data = test_data.rename_column('hard_label_task1', 'label')

data_collator = DataCollatorWithPadding(tokenizer=tokenizer)

transformer_training_args = TrainingArguments(
    output_dir="test_dir",
    learning_rate=1e-6,
    per_device_train_batch_size=4,
    per_device_eval_batch_size=8,
    num_train_epochs=4,
    weight_decay=0.2,
    eval_strategy="epoch",
    save_strategy="epoch",
    load_best_model_at_end=True,
    report_to='none'
)

def compute_metrics(eval_pred):
    predictions, labels = eval_pred[0], eval_pred[1]
    predictions = np.argmax(predictions, axis=1)

    f1 = f1_score(y_true=labels, y_pred=predictions, average='macro')
    acc = accuracy_score(y_true=labels, y_pred=predictions)
    return {'f1': f1, 'acc': acc}
```

```
transformer_trainer = Trainer(
    model=transformer_model,
    args=transformer_training_args,
    train_dataset=train_data,
    eval_dataset=validation_data,
    tokenizer=tokenizer,
    data_collator=data_collator,
    compute_metrics=compute_metrics,
)

transformer_trainer.train()
transformer_trainer.save_model("test_dir")
```

```
<IPython.core.display.HTML object>
```

```
transformer_test_prediction_info =
transformer_trainer.predict(test_data)

# Extract the model predictions and the true labels from the
prediction result
transformer_test_predictions, transformer_test_labels =
transformer_test_prediction_info.predictions,
transformer_test_prediction_info.label_ids
```

```
<IPython.core.display.HTML object>
```

```
# Compute the evaluation metrics (such as F1 score and accuracy) for
the test predictions
transformer_test_metrics =
compute_metrics([transformer_test_predictions,
transformer_test_labels])

# Extract the F1 score and accuracy from the computed metrics
transformer_f1 = transformer_test_metrics['f1']
transformer_accuracy = transformer_test_metrics['acc']

print(f"Accuracy on test: {transformer_accuracy:.4f}\nf1-score on
test: {transformer_f1:.4f}", end="\n\n")
```

```
Accuracy on test: 0.8322
f1-score on test: 0.8310
```

Explainability

Attention Weights

```
transformer_model =
AutoModelForSequenceClassification.from_pretrained(
    "test_dir",
    output_attentions=True
)
transformer_model.eval()
```

```
RobertaForSequenceClassification(
  (roberta): RobertaModel(
    (embeddings): RobertaEmbeddings(
      (word_embeddings): Embedding(50265, 768, padding_idx=1)
      (position_embeddings): Embedding(514, 768, padding_idx=1)
      (token_type_embeddings): Embedding(1, 768)
      (LayerNorm): LayerNorm((768,), eps=1e-05,
elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (encoder): RobertaEncoder(
      (layer): ModuleList(
        (0-11): 12 x RobertaLayer(
          (attention): RobertaAttention(
            (self): RobertaSdpaSelfAttention(
              (query): Linear(in_features=768, out_features=768,
bias=True)
              (key): Linear(in_features=768, out_features=768,
bias=True)
              (value): Linear(in_features=768, out_features=768,
bias=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
            (output): RobertaSelfOutput(
              (dense): Linear(in_features=768, out_features=768,
bias=True)
              (LayerNorm): LayerNorm((768,), eps=1e-05,
elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
          (intermediate): RobertaIntermediate(
            (dense): Linear(in_features=768, out_features=3072,
bias=True)
            (intermediate_act_fn): GELUActivation()
          )
          (output): RobertaOutput(
            (dense): Linear(in_features=3072, out_features=768,
bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-05,
elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
      )
    )
  )
  (classifier): RobertaClassificationHead(
    (dense): Linear(in_features=768, out_features=768, bias=True)
    (dropout): Dropout(p=0.1, inplace=False)
    (out_proj): Linear(in_features=768, out_features=2, bias=True)
  )
)
```

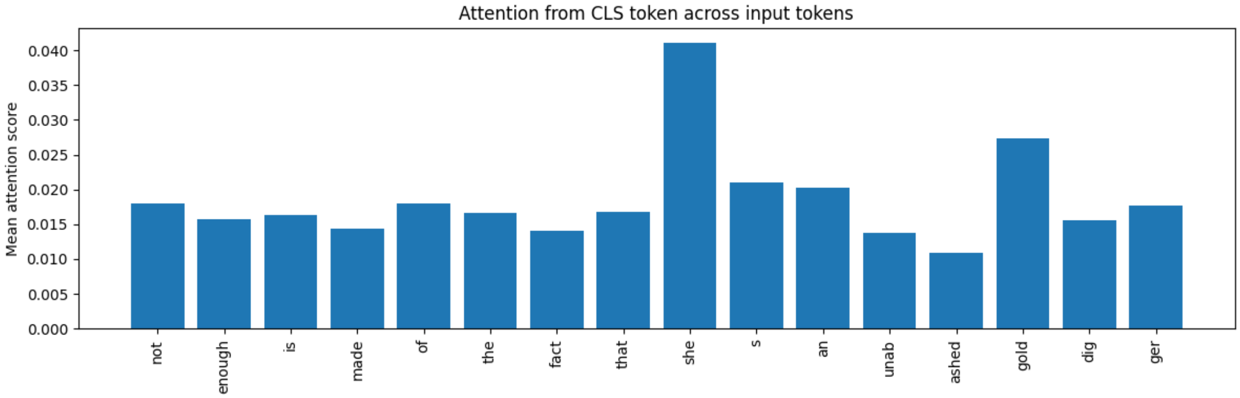
```
from bertviz import head_view
from transformers import AutoModel
import torch
import torch.nn.functional as F

# Choose device
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

# Prepare input sentence
index = 25
text =  [token for token in test_data[index]['padded_tweet'] if token
!= "<PAD>"]
sentence = " ".join(text) # or just use your raw text directly

# Tokenize
inputs = tokenizer(sentence, return_tensors='pt').to(device)  # Move
inputs to device
#print("Input tokens:",
tokenizer.convert_ids_to_tokens(inputs['input_ids'][0]))

# Move model to device
transformer_model = transformer_model.to(device)
transformer_model.eval()

# Inference with no gradients
with torch.no_grad():
    outputs = transformer_model(**inputs)
    logits = outputs.logits         # [1, num_classes]
    attentions = outputs.attentions  # list: [layer1, ..., layerN]


probs = F.softmax(logits, dim=-1)      # Convert logits to
probabilities
pred_class = torch.argmax(probs, dim=-1).item()
confidence = probs[0, pred_class].item()
id2label = {
    0: "non-sexist",
    1: "sexist"
}
label = id2label[pred_class]
print(f"Prediction: {label} (confidence: {confidence:.4f}), label:
```

```python
{id2label[test_df.iloc[index]['hard_label_task1']]}")

# Decode tokens
tokens = tokenizer.convert_ids_to_tokens(inputs["input_ids"][0])
tokens = [token.replace('Ġ', '') for token in tokens]

# Visualize (assumes head_view handles CPU/GPU internally or takes CPU input)
head_view(attention=[att.cpu() for att in attentions], tokens=tokens)
"""
Using test_data[4]['padded_tweet']
5-Red-> seems to connect subject with refering to subject, that-cunt, you-remember, say-shit
0-Grey->each to themself
0-Acqua->each to next tooken
0-Orange->each to previous token
Layer 1 most attend start token
2-Pink -> each attend to himself, u attend to both you and also you,
2-Violet-> u and you attend to previous you
2-Grey-> EACH TOKEN ATTEN DTO THE PREVIOUS 3/4 TOKENS
4-Pink-> each token attend only to start and end
8-Violet->Cunt, Slut, Dumb have a lot of incoming edge
In the last layer most head are all-to-all and like 2or3 are all-to-start/end
"""
```

```
Prediction: sexist (confidence: 0.8509), label: sexist

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.Javascript object>

{"type":"string"}
```

```python
# attention: Tensor of shape [layers, heads, seq_len, seq_len]
# Example: attention = torch.tensor(attention_data)
# Make sure it's a torch.Tensor
attentions_stacked = torch.stack(attentions).squeeze(1)
assert attentions_stacked.dim() == 4, "Expected [layers, heads, seq_len, seq_len]"

# Extract attention from CLS token (position 0) to others
# Shape: [layers, heads, target_tokens]
cls_attention = attentions_stacked[:, :, 0, :]  # attention FROM CLS to each token

# Average across layers and heads
cls_attention_mean = cls_attention.mean(dim=(0, 1))  # [seq_len]
tokens = [token.replace('Ġ', '') for token in tokens][1:-1]
```

```
plt.figure(figsize=(12, 4))
plt.bar(range(len(cls_attention_mean.cpu()[1:-1])),
cls_attention_mean.cpu().numpy()[1:-1])
plt.xticks(range(len(tokens)), tokens, rotation=90)
plt.title("Attention from CLS token across input tokens")
plt.ylabel("Mean attention score")
plt.tight_layout()
plt.show()
```



```
all_cls_attentions = torch.stack([att[:, :, 0, :] for att in
attentions])
print(all_cls_attentions.shape)

mean_cls_attentions = all_cls_attentions.mean(dim=0).mean(dim=1)
mean_cls_attentions.shape
print(mean_cls_attentions)

torch.Size([12, 1, 12, 18])
tensor([[0.5017, 0.0180, 0.0157, 0.0163, 0.0143, 0.0180, 0.0166,
0.0141, 0.0168,
         0.0411, 0.0210, 0.0203, 0.0137, 0.0109, 0.0273, 0.0155,
0.0176, 0.2012]],
       device='cuda:0')

test_data


Dataset({
    features: ['id_EXIST', 'lang', 'tweet', 'label', 'padded_tweet',
'__index_level_0__', 'input_ids', 'attention_mask'],
    num_rows: 286
})

from tqdm import tqdm
import torch
import torch.nn.functional as F
```

```
from datasets import concatenate_datasets


# These will store raw attention data for each class
all_attentions_sexist = []
all_input_ids_sexist = []

all_attentions_non_sexist = []
all_input_ids_non_sexist = []

