Multi-class Sentiment Analysis using Deep Learning

Author : Nagulapati, Venkata Saideep
Student ID: 1105142
Email: vnagulap@lakeheadu.ca
22 March 2020

Abstract— **The objective of the project is to perform sentiment analysis on Rotten Tomatoes movie review dataset (only on train set). The dataset taken is a multi-class sentiment data. The first five rows in of the dataset is printed to give the preview of the dataset. The train data is further divided into train and test splits and preprocessing is done. An opensource library named pytorch is used to construct convolution neural network. There are several hyper parameters which are to be tuned to get the best F1 score by taking care of several scenarios like overfitting, undefitting, vanishing gradient etc. Considering all these requirements and conditions, the non-linear model is developed.**

I. INTRODUCTION

In this section let us elaborate about the libraries used. The concept of Sentiment Analysis and details of Convolution Neural Networks Usually, for building deep learning models, there are three main libraries. They are:

- Keras
- Pytorch
- Tensorflow

Here in our experiment we used pytorch library with the help of pandas library for data computation and matlibplot for plotting the graphs. The key concepts used are:

- **Pytorch library:** Pytorch is an open-source machine learning library based on torch library. This library is mainly used for developing computer vision algorithms and Natural Language Processing algorithms. It is developed by using Facebook AI Research Lab. Pytorch is faster deep learning training than tensorflow. The data parallelism in used over batch dimension which helps to leverage multiple GPUs easily.

- **Natural Language Processing:** Natural Language Processing , usually shortened as NLP, is a field of Artificial Intelligence, which deals with reading, understanding and deriving meaning from human languages. NLP helps computers communicate with humans in their own language and improves any other language related tasks.

- **Sentiment Analysis:** Sentiment Analysis is a process of computationally identifying and categorizing human opinions expressed in the form of text. Usually, this process is done to determine the writers attitude towards a particular topic or a thing. The outcome of such process is generally positive, negative or neutral.

- **Convolution Neural Network:** Convolution Neural Network is a type of deep learning network which takes the input data, assigns weights and bias to various features in data so that we can differentiate between the data and guess the unseen data. A Convolution network mainly learns through filters.

  - **Filters in Convolution:** A filter is a vector of weights which we multiply/convolve with input. The main function of a filter is to extract important features.

  - **Pooling Layer:** Pooling layer is responsible for reducing spatial size of convolved feature. This is mainly done to reduce the computational complexity required to process the data through dimensionality reduction. There are two types of pooling, Average Pooling and Max Pooling.

  - **Activation Functions:** An activation function is a node which is placed at the end or in between the layers. The main purpose of this is to decide when to fire the neuron. It helps in maintaining the output between a range of values. There are many activation functions. Some of them which are widely used are:
    * Sigmoid
    * ReLu
    * Sine

II. LITERATURE SURVEY

In [1] suggests all the possible approaches that can be used for sentiment analysis. Some of the approaches suggested are:

- Bag of Words
- Random Forests

In [2], sentiment analysis is done on the same dataset. This paper used Multinomial Naive Bayes(MNB), Support Vector Machine(SVM) and Deep Learning models in the classification process.
The accuracy obtained by applying MNB is around 75 percent
Whereas the accuracy obtained using SVM is around 72 percent
The accuracy obtained using deep learning model is approximately 60 percent

To increase the accuracy scores and as required by the problem statement, we are using convolution neural network in our assignment.

III. Proposed Model

As we have 1-D row data, we use 1 Dimensional convolution neural network. The main difference between 2D and 1D convolution is the dimension of filter. The flow diagram of proposed model is give in Fig 1.

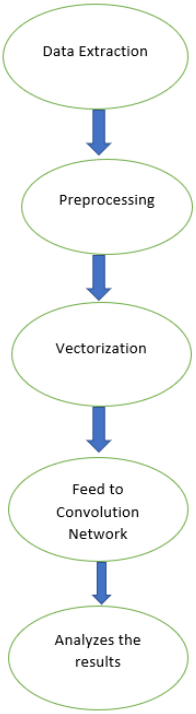

Fig. 1. Flow diagram

- **Dataset :** The dataset contains of 156060 rows, which are nothing but 156060 phrases with 5 columns. It is obtained from [3] The Sentiment column is the class label which is to be predicted using our model. Phrase is the column which is used to predict Sentiment. The values of Sentiment ranges between 1 to 5 where the numbers represent very negative, negative, neutral, positive and very positive respectively.
  .
- **Data Preprocessing :** In this step, preprocessing of the review is done. We have used *WordNetLemmatizer* and *wordtokenize* functions to lemmatizer and tokenize the words is the review. Stopwords and punctutations are also removed from the reviews. Its respective code is shown in Appendix A.
  .
- **Vectorization:** The data we have is in the format of strings. This string data should be converted to numerical form to feed it to a neural network. For this purpose, we have used *TfidfVectorizer* from sklearn libraries. Its respective code is shown in Appendix B

  The hyper-parameters used while vectorizing are
  - max features = 4000
  - n gram range = (1,3)
- **Convolution Model:** A 1D Convolution neural network is built to train the processed data with different kernel sizes in each layer. Its respective code is shown in Appendix C

  The hyper-parameters used while Convolution network are
  - **Filters:** Number of filters ranges from 64 to 256 depending upon the layer.
  - **Kernel Size:** The kernel size used in all the layers is 3
  - **Pool Size:** A pool size of 2 is used in max pooling layers

IV. Challenges Faced

The following are the challenges faced in building the model

- **Max Features :** While vectorizing the data, libraries extract all the features by default. But becuase of this we faced runtime memory overload issue. The issue was because, the resulting array after vectorizing is very large and performing operations on such large data is in efficient. So, we have selected feature size of 4000.

