**Introduction**

This project uses a dataset of labeled tweets to train a Neural Network 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.

**Implementation:**

**Text

Description automatically generated**

Importing the required libraries for running the code and analyzing it.

Text

Description automatically generated

This function definition is used to read the lexicon files using the encoding utf-8, this code reads the data ,cleans the data and assigns positive and negative values to the key.

After parsing all the values the function returns the dictionary of lexicons.

Text

Description automatically generated

In the above code we are taking 2 input files, one is the text file and the other is the label file which is related to the text file.

Text

Description automatically generated

In the above function, the data is cleaned by using regular expressions, by using stop-words from nltk module. We are also using porter stemmer which applies stemming to each word.

Text

Description automatically generated

The above function creates the features for passing. This is being used from project 2. This generates 12 features and returns the features list.

Text

Description automatically generated

Sentiment data set is a class that is extended from Data set of torch library. Here in this we convert the features to the tensors.

The length returns the length of feat variable which is the features that is converted to tensors. Get Item returns the feat and label.

Text

Description automatically generated

Setting the encoding type as utf-8 and assigning all the names of lexicon files to the list to read.

Graphical user interface, text

Description automatically generated

This reads the adjective file.

Text

Description automatically generated

Reading all the lexicon files using a loop and storing in lexicon\_values variable.

Text

Description automatically generated

Stores keys and values in the variables name ad values.

These are used in visualization.

Text

Description automatically generated

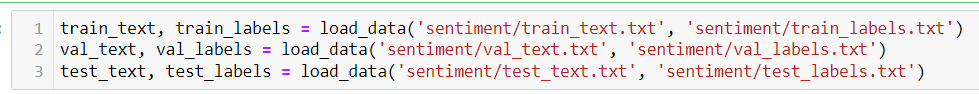
Chart, bar chart

Description automatically generated

This graph shows how many lexicons are there in which file.



Combining lexicon and adjective and frequency values.



Reading the values from the files.

Graphical user interface, text, application

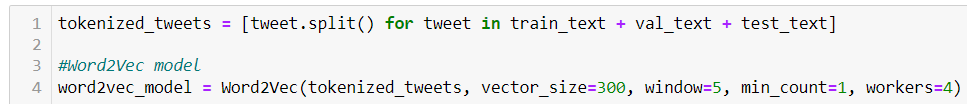
Description automatically generated

Cleaning the data.

Text

Description automatically generated

This is used to get the embedding vector by passing the tweet , model and the size.



Creating tokens by combining all the data.

Then passing all the tokenized to the word 2 vector model.

Text, letter

Description automatically generated

Encoding the labels of the files. The labels are encoded to long variable of torch type.

Text

Description automatically generated

Getting the train validation, testing set of size 300 . After that adding all the values to the x\_all variable as a numpy library.

Graphical user interface, text, application

Description automatically generated

This is extended from the Dataset of the torch module. This checks the input values are of same length or not. It converts the data into torch tensors for running the code.

Text

Description automatically generated

This is extended from the model nn module of torch library. This model uses the linear function in nn module(layers) , relu activation, softmax. The forward function returns the output of the output layer after performing the activation functions on the data.

Text

Description automatically generated

This has variables which holds the data, inputsize. These variables holds the data that is passed to the DataLoader of the torch library and holds those values in the provided variables.

Graphical user interface, text

Description automatically generated

Setting the hidden size of layers as 64, classes division as positive, negative , neutral, batch size as input to neural network as 32 and model as sentimentModel. This is the class object created for the sentiment Model.

Graphical user interface, text, application

Description automatically generated  
using the crossEntropyLoss as the criterion.

Using the adam optimizer for the model.

Text

Description automatically generated

This is the train model that calculates the losses and stores the values for every fold and prints the loss for every epoch. Finally it returns the appened values of losses and accuracies

Text

Description automatically generated

This function sets the model in evaluation This turns off training-specific layers like dropout and batch normalization which should not be active during evaluation.

Runloss is used to keep track of the overall loss. This uses the criterion. This returns the average loss of the batch that is sent.

Text

Description automatically generated

This method predict takes the model and tensor. After all the work this function sends the predicted values from the pytorch model.

Graphical user interface

Description automatically generated with medium confidence

Setting number of epochs for each fold as 100.

Text

Description automatically generated

This is the main cell block which calls the train model and calculates the losses and accuracies of the model for each epoch. In this we hold all the losses and accuracies for graph visualization. This prints the visualizations and confusion matrix , accuracy , f1 score.

Text

Description automatically generated

The above is the sample of the output.

Graphical user interface, text, application

Description automatically generated

The above code shows graph for the epoch and epoch loss for every fold.

Chart, histogram

Description automatically generated

Graphical user interface, text, application

Description automatically generated

The above code shows the epoch number and accuracy of the model.

Chart

Description automatically generated

**Discussion :**

As we are shuffling the data, there are no biases. The average accuracy of the model is 58 % and the average F1-score is 56%.

The above graphs shows the Loss and accuracy of every fold and every epoch.

A confusion matrix is also generated for every fold.

**References:**

<https://pypi.org/project/nets/0.0.3.1/>?

<https://discuss.pytorch.org/t/batch-normalization-and-weight-initialization-in-seq2seq/54934>

<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://www.deeplearningwizard.com/deep_learning/practical_pytorch/pytorch_feedforward_neuralnetwork/>

<https://www.deeplearningwizard.com/deep_learning/boosting_models_pytorch/weight_initialization_activation_functions/>