# Phase 1: Forward pass & store attentions by predicted class
data = concatenate_datasets([validation_data, test_data])
for sample in tqdm(data):
    input_ids = torch.tensor([sample["input_ids"]]).to(device)
    attention_mask =
torch.tensor([sample["attention_mask"]]).to(device)

    with torch.no_grad():
        outputs = transformer_model(input_ids=input_ids,
attention_mask=attention_mask, output_attentions=True)
        logits = outputs.logits
        probs = F.softmax(logits, dim=-1)
        pred_class = torch.argmax(probs, dim=-1).item()

        attentions = [att.cpu() for att in outputs.attentions]  # list
of (1, heads, seq_len, seq_len)

    if pred_class == 0:
        all_attentions_non_sexist.append(attentions)
        all_input_ids_non_sexist.append(sample["input_ids"])
    else:
        all_attentions_sexist.append(attentions)
        all_input_ids_sexist.append(sample["input_ids"])


# Utility to compute average token attention
def compute_avg_token_attention(all_attentions, all_input_ids):
    token_freq = defaultdict(int)
    token_attention_sum = defaultdict(float)

    for i in range(len(all_attentions)):
        attention_layers = torch.stack(all_attentions[i]).squeeze(1)
# shape: [num_layers, num_heads, seq_len, seq_len]

        """input_ids = all_input_ids[i]  # list of token ids
        # Get attention from CLS to all tokens: shape [layers, heads,
seq_len]
        cls_attention = attention_layers[:, :, 0, :]
        # Average over layers and heads: shape [seq_len]
        avg_cls_attention = cls_attention.mean(dim=(0, 1))"""
        last_layer_attention = attention_layers[-1].squeeze(0)  #
shape: [num_heads, seq_len, seq_len]
        input_ids = all_input_ids[i]  # list of token ids
        # Get attention from CLS to all tokens: shape [heads, seq_len]
        cls_attention = last_layer_attention[:, 0, :]  # heads x
seq_len
        # Average over heads: shape [seq_len]
        avg_cls_attention = cls_attention.mean(dim=0)  # seq_len

        for token_id, att_score in zip(input_ids, avg_cls_attention):
            token_freq[token_id] += 1
            token_attention_sum[token_id] += float(att_score)

    # Final average
    token_avg_attention = {
        token_id: token_attention_sum[token_id] / token_freq[token_id]
        for token_id in token_freq
    }
    return token_avg_attention


# Phase 2: Compute attention stats for each group
token_avg_attention_non_sexist =
compute_avg_token_attention(all_attentions_non_sexist,
all_input_ids_non_sexist)
token_avg_attention_sexist =
compute_avg_token_attention(all_attentions_sexist,
all_input_ids_sexist)
```

```
100%|██████████| 444/444 [00:07<00:00, 58.33it/s]
```

```
# Example: print top attended tokens in sexist predictions
top_tokens = sorted(token_avg_attention_sexist.items(), key=lambda x:
-x[1])[:50]
for token_id, avg_score in top_tokens:
    print(tokenizer.decode([token_id]).replace(' ', ''), f"→
{avg_score:.4f}")
```

```
bald → 0.1310
they → 0.1209
que → 0.1207
bathing → 0.1112
you → 0.1100
took → 0.0993
gal → 0.0973
nails → 0.0963
school → 0.0954
skirt → 0.0947
room → 0.0939
```

```
anking → 0.0923
ush → 0.0883
misogyny → 0.0881
whore → 0.0864
woman → 0.0863
fem → 0.0863
calling → 0.0856
ulation → 0.0852
pop → 0.0829
arius → 0.0827
making → 0.0819
<s> → 0.0817
wall → 0.0814
</s> → 0.0814
testosterone → 0.0810
ches → 0.0809
feminism → 0.0800
went → 0.0798
bounce → 0.0798
maybe → 0.0797
forever → 0.0796
slut → 0.0791
hh → 0.0787
which → 0.0780
penis → 0.0760
summer → 0.0757
lesbian → 0.0753
aga → 0.0753
citizens → 0.0750
how → 0.0746
bag → 0.0745
females → 0.0745
boy → 0.0742
azi → 0.0731
economy → 0.0724
teachers → 0.0718
tease → 0.0715
days → 0.0713
prostitute → 0.0712
```

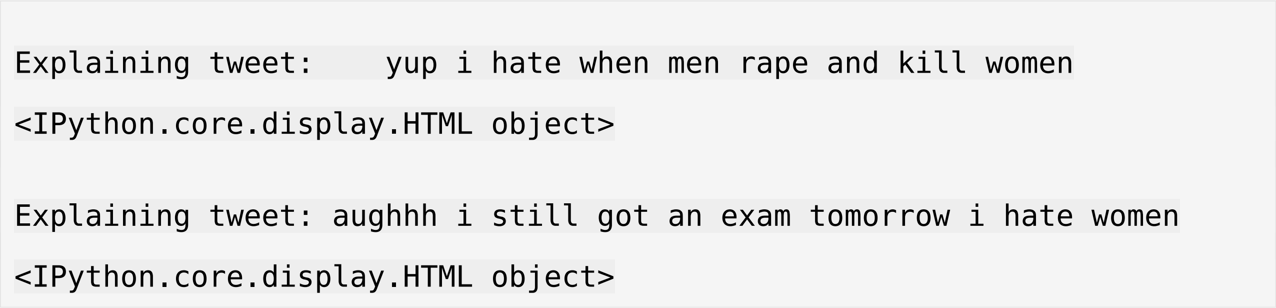
```python
# Example: print top attended tokens in sexist predictions
top_tokens = sorted(token_avg_attention_non_sexist.items(), key=lambda
x: -x[1])[:50]
for token_id, avg_score in top_tokens:
    print(tokenizer.decode([token_id]), f"→ {avg_score:.4f}")
```

```
 witches → 0.2327
 tits → 0.1641
 sex → 0.1556
 congratulations → 0.1539
 blonde → 0.1468
 sexism → 0.1404
<s> → 0.1316
</s> → 0.1316
 stroke → 0.1268
 lady → 0.1161
 taxes → 0.1122
 harm → 0.1076
ika → 0.1054
ude → 0.1052
mith → 0.1037
 fascists → 0.1026
 question → 0.0990
 porn → 0.0979
 birds → 0.0941
 coins → 0.0938
 terrorists → 0.0935
 abuse → 0.0916
 cum → 0.0913
 air → 0.0908
 woman → 0.0862
 beard → 0.0832
 pregnancy → 0.0828
 choice → 0.0827
coins → 0.0817
 legs → 0.0815
 pee → 0.0813
 luck → 0.0810
 girls → 0.0808
 controlling → 0.0807
 masturb → 0.0801
life → 0.0793
 linebackers → 0.0791
 tire → 0.0788
 slap → 0.0787
 harassment → 0.0783
lad → 0.0781
 nails → 0.0781
 boys → 0.0780
 feminist → 0.0780
 consent → 0.0779
 misconduct → 0.0775
 guests → 0.0772
 al → 0.0768
 furry → 0.0766
 feminine → 0.0757
```

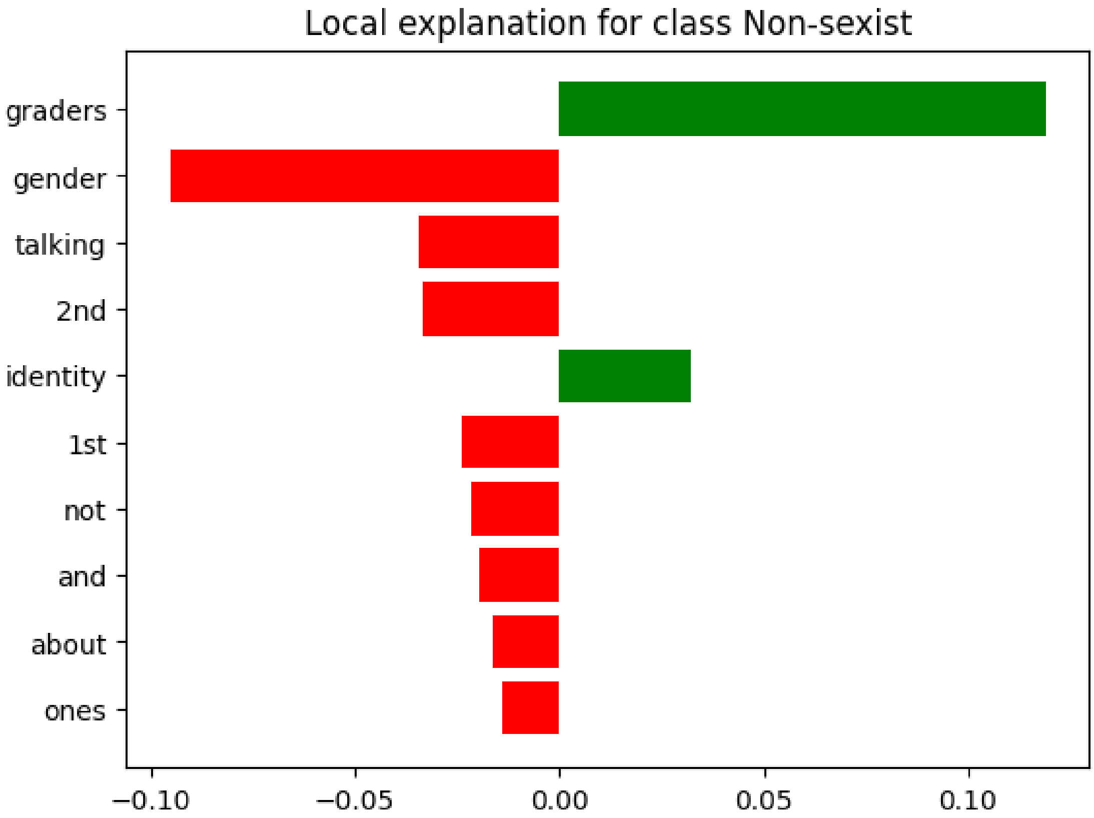
LIME

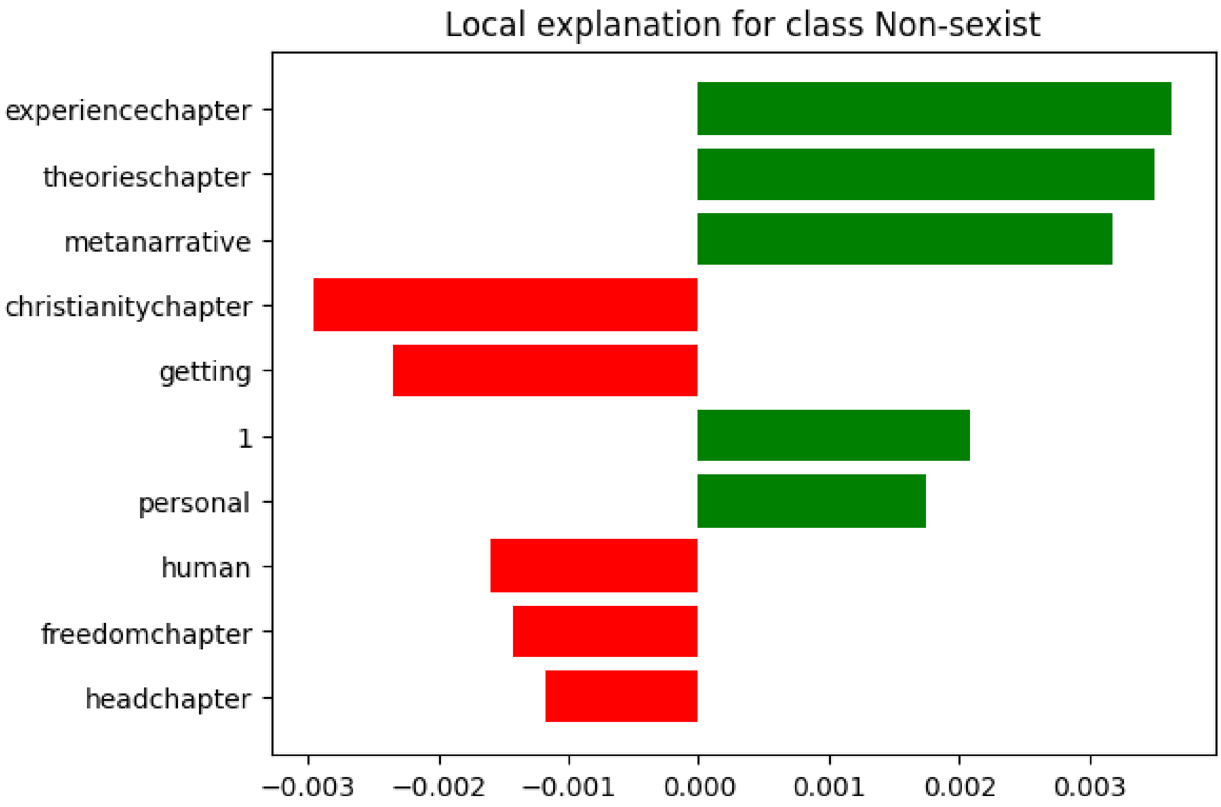
```python
from lime.lime_text import LimeTextExplainer

