Mitigating Unintended Identity Bias in Toxicity Detection: Research Question 1

Amirhossein Hajigholam Saryazdi

Trustworthy Machine Learning, Fall 22

```
# Connecting the notebook to my google drive

from google.colab import drive
drive.mount('/content/drive', force_remount=True)
```

```
    Mounted at /content/drive
```

0 Loading and preprocessing Main Data:

```
import pandas as pd

data_main = pd.read_csv("/content/drive/My Drive/Trustworthy ML/all_data.csv")
```

```
data_main.shape
```

```
    (1999516, 46)
```

```
# train/test split was provided in the data by labels: 90 / 10
train = data_main[data_main['split']=='train']
test = data_main[data_main['split']=='test']

print(r' Train set:',train.shape)
print(r' Test set:',test.shape)
```

```
     Train set: (1804875, 46)
     Test set: (194641, 46)
```

```
# number of toxic and non-toxic comments in the train set
train = train[['comment_text','toxicity','severe_toxicity', 'obscene', 'identity_attack', 'insult', 'threat']]

train_tox = train[train['toxicity'] >= 0.5]
train_nontox = train[train['toxicity'] < 0.5]

print(r' Toxic comments in the train set:', train_tox.shape[0])
print(r' Non-toxic comments in the train set:', train_nontox.shape[0])
```

```
     Toxic comments in the train set: 144334
     Non-toxic comments in the train set: 1660541
```

```
# Frequency of identity terms in toxic and non-toxic train data

identities = ['lesbian','gay','bisexual','transgender','queer','homosexual','transsexual','black']

for i in identities:
  print(i)

  print(r'frequency in train_tox:  ', train_tox['comment_text'].str.contains(i, case=False, regex=False).value_counts()[1]/train_t

  print(r'frequency in train_nontox:  ', train_nontox['comment_text'].str.contains(i, case=False, regex=False).value_counts()[1]/t

  print(r'-------------------------------------------------------')
```

```
    lesbian
    frequency in train_tox:   0.0018498759820970805
    frequency in train_nontox:   0.00031194652827000354
    -------------------------------------------------------
    gay
    frequency in train_tox:   0.015997616639184115
    frequency in train_nontox:   0.0033464997250895944
    -------------------------------------------------------
    bisexual
    frequency in train_tox:   0.00019399448501392603
    frequency in train_nontox:   6.564125787920924e-05
```

```
-------------------------------------------------------
transgender
frequency in train_tox:   0.0031177685091523825
frequency in train_nontox:   0.0010906084221949353
-------------------------------------------------------
queer
frequency in train_tox:   0.0003533470977039367
frequency in train_nontox:   8.912757950571531e-05
-------------------------------------------------------
homosexual
frequency in train_tox:   0.005528842822896892
frequency in train_nontox:   0.0013682287880877376
-------------------------------------------------------
transsexual
frequency in train_tox:   9.699724250696302e-05
frequency in train_nontox:   3.312173562712393e-05
-------------------------------------------------------
black
frequency in train_tox:   0.03783585295218036
frequency in train_nontox:   0.011227063950844935
-------------------------------------------------------
```

```
for i in range(0,7):

  print(identities[i])
  x = train_nontox[train_nontox['comment_text'].str.contains(identities[i], case=False, regex=False)==True]
  print(r'description:  ', x['comment_text'].str.len().describe())
  print(r'-------------------------------------------------------')
```

```
max      1000.000000
Name: comment_text, dtype: float64
-------------------------------------------------------
bisexual
description:   count     109.000000
mean      555.009174
std       302.144419
min        19.000000
25%       278.000000
50%       509.000000
75%       877.000000
max      1000.000000
Name: comment_text, dtype: float64
-------------------------------------------------------
transgender
description:   count    1811.000000
mean      515.979017
std       297.370819
min        21.000000
25%       268.000000
50%       461.000000
75%       783.500000
max      1000.000000
Name: comment_text, dtype: float64
-------------------------------------------------------
queer
description:   count     148.000000
mean      506.277027
std       331.463295
min         7.000000
25%       226.250000
50%       394.000000
75%       853.250000
max      1000.000000
Name: comment_text, dtype: float64
```

```
    75%      764.000000
    max      998.000000
    Name: comment_text, dtype: float64
    -------------------------------------------------------
```

```
# sub-sectioning identity comments in main data train non-toxic

train_nontox_black = train_nontox[train_nontox['comment_text'].str.contains('black', case=False, regex=False)==True]

for i in range(0,7):

  train_nontox_black = train_nontox_black.append(train_nontox[train_nontox['comment_text'].str.contains(identities[i], case=False,

train_nontox_identities = train_nontox_black

train_nontox_identities
```

	comment_text	toxicity	severe_toxicity	obscene	identity_attack	insult	threat
455	Uh hmmm, stepping into the "21st Century," rig...	0.142857	0.000000	0.000000	0.142857	0.142857	0.000000
539	I can understand how a "display of wealth" mig...	0.454545	0.090909	0.000000	0.272727	0.454545	0.000000
700	People think wolves are nothing but big cute d...	0.385714	0.042857	0.014286	0.057143	0.071429	0.342857
834	So if I read you correctly, it’s their fault f...	0.142857	0.000000	0.000000	0.000000	0.142857	0.000000
1072	The limits of the far Right are the same as th...	0.142857	0.000000	0.000000	0.000000	0.142857	0.142857
...	...	...	...	...	...	...	...
1977526	Exposure to nudity of the opposite gender in p...	0.400000	0.100000	0.100000	0.200000	0.300000	0.000000
1991813	When have you ever heard anyone ask a transsex...	0.200000	0.000000	0.000000	0.200000	0.000000	0.000000
1998846	You have decided, based on nothing more than y...	0.400000	0.100000	0.000000	0.200000	0.400000	0.000000
1999045	really please elucidate on your education then...	0.400000	0.000000	0.000000	0.200000	0.400000	0.000000
1999182	The Church has not said that being gay is a si...	0.400000	0.000000	0.000000	0.300000	0.200000	0.000000

