Project-2

Introduction

This project uses a dataset of labeled tweets to train a logistic regression model in PyTorch for sentiment analysis. The model takes 12 different features, including sentiment scores and word count, and outputs a probability of the tweet being positive. The model is trained on a training set and evaluated using accuracy and F1-score on a test set.

Discussion

```
import numpy as np
import pandas as pd
import warnings
import math
import torch
import re
import torch.nn as nn
import torch.optim as optim
import torch.tensor as tt
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
import matplotlib.pyplot as plt
warnings.filterwarnings("ignore")
```

Importing the required libraries to run the code and analyze it.

```
enc_type = 'utf-8'
```

Setting the encoding type to the variable. Defining this as this is used by all the tsv files while loading them on to the system.

Project-2

```
def data_clean_tweets(txt_file):

    with open(txt_file,'r',encoding=enc_type) as f :
        data = f.readlines()

    data = [dt.strip() for dt in data]
    for text1 in data :
        # Remove unnecessary punctuations,spaces brackets
        text1 = re.sub(r'\[\d+\]', '', text1)
        # Remove underscores and hyphens from the sides of words
        text1 = re.sub(r'[-_]', '', text1)
        # Remove numbers used for points
        text1 = re.sub(r'\d+\.', '', text1)
        # remove "
        text1 = re.sub(r'"', '', text1)
        # remove '
        text1 = re.sub(r"'", '', text1)
        # remove special characters
        text1 = re.sub(r'[+-.,!@#$%^&<>?/\{}()*_=:;|]', '', text1)
        # remove multiple spaces
        text1 = re.sub(r'\n\s*\n', '\n', text1)
        # remove numbers
        text1 = re.sub(r'\s\d+\s', ' ', text1)
        # remove new line, tabs
        text1 = re.sub(r'\n|\t', ' ', text1)
        # Remove bullets
        text1 = re.sub(r'^[\s\u2022\u2023\u25E6\u2043]*', '', text1, flags=re.MULTILINE)
        # remove multiple spaces
        text1 = re.sub(r' +', ' ', text1)
        # convert the entire text lo lower case
        text1 = text1.lower()
    return data
```

This function or method that is defined is used to clean the data using the regular expressions. This function clears all the data like extra spaces, emojis, multiple spaces, square brackets, special symbols and punctuation marks. This also converts the data to lowercase.

```
def read_label_data(label_data):
    with open(label_data,'r',encoding = enc_type) as f :
        data = f.readlines()
    data = [dt.strip() for dt in data]
    return data
```

This method is used to read the data from the file using the r mode and encoding format as utf-8.

```
train_tweets = data_clean_tweets('sentiment/train_text.txt')
train_labels = read_label_data('sentiment/train_labels.txt')
test_tweets = data_clean_tweets('sentiment/test_text.txt')
test_labels = read_label_data('sentiment/test_labels.txt')
val_tweets = data_clean_tweets('sentiment/val_text.txt')
val_labels = read_label_data('sentiment/val_labels.txt')
```

Project-2

Reading the text files and label files.

```
train_tweets
np The Way You Make Me Feel - 2012 Remaster by Michael Jackson from the album Bad 25th Anniversary.',
I\'m going to see Paper Towns. Saturday, 22 August 2015 at 17:40 in Leigh #Cineworld"',
user Yepo I came in Milan it was my 1st concert and it was so good you guys did an amazing job! You are f
FECT!',
days #dool Tuesday Hope has to pick up Ciara. Rafe wants a real case not just publicity like Justin'.
```

Train tweets file sample data.

```
1 train_labels
```

```
['2',
 '1',
 '1',
 '1',
 '2',
 '2',
 '2',
 '0',
 '2',
 '1',
```

Train labels sample data

```
# define a function to read the lexicon files
def read_lexicons_files(lex_files):
    with open(lex_files, "r", encoding = enc_type) as ff:
        data = ff.readlines()
#     basic cleaning of data
    data = [dt.strip().split('\t') for dt in data]
    lexicon_dict = {}
    for v in data:
        if len(v) == 3:
            key = v[0]
            value = {'-ve': float(v[1]), '+ve': float(v[2])}
            lexicon_dict[key] = value

    return lexicon_dict
```

Here in this we have defined the function to read the lexicon files and converting it to a dictionary and returning it to the calling function.