label_map = {0: "non-sexist", 1: "sexist"}  # Customize if needed

def predict_proba(texts):
    """
    Generates probability predictions for text classification using a pre-trained model.
    """
    inputs = tokenizer(texts, padding=True, truncation=True, return_tensors="pt").to(model.device)
    with torch.no_grad():
        outputs = model(**inputs)
        probs = torch.nn.functional.softmax(outputs.logits, dim=1)
    return probs.cpu().numpy()

def explain_tweets_with_lime(tweets, explainer, predict_proba, num_features=10, num_samples=500, flg_batch_eval=False):
    """
    Generates LIME explanations for tweet classifications, showing which words influence the model's predictions.
    """
    feature_aggregate = defaultdict(lambda: defaultdict(float))  # class_label -> feature -> weight

    for i, tweet in enumerate(tweets):
        if not flg_batch_eval:
            print(f"\nExplaining tweet: {tweet}")

        exp = explainer.explain_instance(tweet, predict_proba, num_features=num_features, num_samples=num_samples, labels=[0, 1])

        # Get predicted class
        pred_class_idx = int(predict_proba([tweet])[0].argmax())
        class_label = label_map[pred_class_idx]

        # Extract explanation weights
        weights = dict(exp.as_list(label=pred_class_idx))

        if flg_batch_eval:
            # Aggregate weights per class
            for feature, weight in weights.items():
                feature_aggregate[class_label][feature] += weight
        else:
            fig = exp.as_pyplot_figure(label=pred_class_idx)
            fig.savefig(f"lime_explanation_{i}.png", dpi=300, bbox_inches='tight')
            exp.show_in_notebook()
        del exp
        del weights
        torch.cuda.empty_cache()

    if flg_batch_eval:
        # Plot separate bar chart for each class
        for class_label, weights_dict in feature_aggregate.items():
            top_features = sorted(weights_dict.items(), key=lambda x:
abs(x[1]), reverse=True)[:20]
            features, weights = zip(*top_features)
            colors = ['red' if w > 0 else 'green' for w in weights]

            plt.figure(figsize=(8, 6))
            plt.barh(features, weights, color=colors)
            plt.xlabel('Total Weight')
            plt.title(f'Top 20 Aggregated LIME Feature Importance
({class_label})')
            plt.axvline(0, color='black', linewidth=0.8)
            plt.tight_layout()
            plt.show()

        return {class_label: dict(weights) for class_label, weights in
feature_aggregate.items()}
    else:
        return None

tweets = []
for i in [12, 78, 56, 46]:
  tweets.append(test_data[i]['tweet'])

model = transformer_trainer.model
class_names = ['Non-sexist', 'Sexist']

explainer = LimeTextExplainer(class_names=class_names, bow=False)

feature_weights = explain_tweets_with_lime(tweets, explainer,
predict_proba)
```

```
Explaining tweet:  please not the ones talking to 1st and 2nd  graders
about gender identity

<IPython.core.display.HTML object>


Explaining tweet: 23chapter 1 getting inside my headchapter 2 on
theorieschapter 3 the metanarrative of christianitychapter 4 argument
of entitieschapter 5 argument from personal experiencechapter 6 human
freedomchapter 7 gods providence

<IPython.core.display.HTML object>
```

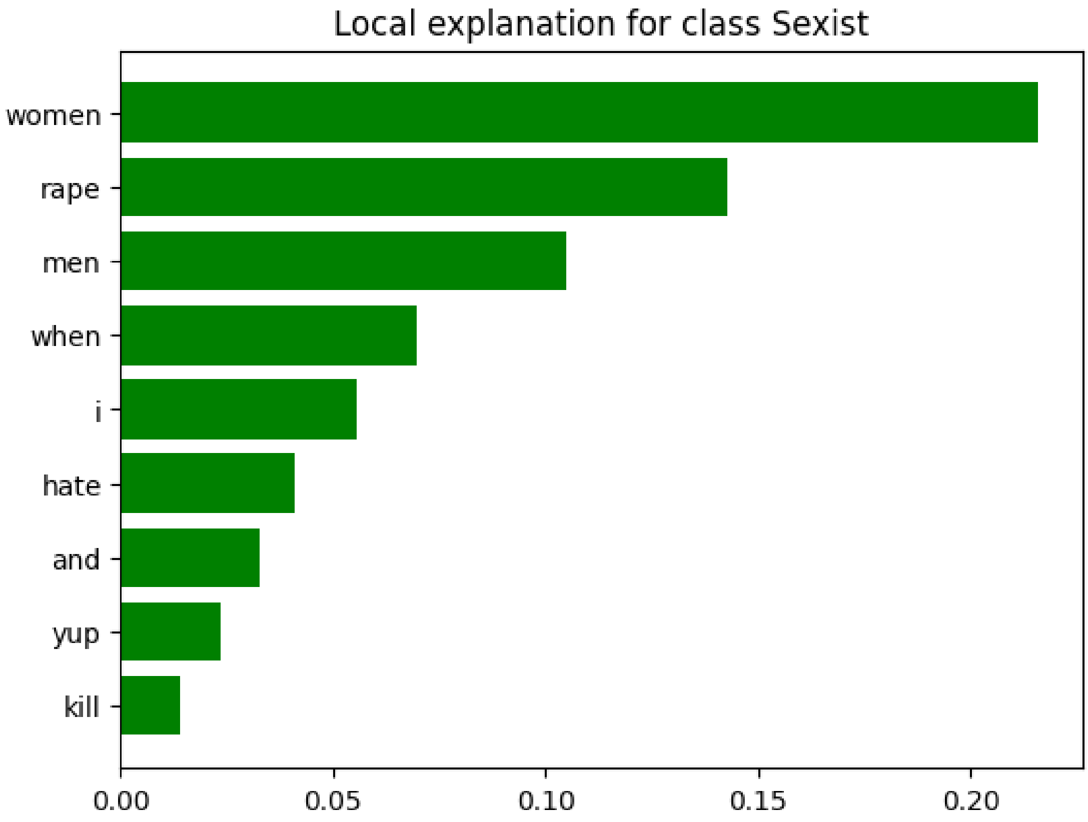


```
Explaining tweet:    yup i hate when men rape and kill women
<IPython.core.display.HTML object>