29113 rows × 7 columns

```
#·distribution·of·non_toxic·comments·length
train_nontox["comment_len"]=·train_nontox["comment_text"].str.len()

train_nontox['comment_len'].describe()
```

```
    <ipython-input-11-f58fcedfde4b>:2: SettingWithCopyWarning:
    A value is trying to be set on a copy of a slice from a DataFrame.
    Try using .loc[row_indexer,col_indexer] = value instead

    See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-
      train_nontox["comment_len"]= train_nontox["comment_text"].str.len()
    count    1.660540e+06
    mean     2.989398e+02
    std      2.710667e+02
    min      1.000000e+00
    25%      9.400000e+01
    50%      2.030000e+02
    75%      4.190000e+02
    max      1.906000e+03
    Name: comment_len, dtype: float64
```

0.1 Balancing Train set wrt certain identities

Adding 9871 non-toxic comments containing LGBTQ and Black identity terms from Reddit, Youtube, and Twitter to have a more balanced training data.

```
pip install datasets
```

```
    Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
    Collecting datasets
      Downloading datasets-2.7.1-py3-none-any.whl (451 kB)
         |████████████████████████████████| 451 kB 35.0 MB/s
    Collecting huggingface-hub<1.0.0,>=0.2.0
      Downloading huggingface_hub-0.11.1-py3-none-any.whl (182 kB)
         |████████████████████████████████| 182 kB 74.5 MB/s
    Collecting responses<0.19
      Downloading responses-0.18.0-py3-none-any.whl (38 kB)
```

```
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.8/dist-packages (from datasets) (1.21.6)
Requirement already satisfied: fsspec[http]>=2021.11.1 in /usr/local/lib/python3.8/dist-packages (from datasets) (2022.11.0)
Collecting xxhash
  Downloading xxhash-3.1.0-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (212 kB)
     |████████████████████████████████| 212 kB 77.2 MB/s
Requirement already satisfied: pandas in /usr/local/lib/python3.8/dist-packages (from datasets) (1.3.5)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.8/dist-packages (from datasets) (6.0)
Requirement already satisfied: tqdm>=4.62.1 in /usr/local/lib/python3.8/dist-packages (from datasets) (4.64.1)
Requirement already satisfied: requests>=2.19.0 in /usr/local/lib/python3.8/dist-packages (from datasets) (2.23.0)
Requirement already satisfied: pyarrow>=6.0.0 in /usr/local/lib/python3.8/dist-packages (from datasets) (9.0.0)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.8/dist-packages (from datasets) (3.8.3)
Requirement already satisfied: packaging in /usr/local/lib/python3.8/dist-packages (from datasets) (21.3)
Collecting multiprocess
  Downloading multiprocess-0.70.14-py38-none-any.whl (132 kB)
     |████████████████████████████████| 132 kB 76.6 MB/s
Requirement already satisfied: dill<0.3.7 in /usr/local/lib/python3.8/dist-packages (from datasets) (0.3.6)
Requirement already satisfied: async-timeout<5.0,>=4.0.0a3 in /usr/local/lib/python3.8/dist-packages (from aiohttp->datasets)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.8/dist-packages (from aiohttp->datasets) (22.1.0)
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.8/dist-packages (from aiohttp->datasets) (1.8.2)
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.8/dist-packages (from aiohttp->datasets) (6.0.3)
Requirement already satisfied: charset-normalizer<3.0,>=2.0 in /usr/local/lib/python3.8/dist-packages (from aiohttp->datasets
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.8/dist-packages (from aiohttp->datasets) (1.3.3)
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.8/dist-packages (from aiohttp->datasets) (1.3.1)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.8/dist-packages (from huggingface-hub<1.0
Requirement already satisfied: filelock in /usr/local/lib/python3.8/dist-packages (from huggingface-hub<1.0.0,>=0.2.0->datase
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/python3.8/dist-packages (from packaging->datasets)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.8/dist-packages (from requests>=2.19.0->datasets) (2.10
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.8/dist-packages (from requests>=2.19.0->datasets)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.8/dist-packages (from requests>=2.19.0->datasets)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.8/dist-packages (from reques
Collecting urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1
  Downloading urllib3-1.25.11-py2.py3-none-any.whl (127 kB)
     |████████████████████████████████| 127 kB 65.6 MB/s
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.8/dist-packages (from pandas->datasets) (2022.6)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.8/dist-packages (from pandas->datasets) (2.8.
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.8/dist-packages (from python-dateutil>=2.7.3->pandas->datas
Installing collected packages: urllib3, xxhash, responses, multiprocess, huggingface-hub, datasets
  Attempting uninstall: urllib3
    Found existing installation: urllib3 1.24.3
    Uninstalling urllib3-1.24.3:
      Successfully uninstalled urllib3-1.24.3
Successfully installed datasets-2.7.1 huggingface-hub-0.11.1 multiprocess-0.70.14 responses-0.18.0 urllib3-1.25.11 xxhash-3.1
```

```
import datasets
dataset = datasets.load_dataset('ucberkeley-dlab/measuring-hate-speech', 'binary')
df = dataset['train'].to_pandas()
```

```
Downloading readme: 100%                                   4.03k/4.03k [00:00<00:00, 108kB/s]

WARNING:datasets.builder:Using custom data configuration ucberkeley-dlab--measuring-hate-speech-c32713cabe528196
Downloading and preparing dataset parquet/ucberkeley-dlab--measuring-hate-speech to /root/.cache/huggingface/datasets/ucberke

Downloading data files: 100%                               1/1 [00:00<00:00, 1.65it/s]

Downloading data: 100%                                     14.1M/14.1M [00:00<00:00, 44.2MB/s]

Extracting data files: 100%                                1/1 [00:00<00:00, 32.26it/s]

Dataset parquet downloaded and prepared to /root/.cache/huggingface/datasets/ucberkeley-dlab___parquet/ucberkeley-dlab--measu
100%                                          1/1 [00:00<00:00, 6.20it/s]
```

```
# according to the source, score under 0.5 is not hateful, but we go with smaller than -1 to be on the safe side.

hug_nontox = df[df['hate_speech_score']<-1]

identities = ['lesbian','gay','bisexual','transgender','queer','homosexual','transsexual','black']

hug_nontox.rename(columns = {'text':'comment_text', 'hate_speech_score':'toxicity'}, inplace = True)

# setting the toxicity value to 0
hug_nontox['toxicity'] = 0

# finding the frequency of lgbtq and black identity in the non-toxic subset of data

for i in identities:
  print(i)
  print(r'frequency in hug_nontox:  ', hug_nontox['comment_text'].str.contains(i, case=False, regex=False).value_counts()[1])
  print(r'-------------------------------------------------------')
```

```
/usr/local/lib/python3.8/dist-packages/pandas/core/frame.py:5039: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-
  return super().rename(
<ipython-input-14-bb69ac2fa1ad>:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-
  hug_nontox['toxicity'] = 0
lesbian
frequency in hug_nontox:   677
-------------------------------------------------------
gay
frequency in hug_nontox:   3353
-------------------------------------------------------
bisexual
frequency in hug_nontox:   414
-------------------------------------------------------
transgender
frequency in hug_nontox:   833
-------------------------------------------------------
queer
frequency in hug_nontox:   374
-------------------------------------------------------
homosexual
frequency in hug_nontox:   230
-------------------------------------------------------
transsexual
frequency in hug_nontox:   8
-------------------------------------------------------
black
frequency in hug_nontox:   3982
-------------------------------------------------------
```

```
# appending lgbtq identities to the black to have a single dataset of nontoxic identity

hug_nontox_black = hug_nontox[hug_nontox['comment_text'].str.contains('black', case=False, regex=False)==True]

for i in range(0,7):

  hug_nontox_black = hug_nontox_black.append(hug_nontox[hug_nontox['comment_text'].str.contains(identities[i], case=False, regex=F

hug_nontox_identities = hug_nontox_black

hug_nontox_identities
```

