Multi-class Sentiment Analysis using Deep learning

NLP Assignment 2 Report

March 21, 2020

Name: Pavankumar Patel Student Id: 1107159

***Abstract*—Today in the world racing towards the artificial intelligence and deep networks it is necessary to progress in the field of language processing also. Language helps humans communicate with each other. In the same way we need to find a more efficient way to make machine learn our language so that we can order the machine to do the task verbally. Since it is a long way to go here i have proposed a Tf-idf vectorizer based CNN model to classify the reviews of multi class movie review dataset. Sentiments are important part of feedback and review survey.To know the real feeling behind the text a sentiment analyser is the one we need to create. The proposed model gave better performance on the tri-gram model compared to bi-gram model. The accuracy of the models in each experiments was around 60-61 percent.**

I. Introduction

Deep learning approach of solving real world problems is becoming efficient way. Deep learning models are giving noteworthy results in various fields like Natural Language Processing, Image Processing. object detection, Disease diagnostics etc. This models have different architectures that can be modified based upon the developer requirements. Convolution layers are the feature extracting layers of the model which are always followed by the dense layer or we can say the fully connected layer. Sentiments are the most important part of a human. sentiments differs humans from machines in today's date. Its hard to make machine understand of show sentiments. As deep learning methods leans towards learning things by its own, we would try to make the model predict the sentiment based on the reviews. So, the convolution layers in the deep learning model architecture will extract the features from the reviews that could be helpful to classify the sentiments of user.

II. Literature Review

A. Sentiment Analysis

Study of sentiments includes the processing of thoughts, views and subjective text. Sentiment analysis offers the public understanding details by evaluating the numerous tweets and comments.User reviews are used for evaluations of a particular individual, such as person, product or venue, and can be found on websites such as Amazon and Yelp. [1] Such views may be marked as negative, optimistic or neutral.The need to evaluate and organise secret information from the social media in the form of unstructured data increases the need for sentiment analysis.

A number of featured values like tri-grammes and bi-gram by polarities and variations are found in sentiments. Thus sentiments are evaluated by various help vector machines in both negative and positive aspects, using training algorithms.The conditional dependencies between several edges and acyclic graph nodes managed by the Bayesian networks are used for the extraction of data at the context level. The quality of learning and data can be accomplished on the social media site by refining words and sentences.Data tokenisation is used at word root level to generate negative and positive data aspects. Techniques are used to the errors in the analysis of emotions so that social media data are more reliable. To work on review and comments given by the user to predict the sentiments we need lots of preprocessing of data. As, the real meaning of the the words come from their root word, it is a good step to lemmatize each words after tokenization. Stop words removal is also a one of the steps that can be done if the stop words doesnt impact your model.

B. Convolution Neural Network

CNN is a powerful deep model in the field of understanding image content. And it's becoming increasingly popular for many classification problems, with the success of ImageNet classification with ConvNets. Because of the great performance of CNN, we have reason to believe that CNN can be applied to the field of sentiment analysis [1]. CNN has very few parameters and are easy to train. The paper that I reffered has shown how they used the word2vec framework for sentiment analysis along with CNN. They first of all vectorized the word to get the the full sentences. They used cafe for modularity, speed, expression in mind.

III. Proposed Model

In this project, I have proposed a CNN model along with TF-IDF vectorizing framework. First, we vectorize the review phrases and then apply CNN model architecture.

A. Dataset

In this project we are using Rotten Tomato's movie review dataset. This movie review dataset contains 4 features and 156060 instances. This is a multi class dataset with 5 classes of sentiments i.e ranging from 0 to 4, where 0 means very negative reviews while 4 depicts very good review. Dataset contains the split parts of the whole review with the same Phrase ID. The first 10 instances of the dataset are observed

in the figure below.

	PhraseId	SentenceId	Phrase	Sentiment
0	1	1	A series of escapades demonstrating the adage ...	1
1	2	1	A series of escapades demonstrating the adage ...	2
2	3	1	A series	2
3	4	1	A	2
4	5	1	series	2
5	6	1	of escapades demonstrating the adage that what...	2
6	7	1	of	2
7	8	1	escapades demonstrating the adage that what is...	2
8	9	1	escapades	2
9	10	1	demonstrating the adage that what is good for ...	2

Fig. 1. First 10 instances of dataset

B. Preprocessing and cleaning

Before moving forward with any type of preprocesing and cleaning, split the dataset in to train and test to validate the model at the end. The anonymous data with no labels is required to check if the model is able to predict the sentiments of the phrase accurately or not. This step is followed by tokenization, stop words removal, punctuation removal, Vectorizing and finally CNN.

