from abc import ABC

import re

from toxicity\_ml\_pipeline.settings.hcomp\_settings import TOXIC\_35

import numpy as np

TOXIC\_35\_set = set(TOXIC\_35)

url\_group = r"(\bhttps?:\/\/\S+)"

mention\_group = r"(\B@\S+)"

urls\_mentions\_re = re.compile(url\_group + r"|" + mention\_group, re.IGNORECASE)

url\_re = re.compile(url\_group, re.IGNORECASE)

mention\_re = re.compile(mention\_group, re.IGNORECASE)

newline\_re = re.compile(r"\n+", re.IGNORECASE)

and\_re = re.compile(r"&\s?amp\s?;", re.IGNORECASE)

class DataframeCleaner(ABC):

def \_\_init\_\_(self):

pass

def \_clean(self, df):

return df

def \_systematic\_preprocessing(self, df):

df.reset\_index(inplace=True, drop=True)

if "media\_url" in df.columns:

print(".... removing tweets with media")

df.drop(df[~df.media\_url.isna()].index, inplace=True, axis=0)

else:

print("WARNING you are not removing tweets with media to train a BERT model.")

print(".... deleting duplicates")

df.drop\_duplicates("text", inplace=True, keep="last")

print(f"Got {df.shape[0]} after cleaning")

return df.reset\_index(inplace=False, drop=True)

def \_postprocess(self, df, \*args, \*\*kwargs):

return df

def \_\_call\_\_(self, df, \*args, \*\*kwargs):

print(f"Got {df.shape[0]} before cleaning")

df["raw\_text"] = df.text

df = self.\_clean(df)

df = self.\_systematic\_preprocessing(df)

return self.\_postprocess(df, \*args, \*\*kwargs)

def mapping\_func(el):

if el.aggregated\_content in TOXIC\_35\_set:

return 2

if el.label == 1:

return 1

return 0

class DefaultENNoPreprocessor(DataframeCleaner):

def \_postprocess(self, df, \*args, \*\*kwargs):

if "toxic\_count" in df.columns and "non\_toxic\_count" in df.columns:

df["vote"] = df.toxic\_count / (df.toxic\_count + df.non\_toxic\_count)

df["agreement\_rate"] = np.max((df.vote, 1 - df.vote), axis=0)

if "label\_column" in kwargs and kwargs["label\_column"] != "label":

if kwargs["label\_column"] == "aggregated\_content":

print("Replacing v3 label by v3.5 label.")

if "num\_classes" in kwargs and kwargs["num\_classes"] < 3:

df["label"] = np.where(df.aggregated\_content.isin(TOXIC\_35\_set), 1, 0)

elif "num\_classes" in kwargs and kwargs["num\_classes"] == 3:

print("Making it a 3-class pb")

df["label"] = df.apply(mapping\_func, axis=1)

else:

raise NotImplementedError

elif kwargs['label\_column'] in df.columns:

df['label'] = df[kwargs['label\_column']]

if kwargs['class\_weight'] is not None:

df["class\_weight"] = np.where(df['label'] == 1, 1-kwargs['class\_weight'],

kwargs['class\_weight'])

else:

raise NotImplementedError

if "filter\_low\_agreements" in kwargs and kwargs["filter\_low\_agreements"] == True:

df.drop(df[(df.agreement\_rate <= 0.6)].index, axis=0, inplace=True)

raise NotImplementedError

return df

class DefaultENPreprocessor(DefaultENNoPreprocessor):

def \_clean(self, adhoc\_df):

print(

".... removing \\n and replacing @mentions and URLs by placeholders. "

"Emoji filtering is not done."

)

adhoc\_df["text"] = [url\_re.sub("URL", tweet) for tweet in adhoc\_df.raw\_text.values]

adhoc\_df["text"] = [mention\_re.sub("MENTION", tweet) for tweet in adhoc\_df.text.values]

adhoc\_df["text"] = [

newline\_re.sub(" ", tweet).lstrip(" ").rstrip(" ") for tweet in adhoc\_df.text.values

]

adhoc\_df["text"] = [and\_re.sub("&", tweet) for tweet in adhoc\_df.text.values]

return adhoc\_df

class Defaulti18nPreprocessor(DataframeCleaner):

def \_clean(self, adhoc\_df):

print(".... removing @mentions, \\n and URLs. Emoji filtering is not done.")

adhoc\_df["text"] = [urls\_mentions\_re.sub("", tweet) for tweet in adhoc\_df.raw\_text.values]

adhoc\_df["text"] = [

newline\_re.sub(" ", tweet).lstrip(" ").rstrip(" ") for tweet in adhoc\_df.text.values

]

return adhoc\_df