Explaining tweet: aughhh i still got an exam tomorrow i hate women
<IPython.core.display.HTML object>
```

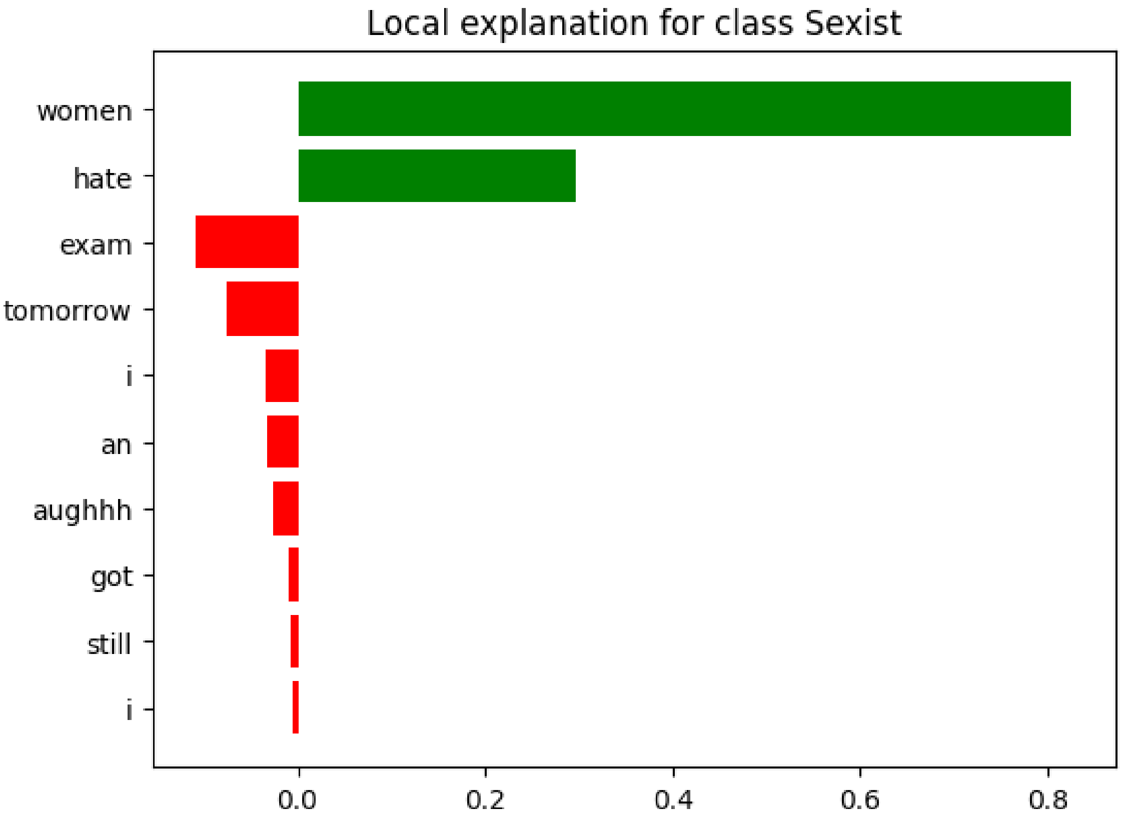




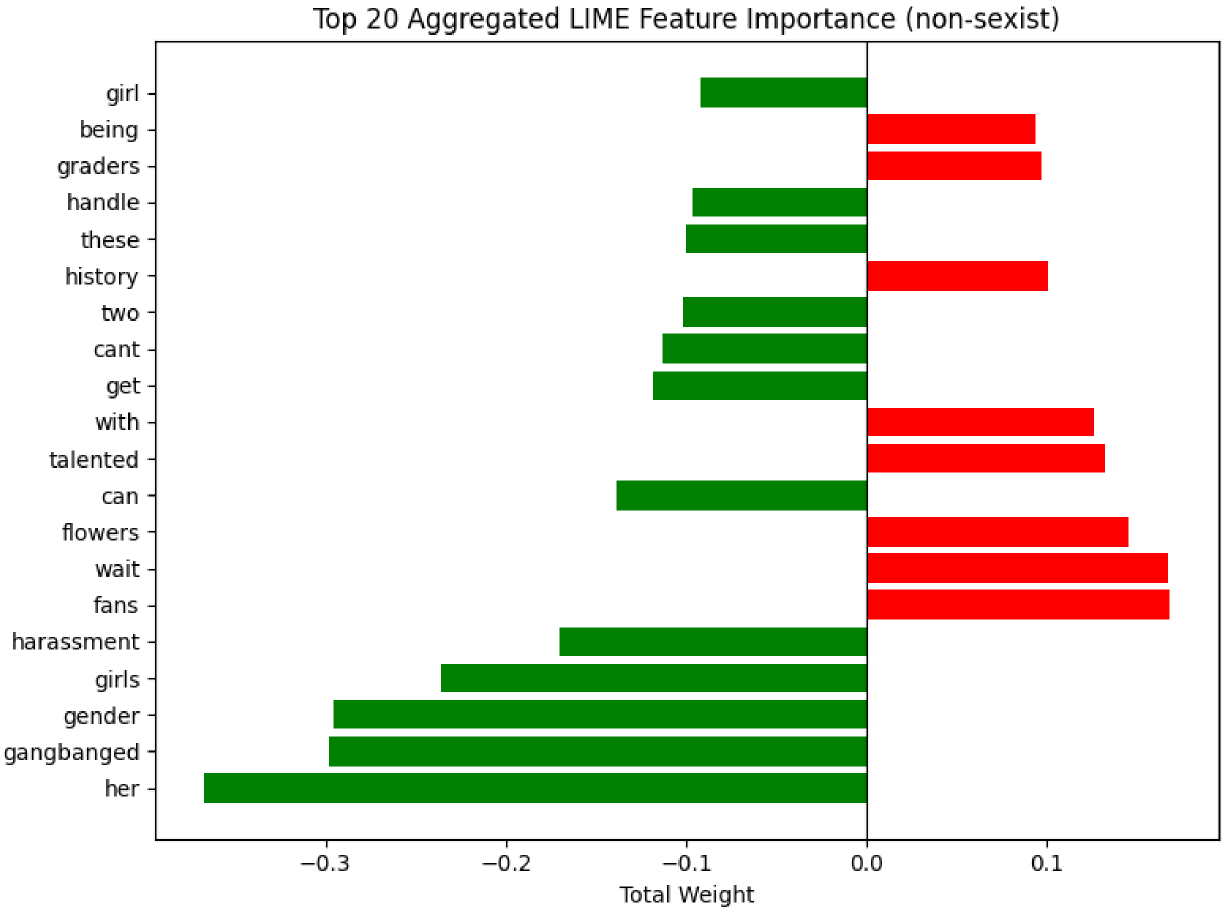
Local explanation for class Non-sexist

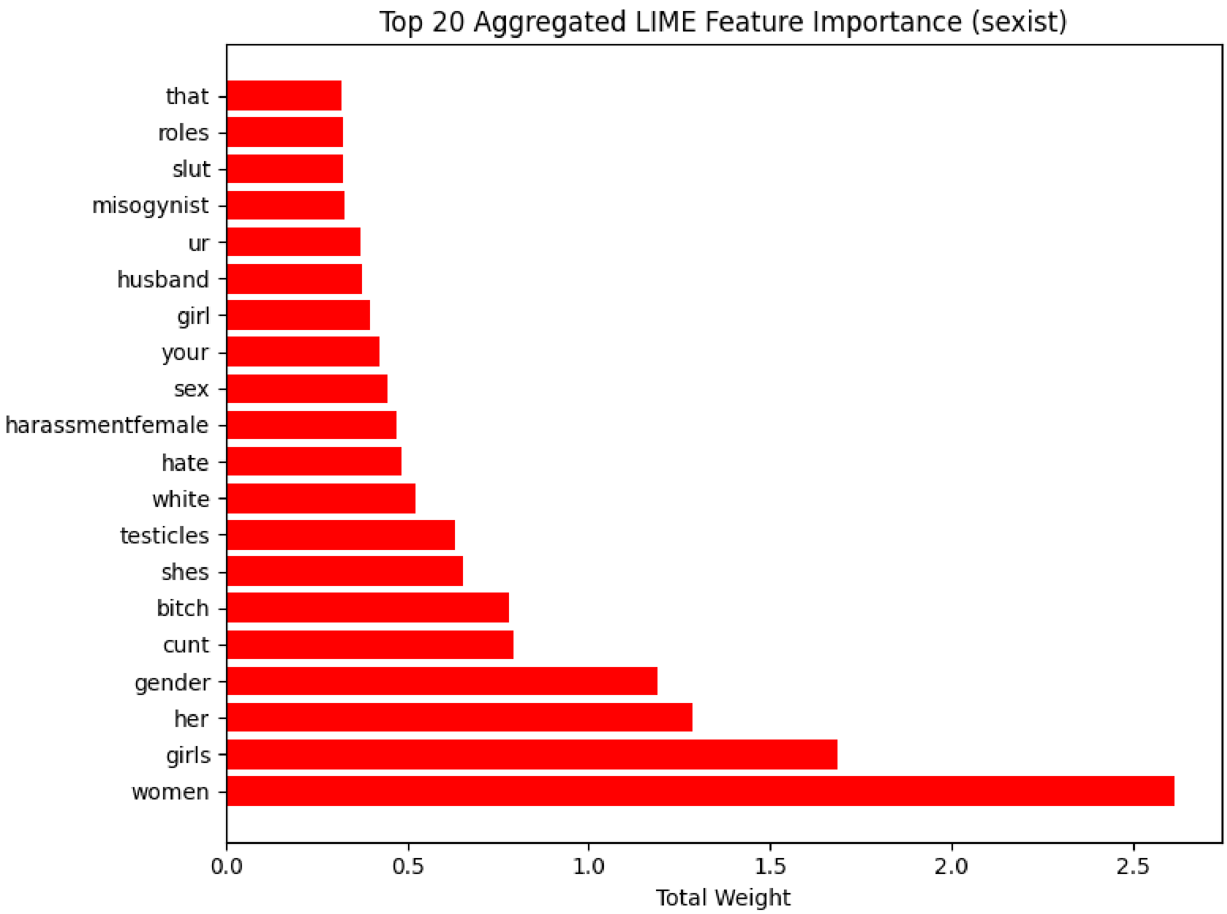


Local explanation for class Sexist



```
tweets = []
for i in range(0, 50):
  tweets.append(test_data[i]['tweet'])

feature_weights = explain_tweets_with_lime(tweets, explainer,
predict_proba, flg_batch_eval=True)
```


Top 20 Aggregated LIME Feature Importance (non-sexist)
girl
being
graders
handle
these
history
two
cant
get
with
talented
can
flowers
wait
fans
harassment
girls
gender
gangbanged
her
−0.3
−0.2
−0.1
0.0
0.1
Total Weight



SHAP