	comment_id	annotator_id	platform	sentiment	respect	insult	humiliate	status	dehumanize	violence	...	annotator_r
5	11001	527	0	1.0	1.0	0.0	0.0	2.0	0.0	0.0	...	
22	15758	8749	0	2.0	2.0	2.0	0.0	2.0	0.0	0.0	...	
56	42490	3123	3	2.0	2.0	2.0	2.0	2.0	2.0	0.0	...	
112	38840	9482	2	3.0	3.0	0.0	0.0	1.0	1.0	0.0	...	
130	3101	1379	0	2.0	2.0	0.0	0.0	2.0	0.0	0.0	...	
...	...	...	...	...	...	...	...	...	...	...	...	
59174	2176	7993	0	1.0	1.0	0.0	0.0	2.0	0.0	0.0	...	
67380	13643	880	0	2.0	2.0	2.0	2.0	2.0	0.0	0.0	...	
69420	34703	684	2	4.0	4.0	4.0	4.0	2.0	1.0	0.0	...	
79956	2176	7902	0	1.0	1.0	1.0	1.0	2.0	1.0	0.0	...	
89254	13643	11128	0	0.0	0.0	0.0	0.0	2.0	0.0	0.0	...	

9871 rows × 131 columns

```
# distribution of non_toxic identitty comments length in hug set
hug_nontox_identities["comment_len"]= hug_nontox_identities["comment_text"].str.len()

hug_nontox_identities['comment_len'].describe()
```

```
count    9871.000000
mean      171.444838
```

```
std       132.252589
min         8.000000
25%        57.000000
50%       133.000000
75%       265.000000
max       602.000000
Name: comment_len, dtype: float64
```

```
hug_nontox["comment_text"].str.len().describe()
```

```
count    53651.000000
mean       165.446683
std        128.458174
min          7.000000
25%         68.000000
50%        130.000000
75%        225.000000
max        603.000000
Name: comment_text, dtype: float64
```

```
# merging the balancer data with main data

hug_nontox_identities = hug_nontox_identities[['comment_text','toxicity']]
train = train[['comment_text','toxicity','severe_toxicity', 'obscene', 'identity_attack', 'insult', 'threat']]

train_b1 = pd.concat([train,hug_nontox_identities])

#train_b1.reset_index()
```

```
train_b1 = train_b1.fillna(0)
train_b1.isna().sum()
```

```
comment_text       0
toxicity           0
severe_toxicity    0
obscene            0
identity_attack    0
insult             0
threat             0
dtype: int64
```

0.2 Random Control train set

Randomly adding the same amount of non toxic comment as we added from certain identity to compare the effect on toxicty regarding the identities.

```
pip install datasets
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting datasets
  Downloading datasets-2.7.1-py3-none-any.whl (451 kB)
     |████████████████████████████████| 451 kB 33.0 MB/s
Requirement already satisfied: fsspec[http]>=2021.11.1 in /usr/local/lib/python3.7/dist-packages (from datasets) (2022.11.0)
Requirement already satisfied: requests>=2.19.0 in /usr/local/lib/python3.7/dist-packages (from datasets) (2.23.0)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.7/dist-packages (from datasets) (6.0)
Requirement already satisfied: pyarrow>=6.0.0 in /usr/local/lib/python3.7/dist-packages (from datasets) (9.0.0)
Requirement already satisfied: importlib-metadata in /usr/local/lib/python3.7/dist-packages (from datasets) (4.13.0)
Collecting xxhash
  Downloading xxhash-3.1.0-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (212 kB)
     |████████████████████████████████| 212 kB 71.7 MB/s
Requirement already satisfied: packaging in /usr/local/lib/python3.7/dist-packages (from datasets) (21.3)
Requirement already satisfied: dill<0.3.7 in /usr/local/lib/python3.7/dist-packages (from datasets) (0.3.6)
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (from datasets) (1.3.5)
Collecting huggingface-hub<1.0.0,>=0.2.0
  Downloading huggingface_hub-0.11.0-py3-none-any.whl (182 kB)
     |████████████████████████████████| 182 kB 74.7 MB/s
Requirement already satisfied: tqdm>=4.62.1 in /usr/local/lib/python3.7/dist-packages (from datasets) (4.64.1)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.7/dist-packages (from datasets) (3.8.3)
Collecting responses<0.19
  Downloading responses-0.18.0-py3-none-any.whl (38 kB)
Collecting multiprocess
  Downloading multiprocess-0.70.14-py37-none-any.whl (115 kB)
     |████████████████████████████████| 115 kB 73.2 MB/s
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.7/dist-packages (from datasets) (1.21.6)
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.7/dist-packages (from aiohttp->datasets) (1.8.1)
Requirement already satisfied: asynctest==0.13.0 in /usr/local/lib/python3.7/dist-packages (from aiohttp->datasets) (0.13.0)
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.7/dist-packages (from aiohttp->datasets) (1.3.3)
