Multiclass Sentiment Analysis using CNN

Bhargavkumar Patel

Department of Computer Science

bpatel22@lakeheadu.ca

1106925

Lakehead University

***Abstract*--In this experiment, we are analysis rotten tomatoes data to do Multi-class Sentiment Analysis using a simple CNN.The original data has 156060 entries. Data has four at- tributes,i.e., Phrase Id, Sentence Id, Phrase, and Sentiment. In this experiment, we use Bag of Words (BoW) and term frequency- inverse document frequency (tf-IDF) for data vectorization. Vectorized data is then passed to our Convolutional neural network (CNN)models. The keras model takes tf-IDF vectorized data and the second model takes BoW vectorized data.The model is based on Accuracy, Recall, Precision, and figure of merit measures. In this model we get an accuracy of 0.6378 for training data and 0.7725 for testing data.**

Index Terms—Bag of Words, term frequency-inverse document frequency, Convolutional neural net- work, Vectorization, Sentiment Analysis.

I. Introduction

Semantic analysis describes the process of understanding natural language– the way that humans communicate–based on meaning and context. The semantic analysis of natural language content starts by reading all of the words in content to capture the real meaning of any text. It identifies the text elements and assigns them to their logical and grammatical role.It analyses context in the surrounding text and it analyses the text structure to accurately disambiguate the proper meaning of words that have more than one definition. Semantic technology processes the logical structure of sentences to identify the most relevant elements in text and understand the topic discussed. It also understands the relationships between different concepts in the text. Bag of Words is a method to extract features from text documents. These features can be used for training machine learning algorithms. It creates a vocabulary of all the unique words occurring in all the documents in the training set. A Term Frequency is a count of how many times a word occurs in a given document (synonymous with bag of words).The Inverse Document Frequency is the number of times a word occurs in a corpus of documents. tf-idf is used to weight words according to how important they are. Words that are used frequently in many documents will have a lower weighting while infrequent ones will have a higher weighting.The tf-idf value increases proportionally to the number of times a word appears in the document, but is offset by the frequency of the word in the corpus, which helps to adjust for the fact that some words appear more frequently in general. tf-idf is used in a number of NLP techniques such as text mining,search queries and summarization. In this experiment, we are using Keras models to train our model. We are using Accuracy,F1 Score, Precision, and Recall as our model Metrics. The dataset comprises of 156060 entries. After the preprocessing and vectorization stage we get " 156060" rows which we split in training and testing data in 70:30 and random state 2003,we get 109242 rows in training data and 46818 rows in testing data.

DATAPREPROCESSING

Steps to achieve Multiclass Semantic Analysis for Rotten- Tomatoes dataset:

1) Import the Rotten Tomatoes dataset from the url.
2) Split the dataset to 70 percent for training and 30 percent for testing using the sklearn train-test split library.
3) Differentiate the sentiment classes and phrases from the dataset.
4) Remove the unwanted words from the BOW(Bag of words).
5) Vectorize the data to compute TFID vectorized data.6) Evaluate the shape of the train part (X_train,Y_train).
7) Define the functions for evaluating the scores like re- call,precision,accuracy and F1.
8) Frame the CNN model with the convolution lay- ers,Maxpooling layer,flatten and dense and the activation function .
9) Retrieve the model architecture by plotting the summary of the model.
10) Fit the model for evaluating for certain number of epochs and with valid optimizer initializing.
11) Evaluate the Testing scores and plot the graph.

Libraries Used

- Pandas
- Numpy
- Nltk
- Sklearn
- String
- Keras

- Random
- Matplotlib

II. RELATED WORK

Importing libraries

```
import numpy as np
import nltk
import re
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
import matplotlib.pyplot as plt
import random
from sklearn.model_selection import train_test_split
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer, PorterStemmer
from sklearn.feature_extraction.text import TfidfVectorizer,CountVectorizer
from string import punctuation
from keras.models import Model
from keras.models import Sequential
from keras.layers import Dense, Activation, Flatten, Dropout, Conv1D, MaxPooling1D
from keras.layers import Embedding
from keras.utils.vis_utils import plot_model
from keras.layers.merge import concatenate
from keras.callbacks import EarlyStopping
from sklearn.utils import class_weight
import keras
from keras import backend as K
from keras import utils
from keras.optimizers import Adam
```

Load the Dataset:

Rotten Tomatoes is one of the most popular film websites, which combines movie information, critic reviews and users reviews.
Code Snippet to download dataset:

```
dataset = pd.read_csv("https://raw.githubusercontent.com/cacoderquan/Sentiment-Analysis-on-the-Rotten-Tomatoes-movie-review-dataset/master/train.tsv", header=0, delimiter="\t", quoting=3)
```

Sentiment and Sentiment class classifier

The Sentiment are divided in 5 different classes namely:
1. Negative
2. Neutral
3. Somewhat Negative
4. Somewhat Positive
5. Positive

Depending upon the word detection from the BOW(Bag of Words) the sentiment are classified into various classes and plotted accordingly.