Project-2

```
all_lexicon_files = ['3DS.tsv', '4chan.tsv', '2007scape.tsv', 'ACTrade.tsv',
                'amiugly.tsv', 'BabyBumps.tsv', 'baseball.tsv', 'canada.tsv',
                'CasualConversation.tsv', 'DarknetMarkets.tsv', 'darksouls.tsv', 'elderscrollsonline.tsv',
                'Eve.tsv', 'Fallout.tsv', 'fantasyfootball.tsv', 'GameDeals.tsv', 'gamegrumps.tsv', 'halo.tsv',
                'Homebrewing.tsv', 'IAmA.tsv', 'india.tsv', 'jailbreak.tsv', 'Jokes.tsv', 'KerbalSpaceProgram.tsv',
                'Keto.tsv', 'leagueoflegends.tsv', 'Libertarian.tsv', 'magicTCG.tsv', 'MakeupAddiction.tsv',
                'Naruto.tsv', 'nba.tsv', 'oculus.tsv', 'OkCupid.tsv', 'Parenting.tsv', 'pathofexile.tsv',
                'raisedbynarcissists.tsv', 'Random_Acts_Of_Amazon.tsv', 'science.tsv', 'Seattle.tsv',
                'TalesFromRetail.tsv', 'talesfromtechsupport.tsv', 'ultrahardcore.tsv', 'videos.tsv',
                'Warthunder.tsv', 'whowouldwin.tsv', 'xboxone.tsv', 'yugioh.tsv']
```

These are all the lexicon files that are used for the project.

```
negative_scores = [adjectives[word]['-ve'] for word in adjectives]
positive_scores = [adjectives[word]['+ve'] for word in adjectives]
```

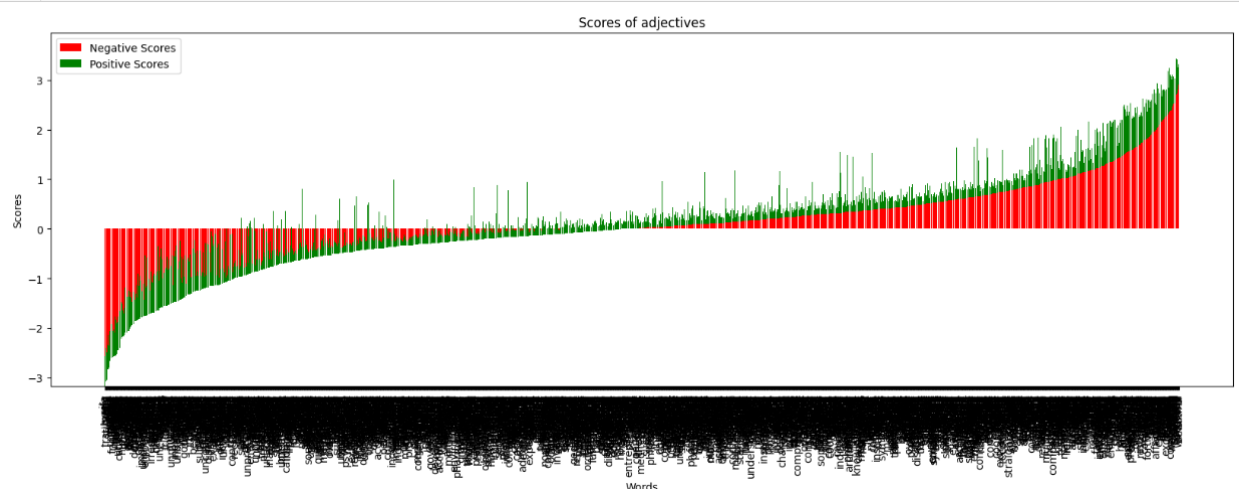
Assigning all the positive and negative values to the lists from the adjectives lexicon file which is named as 2000.tsv.

```
x_axis = list(adjectives.keys())
```

Getting the names of all the words in the 2000.tsv file.

```
# plotting a graph for words in adjectives file
plt.figure(figsize=(20, 6))
plt.bar(x_axis, negative_scores, color='red', label='Negative Scores')
plt.bar(x_axis, positive_scores, color='green', bottom=negative_scores, label='Positive Scores')
plt.xticks(rotation=90)
plt.title('Scores of adjectives')
plt.xlabel('Words')
plt.ylabel('Scores')
plt.legend()
plt.show()
```

Plotting a bar graph to show the positive and -ve scores for the given word.