```
pip install shap

Requirement already satisfied: shap in /usr/local/lib/python3.11/dist-packages (0.47.2)
Requirement already satisfied: numpy in
/usr/local/lib/python3.11/dist-packages (from shap) (2.0.2)
Requirement already satisfied: scipy in
/usr/local/lib/python3.11/dist-packages (from shap) (1.15.3)
Requirement already satisfied: scikit-learn in
/usr/local/lib/python3.11/dist-packages (from shap) (1.6.1)
Requirement already satisfied: pandas in
/usr/local/lib/python3.11/dist-packages (from shap) (2.2.2)
Requirement already satisfied: tqdm>=4.27.0 in
/usr/local/lib/python3.11/dist-packages (from shap) (4.67.1)
Requirement already satisfied: packaging>20.9 in
/usr/local/lib/python3.11/dist-packages (from shap) (24.2)
Requirement already satisfied: slicer==0.0.8 in
/usr/local/lib/python3.11/dist-packages (from shap) (0.0.8)
Requirement already satisfied: numba>=0.54 in
/usr/local/lib/python3.11/dist-packages (from shap) (0.60.0)
```

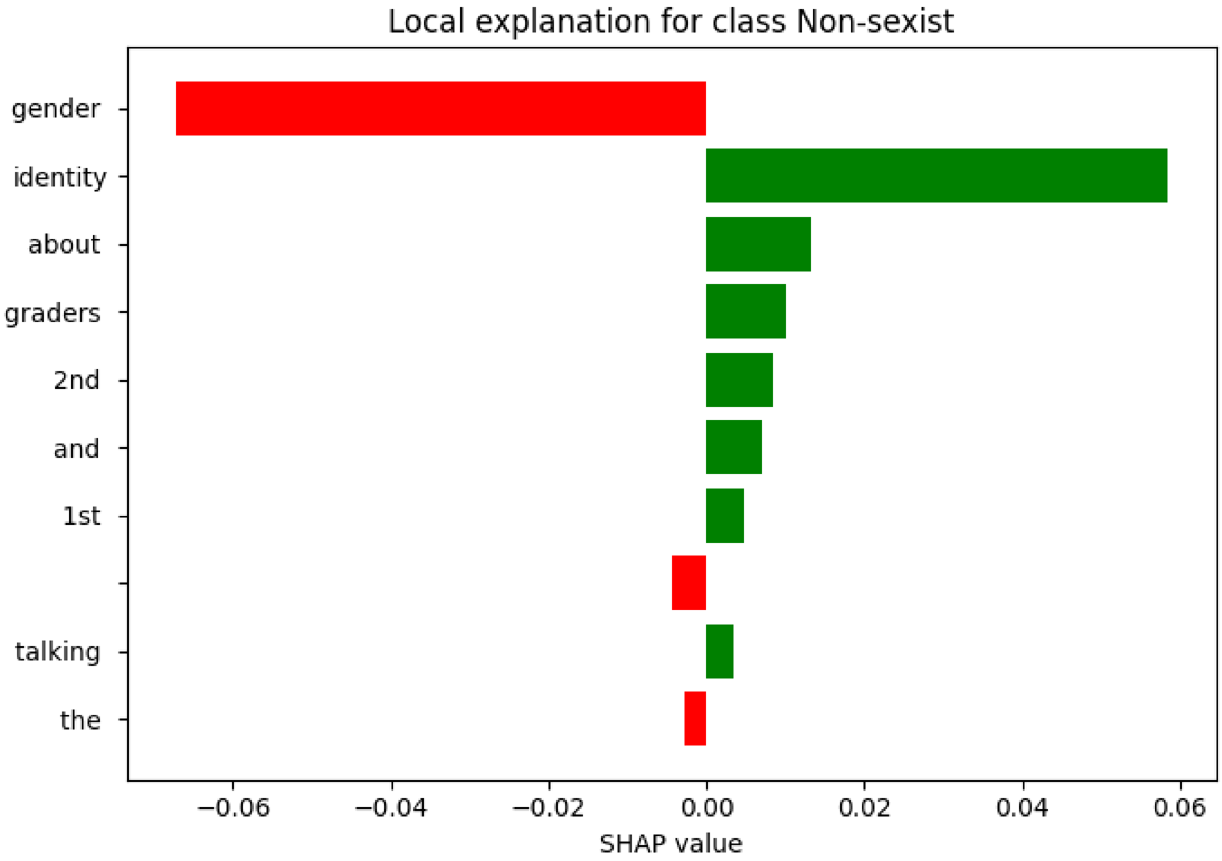
```
Requirement already satisfied: cloudpickle in
/usr/local/lib/python3.11/dist-packages (from shap) (3.1.1)
Requirement already satisfied: typing-extensions in
/usr/local/lib/python3.11/dist-packages (from shap) (4.13.2)
Requirement already satisfied: llvmlite<0.44,>=0.43.0dev0 in
/usr/local/lib/python3.11/dist-packages (from numba>=0.54->shap)
(0.43.0)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.11/dist-packages (from pandas->shap)
(2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in
/usr/local/lib/python3.11/dist-packages (from pandas->shap) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in
/usr/local/lib/python3.11/dist-packages (from pandas->shap) (2025.2)
Requirement already satisfied: joblib>=1.2.0 in
/usr/local/lib/python3.11/dist-packages (from scikit-learn->shap)
(1.5.0)
Requirement already satisfied: threadpoolctl>=3.1.0 in
/usr/local/lib/python3.11/dist-packages (from scikit-learn->shap)
(3.6.0)
Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2-
>pandas->shap) (1.17.0)
```

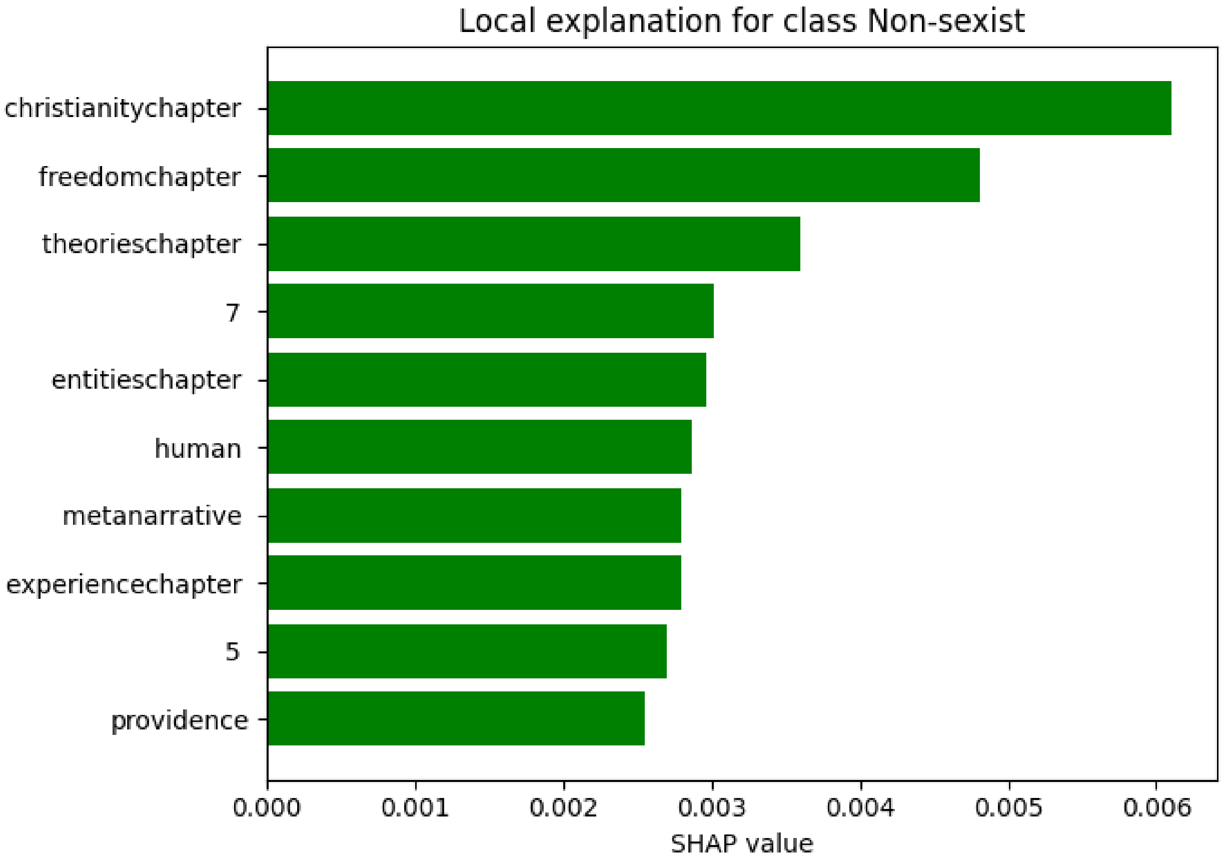
```python
import torch
import numpy as np
import re
import shap

def f(x):
    tv = torch.tensor([
        tokenizer.encode(v, padding="max_length", max_length=128,
truncation=True)
        for v in x
    ]).cuda()
    attention_mask = (tv != 0).type(torch.int64).cuda()
    outputs = transformer_model(tv, attention_mask=attention_mask)
[0].detach().cpu().numpy()
    scores = (np.exp(outputs).T / np.exp(outputs).sum(-1)).T  #
softmax
    val = np.log(scores)  # LOG PROBABILITY
    return val

def custom_tokenizer(s, return_offsets_mapping=True):
    """Tokenizza il testo dividendo per caratteri non alfanumerici."""
    pos = 0
    offset_ranges = []
    input_ids = []
    for m in re.finditer(r"\W", s):
        start, end = m.span(0)
        offset_ranges.append((pos, start))
        input_ids.append(s[pos:start])
        pos = end
    if pos != len(s):
        offset_ranges.append((pos, len(s)))
        input_ids.append(s[pos:])
    out = {
        "input_ids": input_ids,
    }
    if return_offsets_mapping:
        out["offset_mapping"] = offset_ranges
    return out

masker = shap.maskers.Text(custom_tokenizer)
tweets = []
for i in [12, 78, 56, 46]:
    tweets.append(test_data[i]['tweet'])

output_names = ['Non-sexist', 'Sexist']
explainer = shap.Explainer(f, masker, output_names=output_names)
shap_values = explainer(tweets)