- **Pytorch vs Keras :** Initially, we were using pytorch libraries to build the convolution. As pytorch converts the arrays into multidimensional numpy arrays, the memory was overloading. So, we moved to implementing the model in keras,

V. Experimental Results

The experiment is done with different training parameters. The parameters used are:

- **Batch size :** 32 ,64 ,128
- **Number of epoch :** 10
- **Optimizer Used :** Adam

The metrics used in calculating the goodness of the model are:

- **Categorical Accuracy:** As our dataset contains multiple classes, normal accuracy measure will not show good results. So we have used categorical accuracy as a metric.

- **Precision:** Precision gives the idea about what portion of positively identifications are actually correct. It takes true positives and false positives into account

- **Recall:** Recall gives us the idea about what portion of actual positives are identified correctly. It takes true positives and flase negatives into the account.

Batch Size	Accuracy	Precision	Recall	f1_score
32	0.6480	0.6785	0.5910	0.6310
64	0.6452	0.6690	0.6022	0.6333
128	0.6481	0.6740	0.5992	0.6338

TABLE I
TEST RESULTS WITH VARIED BATCH_SIZE

- **f1 score:** F1 score is one of the key metrics used especially when the data is unbalanced. F1 score is measured by using the precision and recall scores which are already calculated.

The achieved results are tabulated in TABLE 1. Based on the experiment, the best accuracy scores are found when the batch size is 128. The respective comparision of the results based on batch size is shown in Fig 2.

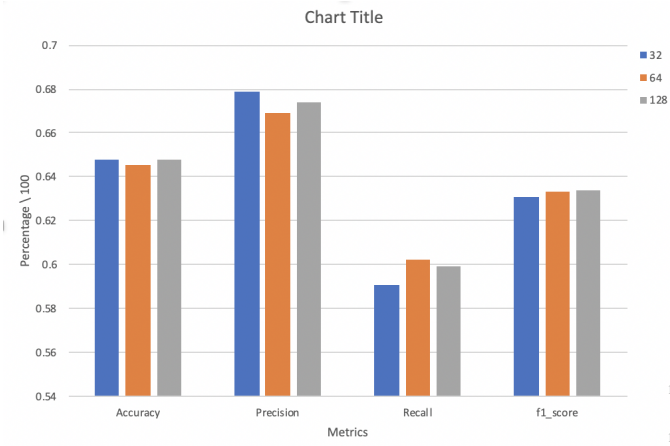

Fig. 2. Results comparision with respect to batch size

VI. CONCLUSION

From the above model it is observed that by increasing kernel size, batch size and number of convolution layers, there is a fair possibility for increasing accuracy, recall, precision and f1 score.

Link to github: https://github.com/saideep3/Test/blob/master/NLP_Assignment_2.ipynb

VII. APPENDIX

A. Preprocessing

```
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
import re

pd.set_option('max_colwidth',400)

wordnet = WordNetLemmatizer()
stopwords_en = stopwords.words("english")
punctuations = "?:!.,;-()"

raw_reviews = data.Phrase.values
cleaned_reviews = []

for i in range(len(raw_reviews)):
  review = str(raw_reviews[i])
  review=re.sub('[^a-zA-Z]',' ',review)
  review=[wordnet.lemmatize(w) for w in
    word_tokenize(str(review).lower())]
  review=' '.join(review)
  cleaned_reviews.append(review)

data['cleaned_reviews'] = cleaned_reviews

data.head()
```

Listing 1. Preprocessing

B. Vectorizer

```

from sklearn.feature_extraction.text import
    TfidfVectorizer
from sklearn.feature_extraction.text import
    CountVectorizer
vectorizer = TfidfVectorizer(max_features=4000,
    stop_words = None, ngram_range=(1,3))
X = vectorizer.fit_transform(X)
```

Listing 2. Vectorizer

C. Convolution model

```

def model_build(data_x):
  #Adding 1st convolution layer
  #with 32 filters and kernel_size = 3
  model.add(Conv1D(filters = 64, kernel_size=3,
    activation='relu', input_shape=(data_x.shape
    [1],1)))
  #Adding MaxPool layer with pool_size as 2
  model.add(MaxPooling1D(pool_size =2))
  #Adding 2nd convolution layer
  #with 64 filters and kernel_size = 5
  model.add(Conv1D(filters = 128, kernel_size=5,
    activation='relu'))
  #Adding MaxPool layer with pool_size as 2
  model.add(MaxPooling1D(pool_size =2))
  #Adding 3rd convolution layer
  #with 128 filters and kernel_size = 7
  model.add(Conv1D(filters = 256, kernel_size=7,
    activation='relu'))
  #Adding MaxPool layer with pool_size as 2
  model.add(MaxPooling1D(pool_size =2))
  model.add(Flatten())
  #Adding Fully connected layer
  model.add(Dense(128, activation='relu'))
  #Adding Dropout layer
  model.add(Dropout(0.5))
  #Output Layer with 5 outputs
  model.add(Dense(5, activation='softmax'))
```

Listing 3. Convolution Network

REFERENCES

[1] Sentiment Analysis, Nakul Dawra, Nikola Kovachki, Alex Lew, Walker Mills, Xiang Ni April 20, 2015
[2] Wu, J.Y. (2012). Predicting Sentiment from Rotten Tomatoes Movie Reviews.
[3] https://github.com/cacoderquan/Sentiment-Analysis-on-the-Rotten-Tomatoes-movie-review-dataset/blob/master/train.tsv