Code Snippet:

```
Sentiment_words=[]
for row in dataset['Sentiment']:
    if row ==0:
        Sentiment_words.append('negative')
    elif row == 1:
        Sentiment_words.append('neutral')
    elif row == 2:
        Sentiment_words.append('somewhat negative')
    elif row == 3:
        Sentiment_words.append('somewhat positive')
    elif row == 4:
        Sentiment_words.append('positive')
    else:
        Sentiment_words.append('Failed')
dataset['Sentiment_words'] = Sentiment_words
```

Plotting the count of moods from the Sentiment Class

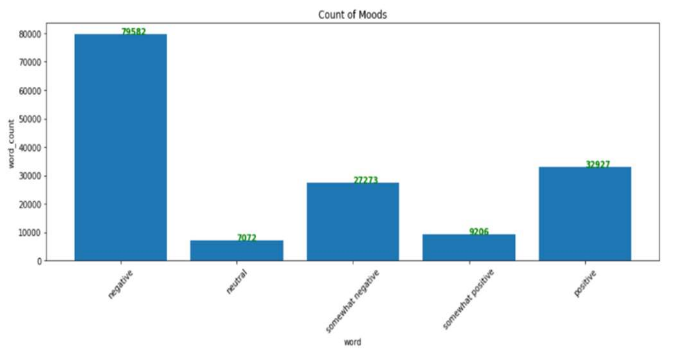


Fig1: Moods detection

Lemmatizing, BOW ,Vectorizing and TF-ID

Lemmatization (or lemmatization) in linguistics is the process of grouping together the inflected forms of a word so they can be analyzed as a single item, identified by the word's lemma, or dictionary form. BOW is a way of extracting features from the text for use in machine learning algorithms. Here we use the tokenized words for each observation and find out the frequency of each token. Vectors convert text that can be used by the machine learning algorithm. CountVectorizer works on Terms Frequency, i.e. counting the occurrences of tokens and building a sparse matrix of documents x tokens. TF-IDF stands for term frequency-inverse document frequency. TF-IDF weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus.

- Term Frequency (TF): is a scoring of the frequency of the word in the current document. Since every document is different in length, it is possible that a term would appear much more times in long documents than shorter ones. The term frequency is often divided by the document length to normalize.

$$TF(t) = \frac{\text{Number of times term } t \text{ appears in a document}}{\text{Total number of terms in the document}}$$

- Inverse Document Frequency (IDF): is a scoring of how rare the word is across documents. IDF is a measure of how rare a term is. Rarer the term, more is the IDF score.

$$IDF(t) = log_e\left(\frac{\text{Total number of documents}}{\text{Number of documents with term } t \text{ in it}}\right)$$

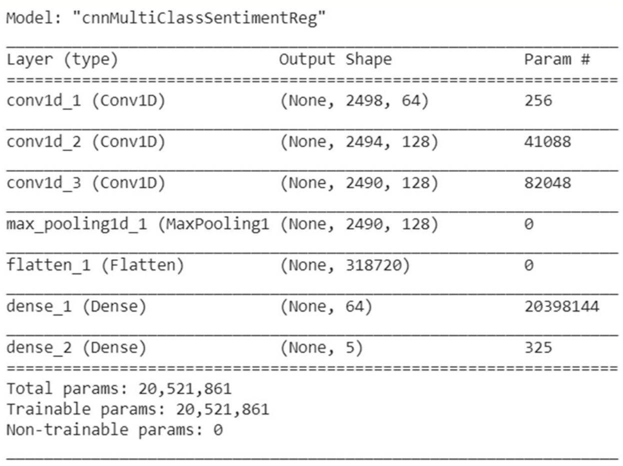
$$TF - IDFscore = TF * IDF$$

Create a CNN model using keras

Steps to follow during the design of the Convolution network:

- Provide input image into convolution layer
- Choose parameters, apply filters with strides, padding if requires. Perform convolution on the dataset and apply ReLU activation to the matrix.
- Perform pooling to reduce dimensionality size
- Add as many convolutional layers until satisfied
- Flatten the output and feed into a fully connected layer (FC Layer)
- Output the class using an activation function (Logistic Regression with cost functions) and classifies the actual class of the sentiment.Activation function used during the output layer are softmax,sigmoid,tanh and many more. Depending upon the linearity of the data the output activation function are used.
- The model architecture can be gained by plotting the model summary that gives the overview of the parameters taking part in each layer by layer.

```
model = Sequential(name='cnnMultiClassSentimentReg')
model.add(Conv1D(filters=64, kernel_size=3,activation='relu',input_shape=(2500,1)))
model.add(Conv1D(128, kernel_size=5, activation='relu'))
model.add(Conv1D(128, kernel_size=5, activation='relu')
model.add(MaxPooling1D(pool_size=1))
model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dense(5, activation='softmax'))
```



```
Model: "cnnMultiClassSentimentReg"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_1 (Conv1D)            (None, 2498, 64)          256
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 2494, 128)         41088
_________________________________________________________________
conv1d_3 (Conv1D)            (None, 2490, 128)         82048
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 2490, 128)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 318720)            0
_________________________________________________________________
dense_1 (Dense)              (None, 64)                20398144
_________________________________________________________________
dense_2 (Dense)              (None, 5)                 325
=================================================================
Total params: 20,521,861
Trainable params: 20,521,861
Non-trainable params: 0
_________________________________________________________________
```