# classification probability
probs = np.exp(f(tweets))

for i, tweet in enumerate(tweets):
    print(f"\nTweet #{i+1}: {tweet}")
    print("Prediction probabilities:")
    for cls, p in zip(output_names, probs[i]):
        print(f"  {cls}: {p:.2f}")
    shap.plots.text(shap_values[i])
```

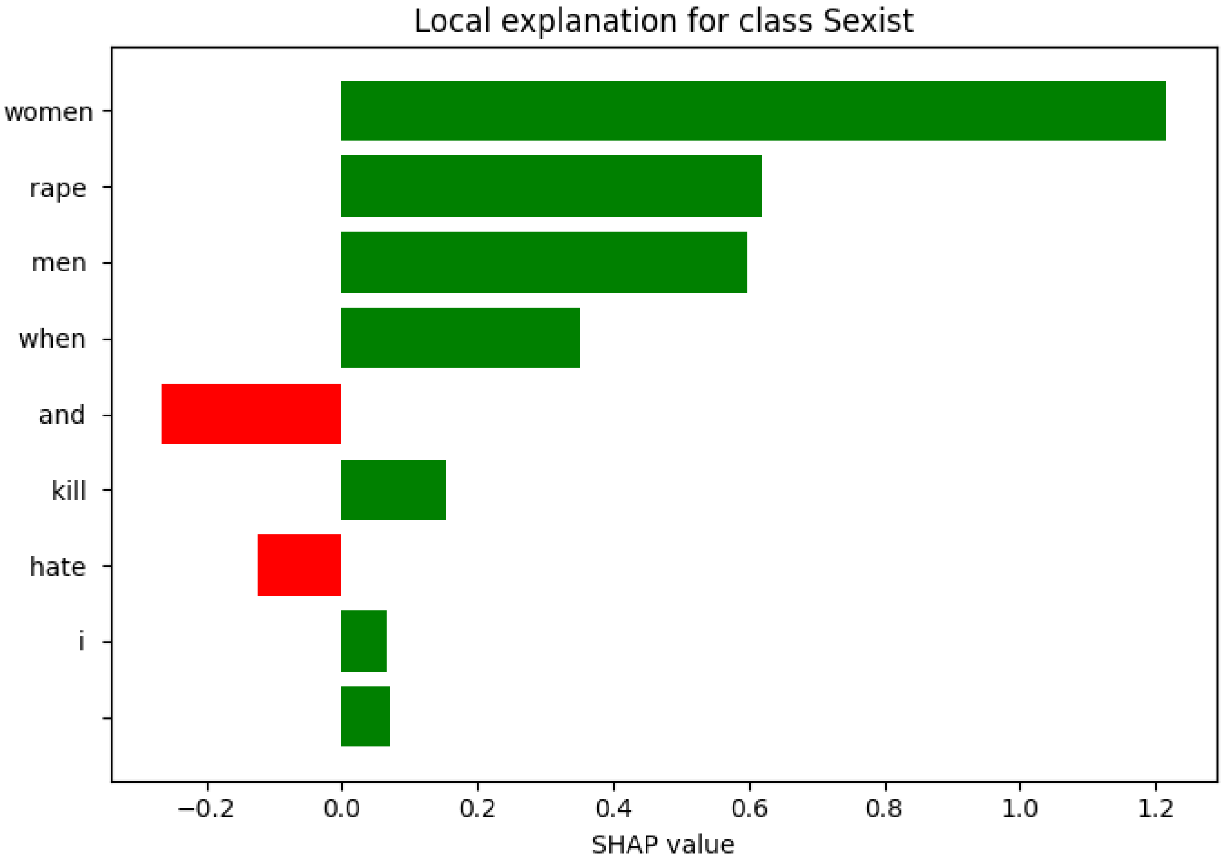
```
PartitionExplainer explainer: 5it [00:16,  4.03s/it]


Tweet #1:  please not the ones talking to 1st and 2nd  graders about
gender identity
Prediction probabilities:
  Non-sexist: 0.98
  Sexist: 0.02



<IPython.core.display.HTML object>


Tweet #2: 23chapter 1 getting inside my headchapter 2 on
theorieschapter 3 the metanarrative of christianitychapter 4 argument
of entitieschapter 5 argument from personal experiencechapter 6 human
freedomchapter 7 gods providence
Prediction probabilities:
  Non-sexist: 0.99
  Sexist: 0.01

<IPython.core.display.HTML object>


Tweet #3:    yup i hate when men rape and kill women
Prediction probabilities:
  Non-sexist: 0.09
  Sexist: 0.91

<IPython.core.display.HTML object>


Tweet #4: aughhh i still got an exam tomorrow i hate women
Prediction probabilities:
  Non-sexist: 0.05
  Sexist: 0.95

<IPython.core.display.HTML object>
```

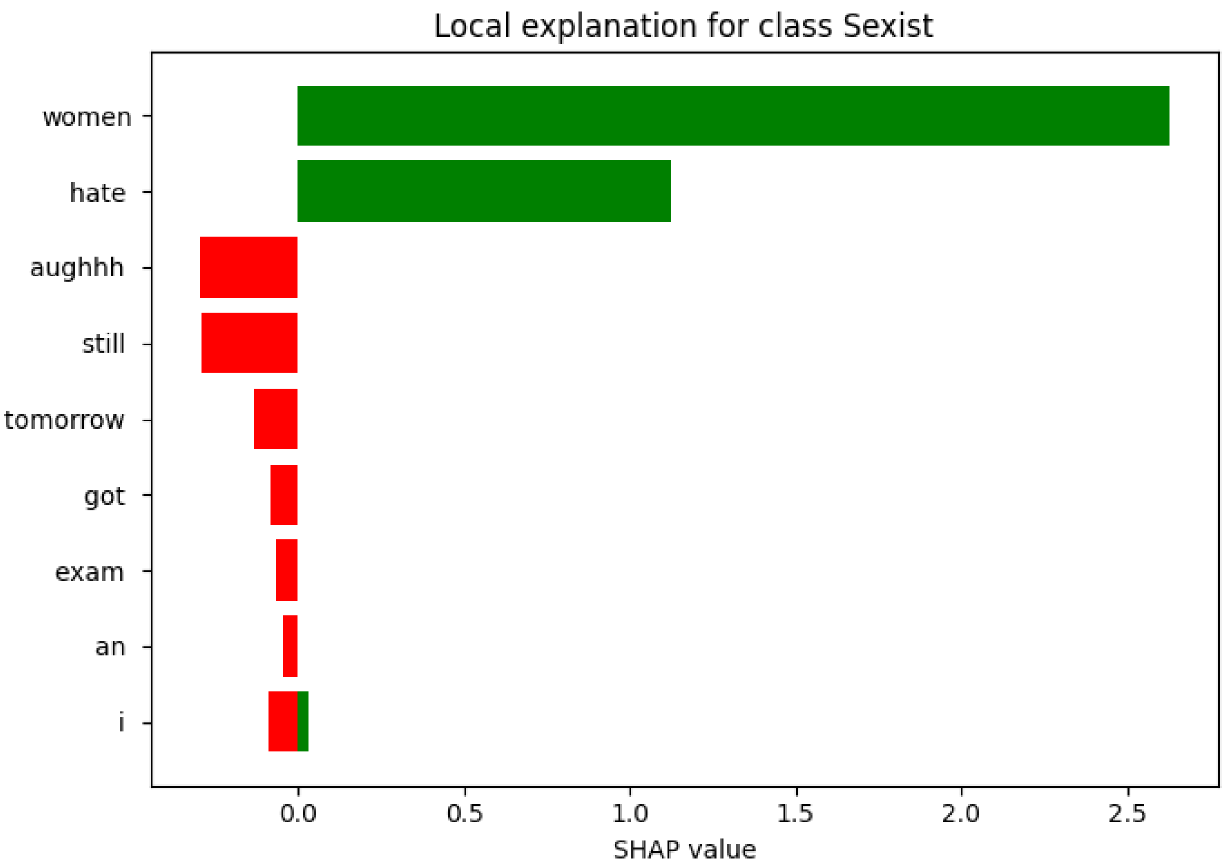
```python
import matplotlib.pyplot as plt

# Funzione per ottenere la classe predetta
def get_predicted_class(probs_row):
    return np.argmax(probs_row)

# Per ogni tweet analizzato
for i, tweet in enumerate(tweets):
    pred_class_idx = get_predicted_class(probs[i])  # indice della
classe predetta
    pred_class_label = output_names[pred_class_idx]

    # Ottieni valori SHAP e token
    shap_vals = shap_values[i].values[:, pred_class_idx]  # valori
SHAP per la classe predetta
    tokens = shap_values[i].data

    # Associa token e valore SHAP e ordina per importanza assoluta
    token_shap_pairs = list(zip(tokens, shap_vals))
    token_shap_pairs.sort(key=lambda x: abs(x[1]), reverse=True)

    # Prendi i top 10 token
    top_tokens, top_shap = zip(*token_shap_pairs[:10])

    # Plot
    plt.figure(figsize=(7, 5))
    colors = ['green' if val > 0 else 'red' for val in top_shap]
    plt.barh(top_tokens[::-1], top_shap[::-1], color=colors[::-1])
    plt.xlabel("SHAP value")
    plt.title(f"Local explanation for class {pred_class_label}")
```

```
#plt.grid(True)
plt.tight_layout()
plt.show()
```




Local explanation for class Non-sexist
christianitychapter
freedomchapter
theorieschapter
7
entitieschapter
human
metanarrative
experiencechapter
5
providence
0.000
0.001
0.002
0.003
0.004
0.005
0.006
SHAP value


Local explanation for class Sexist
women
rape
men
when
and
kill
hate
i
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2
SHAP value



```python
tweets = []
for i in range(0, 50):
  tweets.append(test_data[i]['tweet'])

# Calcola i valori SHAP
shap_values = explainer(tweets)

# Calcola anche le probabilità di classificazione (valori softmax)
probs = np.exp(f(tweets))  # restituisce probabilità vere

# Mostra i risultati per ogni tweet
for i, tweet in enumerate(tweets):
    print(f"\nTweet #{i+1}: {tweet}")
    print("Prediction probabilities:")
    for cls, p in zip(output_names, probs[i]):
        print(f"  {cls}: {p:.2f}")
    shap.plots.text(shap_values[i])
```

{"model_id":"6d3d4574ca95411dae1cf9e018ebc518","version_major":2,"vers
ion_minor":0}

```
PartitionExplainer explainer:  22%|██        | 11/50 [00:29<01:08,
1.77s/it]
```

```
{"model_id":"1d40de9963da41b8bf2b5b768380cdfc","version_major":2,"version_minor":0}

PartitionExplainer explainer:  48%|█████     | 24/50 [01:05<00:48, 1.87s/it]

{"model_id":"b7ec5bb9d95049ad9f2df9c6c326b36c","version_major":2,"version_minor":0}

PartitionExplainer explainer: 51it [02:15,  2.82s/it]


Tweet #1: 1st day at the pool on a beautiful sunday in ny too bad few
dared to go in due to the frigid water temp we lasted 45 min and
figured we proved our point
Prediction probabilities:
  Non-sexist: 0.99
  Sexist: 0.01

<IPython.core.display.HTML object>


Tweet #2: i like your outfit too except when i dress up as a frigid
bitch i try not to look so constipated is my new how are you
Prediction probabilities:
  Non-sexist: 0.40
  Sexist: 0.60

<IPython.core.display.HTML object>


Tweet #3:   same though the angst just comes and goes lonely xie lian
will always break my frigid heart
Prediction probabilities:
  Non-sexist: 0.98
  Sexist: 0.02