Requirement already satisfied: typing-extensions>=3.7.4 in /usr/local/lib/python3.7/dist-packages (from aiohttp->datasets) (4
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.7/dist-packages (from aiohttp->datasets) (6.0.2
Requirement already satisfied: async-timeout<5.0,>=4.0.0a3 in /usr/local/lib/python3.7/dist-packages (from aiohttp->datasets)
Requirement already satisfied: charset-normalizer<3.0,>=2.0 in /usr/local/lib/python3.7/dist-packages (from aiohttp->datasets
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.7/dist-packages (from aiohttp->datasets) (22.1.0)
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.7/dist-packages (from aiohttp->datasets) (1.3.1)
Requirement already satisfied: filelock in /usr/local/lib/python3.7/dist-packages (from huggingface-hub<1.0.0,>=0.2.0->datase
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/python3.7/dist-packages (from packaging->datasets)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests>=2.19.0->datasets) (2.1
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests>=2.19.0->datasets)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests>=2.19.0->datasets)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from reque
Collecting urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1
  Downloading urllib3-1.25.11-py2.py3-none-any.whl (127 kB)
     |████████████████████████████████| 127 kB 69.0 MB/s
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (from importlib-metadata->datasets) (3.10.
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-packages (from pandas->datasets) (2.8.
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (from pandas->datasets) (2022.6)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.7.3->pandas->datas
Installing collected packages: urllib3, xxhash, responses, multiprocess, huggingface-hub, datasets
  Attempting uninstall: urllib3
    Found existing installation: urllib3 1.24.3
    Uninstalling urllib3-1.24.3:
      Successfully uninstalled urllib3-1.24.3
Successfully installed datasets-2.7.1 huggingface-hub-0.11.0 multiprocess-0.70.14 responses-0.18.0 urllib3-1.25.11 xxhash-3.1
```

```python
import datasets
dataset = datasets.load_dataset('ucberkeley-dlab/measuring-hate-speech', 'binary')
df = dataset['train'].to_pandas()
```

```
Downloading readme: 100%                                  4.03k/4.03k [00:00<00:00, 109kB/s]
WARNING:datasets.builder:Using custom data configuration ucberkeley-dlab--measuring-hate-speech-cd96c7d7a29268f7
Downloading and preparing dataset parquet/ucberkeley-dlab--measuring-hate-speech to /root/.cache/huggingface/datasets/ucberke
Downloading data files: 100%                                  1/1 [00:02<00:00, 2.96s/it]

Downloading data: 100%                                  14.1M/14.1M [00:00<00:00, 18.0MB/s]

Extracting data files: 100%                                  1/1 [00:00<00:00, 11.09it/s]


Dataset parquet downloaded and prepared to /root/.cache/huggingface/datasets/ucberkeley-dlab___parquet/ucberkeley-dlab--measu
100%                                  1/1 [00:00<00:00, 4.66it/s]
```

```python
# according to the source, score under 0.5 is not hateful, but we go with smaller than -1 to be on the safe side.

hug_nontox = df[df['hate_speech_score']<-1]

hug_nontox.rename(columns = {'text':'comment_text', 'hate_speech_score':'toxicity'}, inplace = True)

# setting the toxicity value to 0
hug_nontox['toxicity'] = 0

# English only
hug_nontox_en = hug_nontox[hug_nontox['comment_text'].map(lambda x: x.isascii())]

# cheking the comment length
hug_nontox_en["comment_len"]= hug_nontox_en["comment_text"].str.len()

# sampling
hug_nontox_en_s = hug_nontox_en.sample(n=9871, random_state=1234)

# distribution of comment length
hug_nontox_en['comment_len'].describe()
```

```
/usr/local/lib/python3.7/dist-packages/pandas/core/frame.py:5047: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-
  errors=errors,
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:14: SettingWithCopyWarning:
```

```
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-

count    47654.000000
mean       167.303353
std        131.900158
min          7.000000
25%         65.000000
50%        130.000000
75%        229.000000
max        603.000000
Name: comment_len, dtype: float64
```