The wording at the bottom is not clear due to the screen size that is present.

Project-2

```
# Adding all the values to the lexicon_values variable which is a list
lexicon_values = []
xz = {}
for i in all_lexicon_files:
    z = read_lexicons_files('subreddits/'+i)
    xz[i] = len(z)
    lexicon_values.append(z)
```

Adding all the lexicon values to the list.

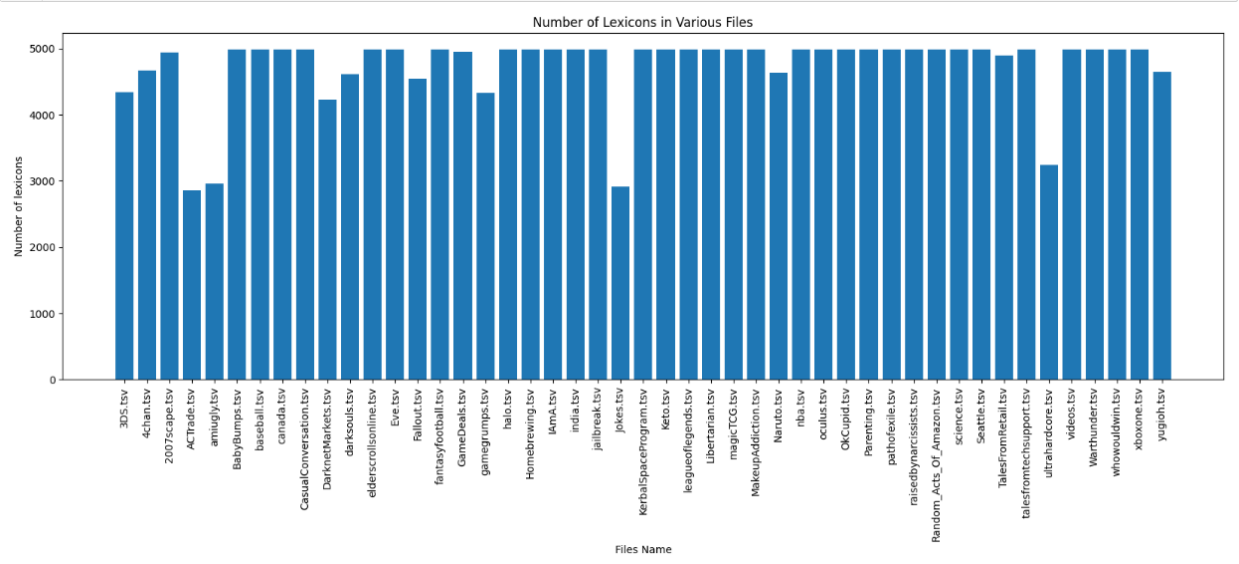
```
names = list(xz.keys())
values = list(xz.values())
```

This stores the list of file names and contents length of all the files.

```
plt.figure(figsize=(20, 6))
plt.bar(names, values)
plt.xticks(rotation=90)
# for i, v in enumerate(values):
#     plt.annotate(str(v), xy=(i, v), ha='center', va='bottom',rotation=90)
plt.xlabel('Files Name')
plt.ylabel('Number of lexicons')
plt.title('Number of Lexicons in Various Files')

# Display the chart
plt.show()
```

This prints the below graph



Project-2

```
# adding the lexicons of adjectives and lexicons from other files
combined = [adjectives, frequency] + lexicon_values
```

This is all the lexicons combined from all the files.

```
def get_feature(tweets, combined):
    # divide into list of words
    wordings = tweets.split()
    # Count words in the tweet
    total = len(wordings)
    # Finding the longest word
    longest = max(wordings, key=len)

    # set 12 features to the list
    feature_set = [0] * 12

    for i, lex_dict in enumerate(combined[:9]):
        score = 0
        for word in wordings:
            sentiment_dict = lex_dict.get(word, {'-ve': 0, '+ve': 0})
            score += sentiment_dict['-ve'] + sentiment_dict['+ve']
        feature_set[i] = score


    # log of the word count for the tweet
    if total > 0:
        feature_set[9] = math.log(total)
    else:
        feature_set[9] = 0