<IPython.core.display.HTML object>


Tweet #4:   fuck that cunt tried to vote her out multiple times
Prediction probabilities:
  Non-sexist: 0.16
  Sexist: 0.84

<IPython.core.display.HTML object>


Tweet #5:  u gotta say some shit like ill fuck that cunt dumb until
you cant remember anything but my name you stupid slut
Prediction probabilities:
  Non-sexist: 0.03
  Sexist: 0.97

<IPython.core.display.HTML object>


Tweet #6:  mmm fuck that cunt is hungry for this dick isnt it such a
shame that you can lie all you want about not wanting this but your
body never canhe says in her ear before he starts kissing down her
neck making sure to leave plenty of love marks on her pale skin
Prediction probabilities:
  Non-sexist: 0.03
  Sexist: 0.97

<IPython.core.display.HTML object>


Tweet #7: suika gets gangbanged to death
Prediction probabilities:
  Non-sexist: 0.87
  Sexist: 0.13

<IPython.core.display.HTML object>


Tweet #8: i mean i do but wouldnt it be fun to get gangbanged together

Prediction probabilities:
  Non-sexist: 0.87
  Sexist: 0.13

<IPython.core.display.HTML object>


Tweet #9:  maybe something involving her getting gangbanged
Prediction probabilities:
  Non-sexist: 0.91
  Sexist: 0.09

<IPython.core.display.HTML object>


Tweet #10:   sex as in gender harassment is what they are inferring
Prediction probabilities:
  Non-sexist: 0.56
  Sexist: 0.44

<IPython.core.display.HTML object>


Tweet #11:  kabul islamic emirate of afghanistan has announced the end
of coeducation in the country citing gender harassmentfemale students
will attend universities on monday wednesday and saturday while
```

```
tuesday thursday and sunday are fixed for males
Prediction probabilities:
  Non-sexist: 0.15
  Sexist: 0.85

<IPython.core.display.HTML object>


Tweet #12: appreciate being included along with  on the uvalde gunmans
history of cyber gender harassment regrettably unshocking
Prediction probabilities:
  Non-sexist: 0.95
  Sexist: 0.05

<IPython.core.display.HTML object>


Tweet #13:  please not the ones talking to 1st and 2nd  graders about
gender identity
Prediction probabilities:
  Non-sexist: 0.98
  Sexist: 0.02

<IPython.core.display.HTML object>


Tweet #14:  and  took the case of  to the  and won today the
government of  has committed to honor the sentencing which can
guarantees the rights to gender identity in
Prediction probabilities:
  Non-sexist: 1.00
  Sexist: 0.00

<IPython.core.display.HTML object>


Tweet #15:   gender gender identity amp sexual orientationparents
place a great deal of trust in the public school system to care for
the education of their childrenthat trust is being betrayed in a
shameful and deceitful way
Prediction probabilities:
  Non-sexist: 0.99
  Sexist: 0.01

<IPython.core.display.HTML object>


Tweet #16: the violent antifeminism of a farright movement that sees
principally as vessels for breeding a new white generation expresses
itself in a fixation on areturn to traditional gender rolesworth every
minute to read  take 5 min out of your sun
Prediction probabilities:
  Non-sexist: 0.53
  Sexist: 0.47

<IPython.core.display.HTML object>


Tweet #17:   ah look gender roles are back too the dont believe their
own lies
Prediction probabilities:
  Non-sexist: 0.48
  Sexist: 0.52

<IPython.core.display.HTML object>


Tweet #18:  yes hayley hafu representation give our japanesehispanic
girl her flowers
Prediction probabilities:
  Non-sexist: 0.97
  Sexist: 0.03

<IPython.core.display.HTML object>


Tweet #19: isa hot girl summer now pop yall shit
Prediction probabilities:
  Non-sexist: 0.90
  Sexist: 0.10

<IPython.core.display.HTML object>


Tweet #20: my baby called me mommy sha for the first time today twice
yall dont understand how hype that made me baby girl has autism and
getting her to talk without being prompted has been a challenge shes
come so far
Prediction probabilities:
  Non-sexist: 0.98
  Sexist: 0.02

<IPython.core.display.HTML object>


Tweet #21:  how did we reach a point where blonks are having the
audacity to talk about yoongis rap do yall see who you stan  are yall
ok  your girls cant write cant deliver cant come close to btss
lyricism tf are yall getting brave for  fashion deals
Prediction probabilities:
  Non-sexist: 0.18
  Sexist: 0.82

<IPython.core.display.HTML object>
```

```
Tweet #22:  absolutely brilliant wkend im off on july with the girls
cant wait
Prediction probabilities:
  Non-sexist: 0.96
  Sexist: 0.04

<IPython.core.display.HTML object>


Tweet #23: hot girls cant get nowhere without a gps
Prediction probabilities:
  Non-sexist: 0.07
  Sexist: 0.93

<IPython.core.display.HTML object>


Tweet #24:    that gold digger is nice but the picture and set up for
them are elite pinwork is like the revealing of the mona lisa always a
work of art
Prediction probabilities:
  Non-sexist: 0.98
  Sexist: 0.02

<IPython.core.display.HTML object>


Tweet #25: he tries to expose this gold digger but it backfires  90
day fiance b  via
Prediction probabilities:
  Non-sexist: 0.96
  Sexist: 0.04

<IPython.core.display.HTML object>


Tweet #26:  not enough is made of the fact that shes an unabashed gold
digger
Prediction probabilities:
  Non-sexist: 0.48
  Sexist: 0.52

<IPython.core.display.HTML object>


Tweet #27:  well let me ask for the guys do you ever just grab your
testicles because you can
Prediction probabilities:
  Non-sexist: 0.73
  Sexist: 0.27
```

```
<IPython.core.display.HTML object>


Tweet #28:  i cant imagine what the possible beef would have been but
you do realize that you two live close enough to each other that your
families could grab dinner together
Prediction probabilities:
  Non-sexist: 0.98
  Sexist: 0.02

<IPython.core.display.HTML object>


Tweet #29: there is still time to grab an  print for  use code 10off
at check out for 10 off one time use
Prediction probabilities:
  Non-sexist: 0.99
  Sexist: 0.01

<IPython.core.display.HTML object>


Tweet #30: they tried to censor threaten harassbut ultimately murdered
yeshua for saying what were not allowed to say online  he called them
children of the devil the synagogue of satanand not of his sheep
Prediction probabilities:
  Non-sexist: 0.98
  Sexist: 0.02

<IPython.core.display.HTML object>


Tweet #31:  u r allowing these terrorists to harass the countrymen
Prediction probabilities:
  Non-sexist: 0.91
  Sexist: 0.09

<IPython.core.display.HTML object>


Tweet #32: dear  fans can we not harass talented actors of color stop
being racist douche nozzles this is why we cant have nice things moses
is a badass inquisitor and yall cant handle her badassary
Prediction probabilities:
  Non-sexist: 0.95
  Sexist: 0.05

<IPython.core.display.HTML object>


Tweet #33: the cosmic hegemony is taking over and its your job to find
their superweapons before its too late my review of the darn difficult
```

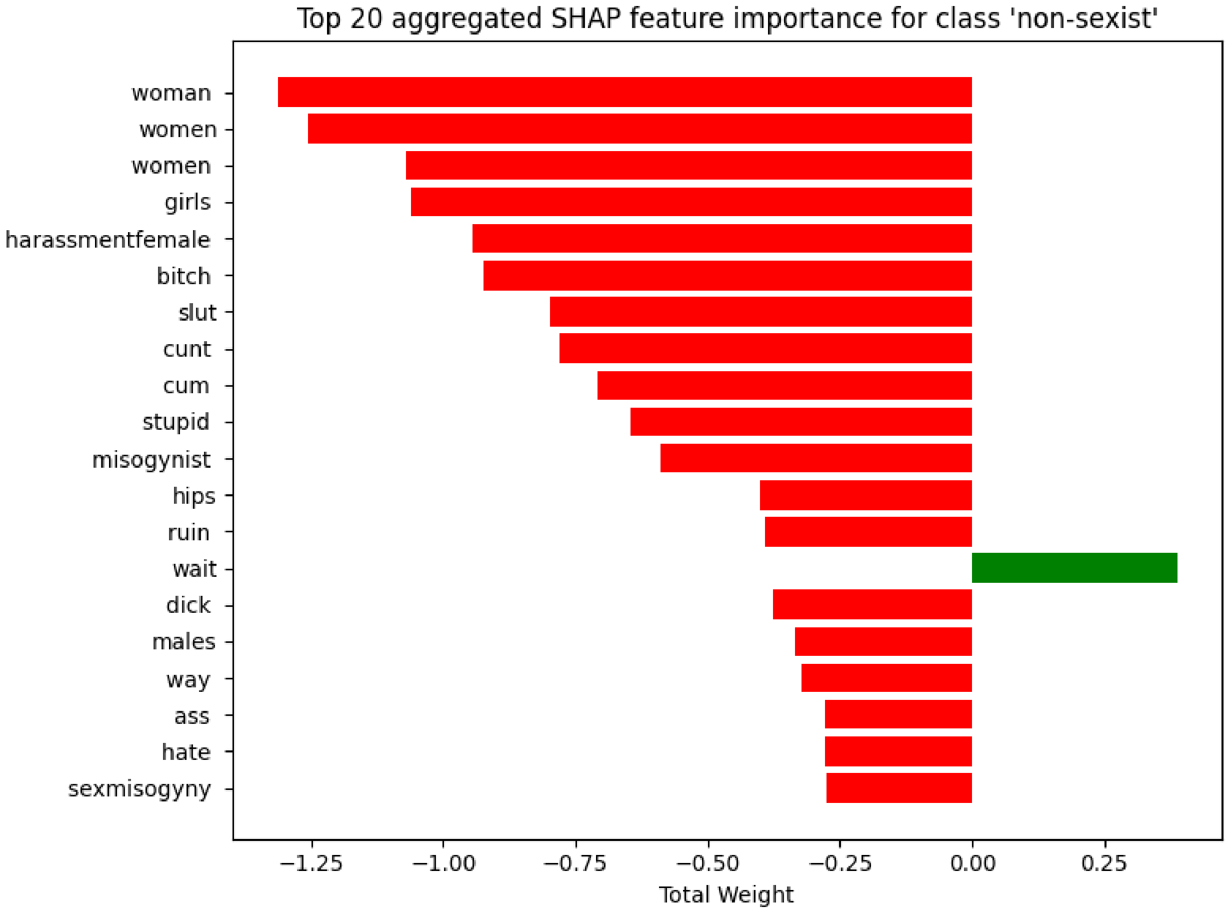
```
but beautifully themed  solo game pulp invasion
Prediction probabilities:
  Non-sexist: 0.96
  Sexist: 0.04