Fig2: Model Summary

Accuracy Matrix

Matrix	Training	Testing
Accuracy	0.7573	0.6217
Precision	0.7753	0.5962
Reacall	0.7325	0.6359
F1 score	0.7540	0.6153

Graph plots for measured parameters

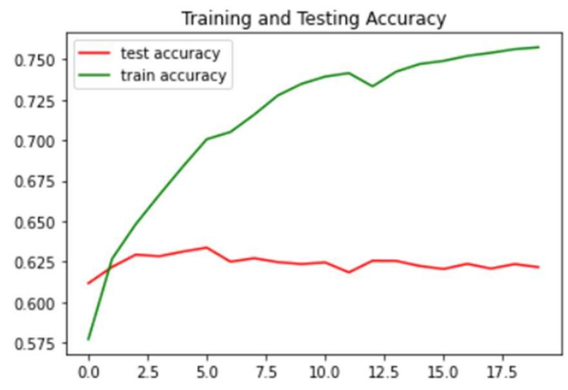


Fig 3: Accuracy Plot

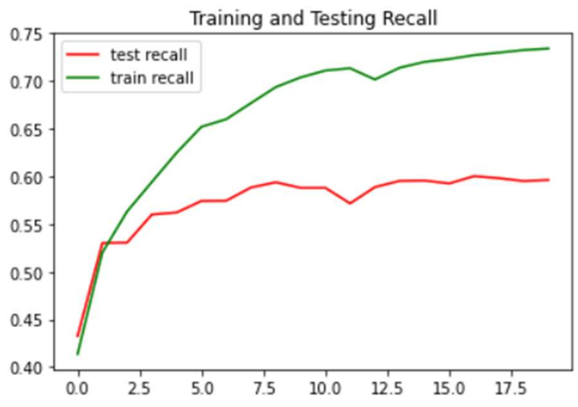


Fig 4: Recall Plot

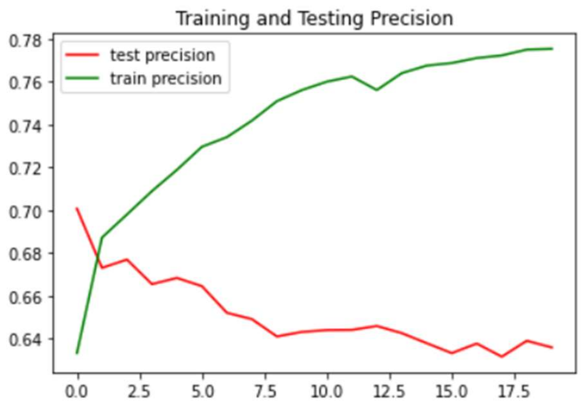


Fig 5: Precision Plot

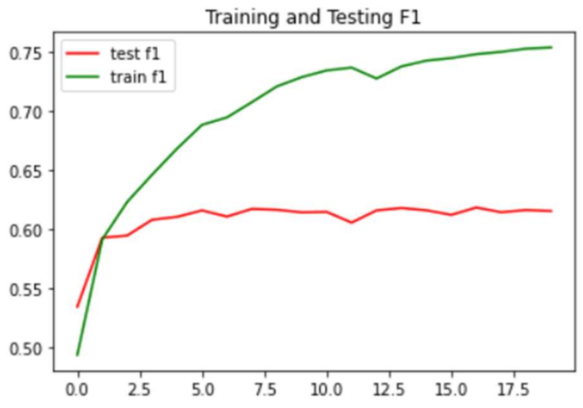


Fig 6: F1 score

III CONCLUSION

In this Assignment I have created a 1D convolution model that trains the model on basis of certain parameters and tests the model with a loss of approximately 0.6258 (62.58 percent) and average F1 score approximately up to 0.6153 (61.35%) on final notes. This assignment demonstrates the idea of Multiclass Sentiment analysis to classify the Rotten Tomato movie review dataset. Apart from this it helped me gain a vast knowledge of how modular programming is executed and its importance when working with large codes, how to create a CNN model for string data and to manipulate the data to retrieve the sentiment analysis and how altering the parameters results in drastic change to accuracy and the f1 score detection. It gave me a great hands on experience by formulating different aspects of code and terminologies to design a user friendly and efficient program. The biggest challenge I faced is to design a CNN model is to deal with the complex large memory dataset and lots of operations on the review sentences like vectorising, lemmatization, BOW , TF-IDF operations. Also I realised that the only way you can turn an inefficient code to efficient code is by performing error solving method. For information on source code and to learn I have created a git repository: https://github.com/Bhargav0312/Multiclass-Sentiment-Analysis

IV REFERENCE

1. https://medium.com/
2. https://keras.io
3. https://www.nltk.org
4. https://machinelearningmastery.com/