Double-click (or enter) to edit

```
# merging the random non_toxic comments  with main data

hug_nontox_en_s = hug_nontox_en_s[['comment_text','toxicity']]
train = train[['comment_text','toxicity','severe_toxicity', 'obscene', 'identity_attack', 'insult', 'threat']]

train_b2 = pd.concat([train,hug_nontox_en_s])
```

```
train_b2 = train_b2.fillna(0)
train_b2.isna().sum()
```

```
comment_text       0
toxicity           0
severe_toxicity    0
obscene            0
identity_attack    0
insult             0
threat             0
dtype: int64
```

1 Loading Model: simple_lstm_pytorch_version

https://www.kaggle.com/code/bminixhofer/simple-lstm-pytorch-version/notebook

```
import numpy as np
import random
import pandas as pd
import os
import time
import gc
import random
from tqdm._tqdm_notebook import tqdm_notebook as tqdm
from keras.preprocessing import text, sequence
from keras.utils import pad_sequences
import torch
from torch import nn
from torch.utils import data
from torch.nn import functional as F

device = torch.device("cuda")
torch.cuda.empty_cache()
```

```
<ipython-input-20-a31f7c147919>:8: TqdmDeprecationWarning: This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.*` instead of `tqdm._tqdm_notebook.*`
  from tqdm._tqdm_notebook import tqdm_notebook as tqdm
```

```
random.seed(1234)

# disable progress bars when submitting
def is_interactive():
   return 'SHLVL' not in os.environ

if not is_interactive():
    def nop(it, *a, **k):
        return it
    tqdm = nop

###

def seed_everything(seed=1234):
    random.seed(seed)
    os.environ['PYTHONHASHSEED'] = str(seed)
    np.random.seed(seed)
    torch.manual_seed(seed)
    torch.cuda.manual_seed(seed)
    torch.backends.cudnn.deterministic = True
seed_everything()

################## I do not know what "NUM_MODELS" is. It is 2 in original code and lead to the training of 2 models.

# Embeddings, note that these two pre-trained models give 300d vectors
CRAWL_EMBEDDING_PATH = '/content/drive/My Drive/Trustworthy ML/crawl-300d-2M.vec'
GLOVE_EMBEDDING_PATH = '/content/drive/My Drive/Trustworthy ML/glove.840B.300d.txt'
NUM_MODELS = 1
# it it the dimension of the output vector of each LSTM cell.
LSTM_UNITS = 128
DENSE_HIDDEN_UNITS = 4 * LSTM_UNITS
#we will convert each word in a comment_text to a number.
#So a comment_text is a list of number. How many numbers in this list?
#we want the length of this list is a constant -> MAX_LEN
#As the sentences are vary in length, they must be padded/truncated into a fixed length -> MAX_LEN
MAX_LEN = 220

###

#each line in the file looks like
    # apple 0.3 0.4 0.5 0.6 ...
    # that is a word followed by 300 float numbers
def get_coefs(word, *arr):
    return word, np.asarray(arr, dtype='float32')

def load_embeddings(path):
    with open(path) as f:
        return dict(get_coefs(*line.strip().split(' ')) for line in tqdm(f))

#embedding_matrix is a matrix of len(word_index)+1  x 300
# word_index is a dict. Each element is (word:i) where i is the index of the word
#word_index is a dict of the form ('apple': 123, 'banana': 349, etc)
# that means word_index[word] gives the index of the word
# word_index was built from all commment_texts
#path: a path that contains embedding matrix

def build_matrix(word_index, path):
    embedding_index = load_embeddings(path)
    embedding_matrix = np.zeros((len(word_index) + 1, 300))
    unknown_words = []

    for word, i in word_index.items():
        try:
            embedding_matrix[i] = embedding_index[word]
        except KeyError:
            unknown_words.append(word)
    return embedding_matrix, unknown_words

####################### model hyper-parameters can be modified to accelarte the training.

def sigmoid(x):
    return 1 / (1 + np.exp(-x))

def train_model(model, train, test, loss_fn, output_dim, lr=0.001,
                batch_size=512, n_epochs=4,
                enable_checkpoint_ensemble=True):
    param_lrs = [{'params': param, 'lr': lr} for param in model.parameters()]
    optimizer = torch.optim.Adam(param_lrs, lr=lr)

    scheduler = torch.optim.lr_scheduler.LambdaLR(optimizer, lambda epoch: 0.6 ** epoch)

    train_loader = torch.utils.data.DataLoader(train, batch_size=batch_size, shuffle=True)
    test_loader = torch.utils.data.DataLoader(test, batch_size=batch_size, shuffle=False)
    all_test_preds = []
    checkpoint_weights = [2 ** epoch for epoch in range(n_epochs)]
    for epoch in range(n_epochs):
        start_time = time.time()

        scheduler.step()

        model.train()
        avg_loss = 0.

        for data in tqdm(train_loader, disable=False):
            x_batch = data[:-1]
            y_batch = data[-1]

            y_pred = model(*x_batch)
            loss = loss_fn(y_pred, y_batch)

            optimizer.zero_grad()
            loss.backward()

            optimizer.step()
            avg_loss += loss.item() / len(train_loader)

        model.eval()
        test_preds = np.zeros((len(test), output_dim))

        for i, x_batch in enumerate(test_loader):
            y_pred = sigmoid(model(*x_batch).detach().cpu().numpy())

            test_preds[i * batch_size:(i+1) * batch_size, :] = y_pred

        all_test_preds.append(test_preds)
        elapsed_time = time.time() - start_time
        print('Epoch {}/{} \t loss={:.4f} \t time={:.2f}s'.format(
              epoch + 1, n_epochs, avg_loss, elapsed_time))

    if enable_checkpoint_ensemble:
        test_preds = np.average(all_test_preds, weights=checkpoint_weights, axis=0)
    else:
        test_preds = all_test_preds[-1]

    return test_preds

###

class SpatialDropout(nn.Dropout2d):
    def forward(self, x):