<IPython.core.display.HTML object>


Tweet #34:  the key is whats your view on prof mearsheimers view  do
you agree with him or not china would like to be the hegemony in asia
pacific is china working hard to displace usa from asia pacific
Prediction probabilities:
  Non-sexist: 0.97
  Sexist: 0.03

<IPython.core.display.HTML object>


Tweet #35:    yes they operate covertly as fascists you can silence
opposition engage in psychological warfare and broaden hegemony
without doing it openly
Prediction probabilities:
  Non-sexist: 0.96
  Sexist: 0.04

<IPython.core.display.HTML object>


Tweet #36: so deep painfully deep our history and herstory of slave
trade colonialism and neocolonialism
Prediction probabilities:
  Non-sexist: 0.99
  Sexist: 0.01

<IPython.core.display.HTML object>


Tweet #37: met the talented  at cre8tivecon la today she has shown me
what disciple and grit and can get you im so glad to have heard her
story today
Prediction probabilities:
  Non-sexist: 0.99
  Sexist: 0.01

<IPython.core.display.HTML object>


Tweet #38: stop saying i said this cus she has short hair some of her
gayest looks came from 201718 blonde ponytail era know ur herstory
Prediction probabilities:
  Non-sexist: 0.91
  Sexist: 0.09
```

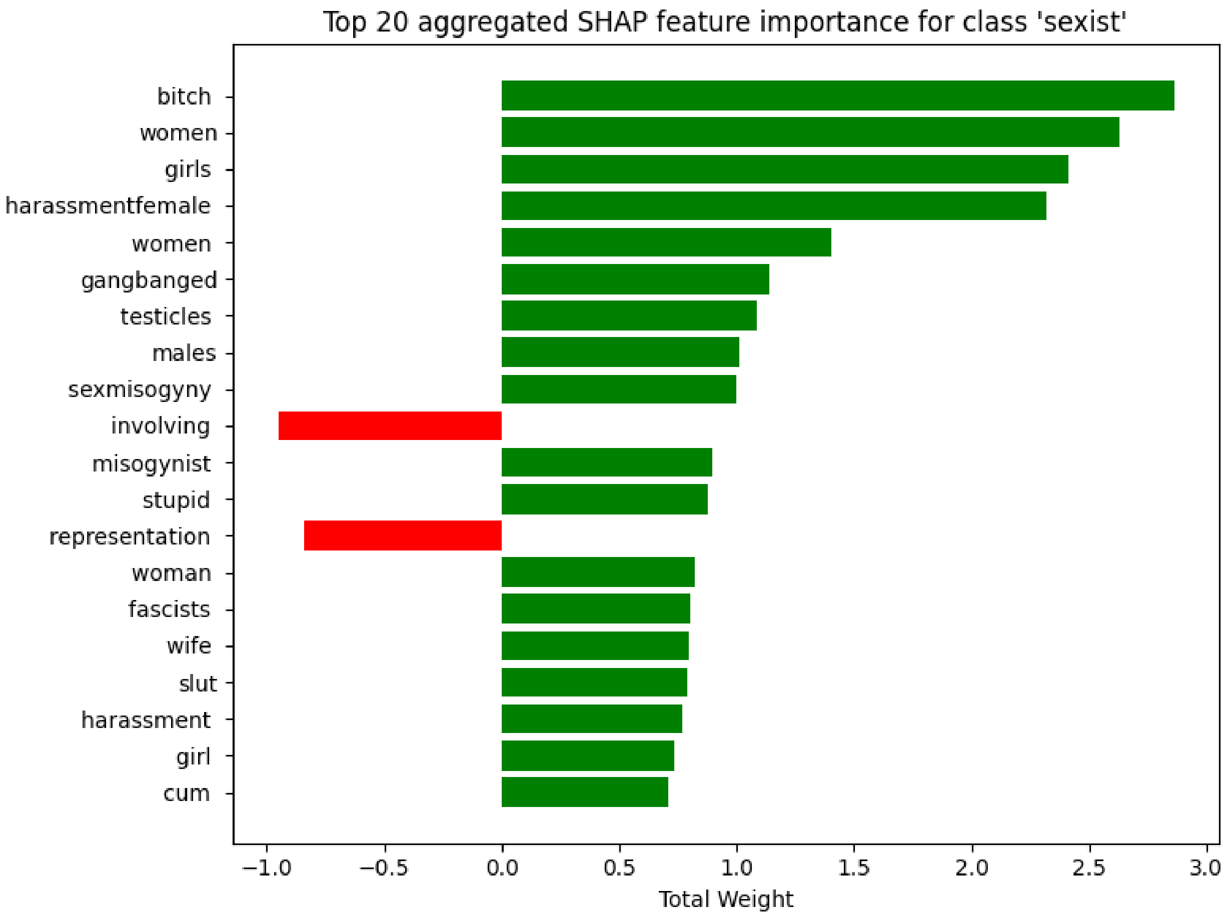
```
<IPython.core.display.HTML object>


Tweet #39:  best way to do that is from behind her with your hands on
her hips
Prediction probabilities:
  Non-sexist: 0.12
  Sexist: 0.88

<IPython.core.display.HTML object>


Tweet #40: some one get those white boy hips moving man cannot bounce
Prediction probabilities:
  Non-sexist: 0.85
  Sexist: 0.15

<IPython.core.display.HTML object>


Tweet #41: same energy as kiryu couldnt be in smash because he wouldnt
hit women it is a noncanon crossover who gives a fuck
Prediction probabilities:
  Non-sexist: 0.14
  Sexist: 0.86

<IPython.core.display.HTML object>


Tweet #42:  exactly its drilled into men not to hit women why isnt it
drilled into women not to hit men so many times on nights out i have
witnessed women hitting punching scratching at men and them standing
there and just taking it as they know they are damned if they react
Prediction probabilities:
  Non-sexist: 0.02
  Sexist: 0.98

<IPython.core.display.HTML object>


Tweet #43: why would a woman accuse someone of hitting her if it was
lies i dont hit women and never been accused of sucheven my exnever
said that in a nasty divorce
Prediction probabilities:
  Non-sexist: 0.02
  Sexist: 0.98

<IPython.core.display.HTML object>


Tweet #44:  whats the deal did my husband and i just take the same
test one was bought off the website and the other from
```

```
Prediction probabilities:
  Non-sexist: 0.99
  Sexist: 0.01

<IPython.core.display.HTML object>


Tweet #45:  yall are husband and wife and best friends you guys have
so much fun its awesome you keep each other laughing and smiling
Prediction probabilities:
  Non-sexist: 0.99
  Sexist: 0.01

<IPython.core.display.HTML object>


Tweet #46:  wait till ur husband messes up u gonna love the scouser
death threats
Prediction probabilities:
  Non-sexist: 0.92
  Sexist: 0.08

<IPython.core.display.HTML object>


Tweet #47: aughhh i still got an exam tomorrow i hate women
Prediction probabilities:
  Non-sexist: 0.05
  Sexist: 0.95

<IPython.core.display.HTML object>


Tweet #48: i am not like other queers i hate women misogynist and
proud
Prediction probabilities:
  Non-sexist: 0.31
  Sexist: 0.69

<IPython.core.display.HTML object>


Tweet #49: ill share some stuff when i archive but antifujo comment
threads are 100im  is it ok for me to make mlmyesnoyou can as long as
no sexmisogyny is a liei hate women they ruin everythingim a trans man
amp cant believe i used to think i was a dirty gross fujo
Prediction probabilities:
  Non-sexist: 0.06
  Sexist: 0.94

<IPython.core.display.HTML object>
```

```
Tweet #50:  i want to give you delicious black kiss your ass looks
very rich then i would like to fuck you and cum inside you
Prediction probabilities:
  Non-sexist: 0.07
  Sexist: 0.93

<IPython.core.display.HTML object>
```

```python
import matplotlib.pyplot as plt
import numpy as np
from collections import defaultdict, Counter

# Normalizza i nomi delle classi a minuscolo
output_names = [name.lower() for name in output_names]

# Inizializza dizionari per sommare i valori SHAP e contare le
occorrenze per ciascun token
shap_token_sums = {
    'sexist': defaultdict(float),
    'non-sexist': defaultdict(float)
}
shap_token_counts = {
    'sexist': defaultdict(int),
    'non-sexist': defaultdict(int)
}

# Analizza tutti i tweet
for i in range(len(tweets)):
    for class_idx, class_label in enumerate(output_names):
        shap_vals = shap_values[i].values[:, class_idx]  # valori SHAP
per la classe
        tokens = shap_values[i].data

        for token, val in zip(tokens, shap_vals):
            shap_token_sums[class_label][token] += val
            shap_token_counts[class_label][token] += 1

# Calcola la media dei valori SHAP per ciascun token
shap_token_means = {
    class_label: {
        token: shap_token_sums[class_label][token] /
shap_token_counts[class_label][token]
        for token in shap_token_sums[class_label]
    }
    for class_label in output_names
}

# Funzione per il plotting
def plot_top_tokens(token_means, class_label):
    # Prendi i 20 token con valore SHAP medio assoluto più alto
    top_tokens = sorted(token_means.items(), key=lambda x: abs(x[1]),
reverse=True)[:20]
    tokens, means = zip(*top_tokens)

    colors = ['green' if val > 0 else 'red' for val in means]

    plt.figure(figsize=(8, 6))
    plt.barh(tokens[::-1], means[::-1], color=colors[::-1])
    plt.xlabel("Total Weight")
    plt.title(f"Top 20 aggregated SHAP feature importance for class
'{class_label}'")
    plt.tight_layout()
    plt.show()

# Plot per ciascuna classe
for class_label in output_names:
    plot_top_tokens(shap_token_means[class_label], class_label)
```




Top 20 aggregated SHAP feature importance for class 'sexist'
bitch
women
girls
harassmentfemale
women
gangbanged
testicles
males
sexmisogyny
involving
misogynist
stupid
representation
woman
fascists
wife
slut
harassment
girl
cum
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Total Weight