```

Project-2

```
    # log of length of longest word
    if longest:
        feature_set[10] = math.log(len(longest))
    else:
        feature_set[10] = 0


    # Count of words that have 5 characters or more
    long_word_count = 0
    for word in wordings:
        if len(word) >= 5:
            long_word_count += 1

    # log of count of long words
    if long_word_count > 0:
        feature_set[11] = math.log(long_word_count)
    else:
        feature_set[11] = 0


    return feature_set
```

The above code will store the features on feature_set variable. The first nine are taken from the lexicon files and the other three features manually added by doing some actions on dataset.

```
1 train_features = [get_feature(tweet, combined) for tweet in train_tweets]
2 validation_features = [get_feature(tweet, combined) for tweet in val_tweets]
3 testing_features = [get_feature(tweet, combined) for tweet in test_tweets]
```

Loading the data on the variables.

Project-2

```
# making as a input using PyTorch for training set
dtype = torch.float32
label_dtype = torch.float32
X_train = tt(train_features, dtype=dtype)
y_train = tt(list(map(int, train_labels)), dtype=label_dtype).unsqueeze(1)

# making as a input using PyTorch for validation set
X_val = tt(validation_features, dtype=dtype)
y_val = tt(list(map(int, val_labels)), dtype=label_dtype).unsqueeze(1)

# making as a input using PyTorch for test set
X_test = tt(testing_features, dtype=dtype)
y_test = tt(list(map(int, test_labels)), dtype=label_dtype).unsqueeze(1)
```

Preparing the framework using pytorch to create the data to provide data as input.

Project-2

```
class LogisticRegressionDef(nn.Module):
    def __init__(self, input_size):
        super(LogisticRegressionDef, self).__init__()
        self.linear = nn.Linear(input_size, 1)
#         using sigmoid function from torch library
        self.sigmoid = nn.Sigmoid()

    def forward(self, x):
        nex = self.linear(x)
        nex = self.sigmoid(nex)
        return nex

    def train(self, X_train, y_train, lr=0.01, epochs=100):
        optimizer = optim.SGD(self.parameters(), lr=lr)
        loss_fn = nn.BCELoss()

        for i in range(epochs):
            y_pred = self(X_train)
            loss = loss_fn(y_pred, y_train)
            loss.backward()
            optimizer.step()
            optimizer.zero_grad()

    def predict(self, X):
        with torch.no_grad():
            y_predict = self(X)
            y_predict = (y_predict >= 0.5).float()
        return y_predict

    def evaluate(self, X, y):
        y_predict = self.predict(X)
        find_accuracy = accuracy_score(y, y_predict)
        find_f1 = f1_score(y, y_predict, average='weighted')
        return find_accuracy, find_f1
```

The above class is defined to provide a base for performing the logistic regression for the tweets.

Project-2

```
# creating the object for the class
lr1 = LogisticRegressionDef(12)
# calling the train method
lr1.train(X_train, y_train,lr=0.01, epochs=200)
# getting the accuracy and f1-score for the test sets
accuracy, f1_score = lr1.evaluate(X_test, y_test)
print("Accuracy : ",accuracy)
print("F1 Score : ",f1_score)
```

This provides a reference for the class that is defined above. The object that is generated is calling the logistic regression functions and performing the training and test to find the accuracy and f1_score.

Output Analysis:

```
Accuracy :  0.48363725170954086
F1 Score :  0.31582110771558486
```

The Accuracy of the model is 48.3% where as the f1 score is 31.5%

References

https://numpy.org/

https://pandas.pydata.org/

https://matplotlib.org/

https://scikit-learn.org/stable/

https://pytorch.org/get-started/locally/

https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

https://stackoverflow.com/questions/34093264/python-logistic-regression-beginner

https://pytorch.org/docs/stable/optim.html

https://stackoverflow.com/questions/22540449/how-can-i-rotate-a-matplotlib-plot-through-90-degrees

https://stackoverflow.com/questions/53975717/pytorch-connection-between-loss-backward-and-optimizer-step

https://pytorch.org/docs/stable/generated/torch.nn.BCELoss.html

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html

https://stackoverflow.com/questions/42471082/how-to-find-out-the-accuracy

Harshavardhan Reddy Mallannagari
11603389

Project-2