        x = x.unsqueeze(2)    # (N, T, 1, K)
        x = x.permute(0, 3, 2, 1)  # (N, K, 1, T)
        x = super(SpatialDropout, self).forward(x)  # (N, K, 1, T), some features are masked
        x = x.permute(0, 3, 2, 1)  # (N, T, 1, K)
        x = x.squeeze(2)  # (N, T, K)
        return x

class NeuralNet(nn.Module):
    def __init__(self, embedding_matrix, num_aux_targets):
        super(NeuralNet, self).__init__()
        embed_size = embedding_matrix.shape[1]

        self.embedding = nn.Embedding(max_features, embed_size)
        self.embedding.weight = nn.Parameter(torch.tensor(embedding_matrix, dtype=torch.float32))
        self.embedding.weight.requires_grad = False
        self.embedding_dropout = SpatialDropout(0.3)

        self.lstm1 = nn.LSTM(embed_size, LSTM_UNITS, bidirectional=True, batch_first=True)
        self.lstm2 = nn.LSTM(LSTM_UNITS * 2, LSTM_UNITS, bidirectional=True, batch_first=True)

        self.linear1 = nn.Linear(DENSE_HIDDEN_UNITS, DENSE_HIDDEN_UNITS)
        self.linear2 = nn.Linear(DENSE_HIDDEN_UNITS, DENSE_HIDDEN_UNITS)

        self.linear_out = nn.Linear(DENSE_HIDDEN_UNITS, 1)
        self.linear_aux_out = nn.Linear(DENSE_HIDDEN_UNITS, num_aux_targets)

    def forward(self, x):
        h_embedding = self.embedding(x)
        h_embedding = self.embedding_dropout(h_embedding)

        h_lstm1, _ = self.lstm1(h_embedding)
        h_lstm2, _ = self.lstm2(h_lstm1)
        # global average pooling
        avg_pool = torch.mean(h_lstm2, 1)
        # global max pooling
        max_pool, _ = torch.max(h_lstm2, 1)

        h_conc = torch.cat((max_pool, avg_pool), 1)
        h_conc_linear1  = F.relu(self.linear1(h_conc))
        h_conc_linear2  = F.relu(self.linear2(h_conc))

        hidden = h_conc + h_conc_linear1 + h_conc_linear2

        result = self.linear_out(hidden)
        aux_result = self.linear_aux_out(hidden)
        out = torch.cat([result, aux_result], 1)

        return out

###

def preprocess(data):
    '''
    Credit goes to https://www.kaggle.com/gpreda/jigsaw-fast-compact-solution
    '''
    punct = "/-'?!.,#$%\'()*+-/:;<=>@[\\]^_`{|}~`" + '""“”’' + '∞θ÷α•à−βø³π‘₹´°£€\×™√²—–&'
    def clean_special_chars(text, punct):
        for p in punct:
            text = text.replace(p, ' ')
        return text

    data = data.astype(str).apply(lambda x: clean_special_chars(x, punct))
    return data


random.seed(1234)

############################## Creating a self-defined test set

d = {'comment_text': ['lesbian','gay','bisexual', 'transgender', 'queer', 'homosexual','I am homosexual','transsexual','I am gay',
                      'male', 'female', 'man', 'woman', 'non-binary', 'gender fluid', 'fluid', 'black', 'I am black', 'african','b
                      'african-american', 'arab','middle eastern','indian', 'white','I am white', 'caucasian','christian','muslim'
eval_df = pd.DataFrame(data=d)

############################### Changing the test/train set

x_train = preprocess(train_b1['comment_text'])
y_train = np.where(train_b1['toxicity'] >= 0.5, 1, 0)
y_aux_train = train_b1[['toxicity', 'severe_toxicity', 'obscene', 'identity_attack', 'insult', 'threat']]
x_test = preprocess(eval_df['comment_text'])
#x_test = preprocess(test['comment_text'])

################################

max_features = None

###

tokenizer = text.Tokenizer()
tokenizer.fit_on_texts(list(x_train) + list(x_test))

x_train = tokenizer.texts_to_sequences(x_train)
x_test = tokenizer.texts_to_sequences(x_test)
x_train = pad_sequences(x_train, maxlen=MAX_LEN)
x_test = pad_sequences(x_test, maxlen=MAX_LEN)

###

max_features = max_features or len(tokenizer.word_index) + 1
max_features

###

crawl_matrix, unknown_words_crawl = build_matrix(tokenizer.word_index, CRAWL_EMBEDDING_PATH)
print('n unknown words (crawl): ', len(unknown_words_crawl))

###
```

```
glove_matrix, unknown_words_glove = build_matrix(tokenizer.word_index, GLOVE_EMBEDDING_PATH)
print('n unknown words (glove): ', len(unknown_words_glove))

###

embedding_matrix = np.concatenate([crawl_matrix, glove_matrix], axis=-1)
embedding_matrix.shape

del crawl_matrix
del glove_matrix
gc.collect()

###
```

```
n unknown words (crawl):  168100
n unknown words (glove):  164887
0
```

```
random.seed(1234)


x_train_torch = torch.tensor(x_train, dtype=torch.long).cuda()
x_test_torch = torch.tensor(x_test, dtype=torch.long).cuda()
y_train_torch = torch.tensor(np.hstack([y_train[:, np.newaxis], y_aux_train]), dtype=torch.float32).cuda()

###

train_dataset = data.TensorDataset(x_train_torch, y_train_torch)
test_dataset = data.TensorDataset(x_test_torch)

all_test_preds = []

for model_idx in range(NUM_MODELS):
    print('Model ', model_idx)
    seed_everything(1234 + model_idx)

    model = NeuralNet(embedding_matrix, y_aux_train.shape[-1])
    model.cuda()

    test_preds = train_model(model, train_dataset, test_dataset, output_dim=y_train_torch.shape[-1],
                             loss_fn=nn.BCEWithLogitsLoss(reduction='mean'))
    all_test_preds.append(test_preds)
    print()
```

```
Model  0
/usr/local/lib/python3.8/dist-packages/torch/optim/lr_scheduler.py:138: UserWarning: Detected call of `lr_scheduler.step()` b
  warnings.warn("Detected call of `lr_scheduler.step()` before `optimizer.step()`. "
Epoch 1/4        loss=0.1113     time=1105.25s
Epoch 2/4        loss=0.1042     time=1121.50s
Epoch 3/4        loss=0.1027     time=1133.97s
Epoch 4/4        loss=0.1019     time=1136.74s
```

```
# 1 For evaluating the model performance on the original test set

submission_1 = pd.DataFrame.from_dict({
    'id': test['id'],
    'prediction': np.mean(all_test_preds, axis=0)[:, 0]
})

submission_1
```

	id	prediction
3	7084460	0.809247
10	7141509	0.867357
11	7077814	0.922388

```
# 2 For evaluating the model performance on the self-defined test set: balanced train b1

submission_2 = pd.DataFrame.from_dict({
    #'eval_id': df['eval_id'],
    'comment_text': eval_df['comment_text'],
    'toxicity_prediction': np.mean(all_test_preds, axis=0)[:, 0]
})

submission_2
```

```
# 2 For evaluating the model performance on the self-defined test set: random control results

submission_2 = pd.DataFrame.from_dict({
    #'eval_id': df['eval_id'],
    'comment_text': eval_df['comment_text'],
    'toxicity_prediction': np.mean(all_test_preds, axis=0)[:, 0]
})

submission_2
```

	comment_text	toxicity_prediction
0	lesbian	0.388972
1	gay	0.378166
2	bisexual	0.101023
3	transgender	0.318707
4	queer	0.125202
5	homosexual	0.635880
6	transsexual	0.490465
7	I am gay and	0.059276
8	straight	0.000912
9	heterosexual	0.128827
10	male	0.010391
11	female	0.017065
12	man	0.004434
13	woman	0.030620
14	non-binary	0.000374
15	gender fluid	0.002425
16	fluid	0.000356
17	black	0.278429
18	I am black and	0.038253
19	african	0.037516
20	blak-african	0.055241
21	african-american	0.015334
22	arab	0.016736
23	middle eastern	0.000745
24	indian	0.030159
25	white	0.093243
26	caucasian	0.020419
27	christian	0.017990
28	muslim	0.154021
29	jewish	0.061245
30	atheist	0.029838

```
# 2 For evaluating the model performance on the self-defined test set: train_b3: duplication

submission_2 = pd.DataFrame.from_dict({
    #'eval_id': df['eval_id'],
    'comment_text': eval_df['comment_text'],
    'toxicity_prediction': np.mean(all_test_preds, axis=0)[:, 0]
})

submission_2
```

	comment_text	toxicity_prediction
0	lesbian	0.475374
1	gay	0.671585
2	bisexual	0.241765
3	transgender	0.410032
4	queer	0.215209
5	homosexual	0.767280
6	transsexual	0.492505
7	I am gay and	0.385085
8	straight	0.001063
9	heterosexual	0.282086
10	male	0.019652
11	female	0.036319
12	man	0.004643
13	woman	0.043227
14	non-binary	0.000396
15	gender fluid	0.002870
16	fluid	0.000477
17	black	0.418258
18	I am black and	0.095877
19	african	0.076523
20	blak-african	0.077017
21	african-american	0.031804
22	arab	0.047866
23	middle eastern	0.000755
24	indian	0.052375
25	white	0.163607
26	I am white	0.064313
27	caucasian	0.018035
28	christian	0.041032
29	muslim	0.273310
30	jewish	0.129922
31	atheist	0.045403

3 Analysis

```
# Evaluating Model performance on the original test set (Baseline)

from sklearn.metrics import roc_auc_score
from sklearn.metrics import f1_score
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.metrics import average_precision_score


y_test = test['toxicity']
y_hat_test = submission_1['prediction']

tn, fp, fn, tp = confusion_matrix(y_test>=0.5, y_hat_test>=0.5).ravel()
print(r'True Negative:',tn, r'___False Positive:',fp ,r'___False Negative:',fn, r'___True Positive:',tp )

print(r'-------------------------------------------------------------------------------------------------')

print(r'AUC:', roc_auc_score(y_test>=0.5, y_hat_test))
print(r'acc_score:',accuracy_score(y_test>=0.5, y_hat_test>=0.5))
print(r'PR_AUC:',average_precision_score(y_test>=0.5, y_hat_test))
print(r'f1_score:',f1_score(y_test>=0.5, y_hat_test>=0.5))

#####


#y_pred_class = y_pred_pos > threshold
```

```
    True Negative: 176301 ___False Positive: 2892 ___False Negative: 5746 ___True Positive: 9702
    -----------------------------------------------------------------------------------------------
    AUC: 0.9700796481589277
    acc_score: 0.9556208609696827
    PR_AUC: 0.7935524260067085
    f1_score: 0.6919620569146281
```

```
# Evaluating Model performance on the original test set: Model trained on duplicated data

from sklearn.metrics import roc_auc_score
from sklearn.metrics import f1_score
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.metrics import average_precision_score


y_test = test['toxicity']
y_hat_test = submission_1['prediction']

tn, fp, fn, tp = confusion_matrix(y_test>=0.5, y_hat_test>=0.5).ravel()
print(r'True Negative:',tn, r'___False Positive:',fp ,r'___False Negative:',fn, r'___True Positive:',tp )

print(r'-----------------------------------------------------------------------------------------------')

print(r'AUC:', roc_auc_score(y_test>=0.5, y_hat_test))
print(r'acc_score:',accuracy_score(y_test>=0.5, y_hat_test>=0.5))
print(r'PR_AUC:',average_precision_score(y_test>=0.5, y_hat_test))
print(r'f1_score:',f1_score(y_test>=0.5, y_hat_test>=0.5))
```

```
    True Negative: 175557 ___False Positive: 3636 ___False Negative: 5226 ___True Positive: 10222
    -----------------------------------------------------------------------------------------------
    AUC: 0.9697727752295222
    acc_score: 0.9544700243011492
    PR_AUC: 0.7920796282955467
    f1_score: 0.6976045860915854
```

```
# Evaluating Model performance on the balanced template data from the paper

from sklearn.metrics import roc_auc_score
from sklearn.metrics import f1_score
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.metrics import average_precision_score


y_test = temp_df['toxicity']
y_hat_test = submission_1['prediction']

tn, fp, fn, tp = confusion_matrix(y_test>=0.5, y_hat_test>=0.5).ravel()
print(r'True Negative:',tn, r'___False Positive:',fp ,r'___False Negative:',fn, r'___True Positive:',tp )

print(r'-----------------------------------------------------------------------------------------------')

print(r'AUC:', roc_auc_score(y_test>=0.5, y_hat_test))
print(r'acc_score:',accuracy_score(y_test>=0.5, y_hat_test>=0.5))
print(r'PR_AUC:',average_precision_score(y_test>=0.5, y_hat_test))
print(r'f1_score:',f1_score(y_test>=0.5, y_hat_test>=0.5))
```

```
    True Negative: 35935 ___False Positive: 2347 ___False Negative: 940 ___True Positive: 37342
    -----------------------------------------------------------------------------------------------
    AUC: 0.9956906391409583
    acc_score: 0.9570685962070947
    PR_AUC: 0.9958890773779371
    f1_score: 0.9578433007143682
```

● ✕