The splitiing of train and test data from the original dataset is very easy. With the help of "traintestsplit" function given by sklearn.modelselection package. this functions takes the dataset as input along with hyperparameters like test split ratio, random state, train size,and test size.The code snippet is added in the Appendix under title Train Test split
Following are the terms that play significant role in sentimental review analysis.

1) Tokenization: Tokenization in NLP is a very simple process. It's essentially a process to break a character into bits, called the token, and to throw other characters away, like punctuation.Every single words and punctuation are seperated.

2) Stopwords Removal: Stopwords are the commonly used words that are used abundantly in books or more often in a short review like 'the', 'a', 'an', 'in', etc. This words are a trouble maker if you are using bag of words framework for tokenization.

3) Punctuations: punctuations are the most important part of language in certain conditions. Wrong punctuations can lead to change of meaning and symentic.

4) Stemming: stemming is one of the methodology for finding the root word of any word. prefix and sufix along with the tense are depended for the meaning of the word but for undersanding from scratch we have to provide machine the root word so that it could understand easily.for instance running, ran ,runner, etc all will be converted to run.

5) TF-IDF: Term Frequency- Inverse Document Frequency is framework that is used to imply vectorizer to the dataset. It basically reflects the relevance of document [2]. tfidfvectorizer function takes max feature and ngram range as input parameter and returns the vectorizer model. the code snipet of the vectorizer is added in Appendix under name VECTORIZER.

The above explained stop words removal and punctuation are removed by iterating over the tokenized words from the phrases. To check for word being a stop word a dictionary of stopwords is imported from 'nltk' library. Stemming is done with the use of lancaster stemmer or porter stemmer provided by nltk.stem library. In the proposed model lancaster stemmer is used for stemming. The lancster.stem() take a words from the tokenized word documents as parameters and returns the root word. we are not using lematizing as for lematizing we need a full sentence while here we are processing tokenized single words.

C. The CNN model

The convolution neural network can now be implemented on the processed data to train the model. The cNN architecture is combination of various processing layers like convolution layer, pooling layer, flatten layer, dropout layer, and dense layer. The code snippet of model architecture is added in Appendix under name CNN architecture.

1) Convolution layer: This function extracts maximum feature from the images like edges.Upon applying filters we get the feature map.This layer uses a convolution kernel to generate a tensor for the outputs with the origin of the layer over a temporal dimension.Therefore, the tensor is synthesised in a mathematical process into one smaller. When there is a single dimension in the input matrix, it is condensed in the same dimensions; if it has n dimensions then it is synthesised in n dimensions. And using the data set is in 1 dimension, so it takes Conv1D layer into consideration.

2) Pooling Layer: Pooling layer decreases the representation spatial size, thus reducing the number of parameters and managing overlay. their are two types of pooling methods first is max pooling and second is avg pooling.

3) Flatten layer: In flatten layer the output from the last pooling layer is converted into single column which is then forwarded into dense network for classification.

4) Dense layer: Dense layers are also called fully connected layers. Here, all the nodes of the flatten layer are connected with all the neurons of the dense layer. This layers are specifically added for classification purpose. The last dense layer or we say output layer has same number of neurons as the number of classes in the dataset.

IV. Experimental Results

For the efficiency we have various hyper parameters that can be modified to change the performance of the model. The features like n-gram, batch size can be changed, while keeping the number of epochs equivalent to 10 as constant. Following is the model architecture visible in the image below. This keras model [3] has 3 convolution layer 2 max pooling layer, drop out layer, flatten layer, and 2 dense layer.

```
model.summary()

Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_16 (Conv1D)           (None, 1998, 64)          256
_________________________________________________________________
conv1d_17 (Conv1D)           (None, 1994, 128)         41088
_________________________________________________________________
max_pooling1d_11 (MaxPooling (None, 997, 128)          0
_________________________________________________________________
conv1d_18 (Conv1D)           (None, 993, 128)          82048
_________________________________________________________________
max_pooling1d_12 (MaxPooling (None, 496, 128)          0
_________________________________________________________________
dropout_6 (Dropout)          (None, 496, 128)          0
_________________________________________________________________
flatten_6 (Flatten)          (None, 63488)             0
_________________________________________________________________
dense_14 (Dense)             (None, 15)                952335
_________________________________________________________________
dense_15 (Dense)             (None, 5)                 80
=================================================================
Total params: 1,075,807
Trainable params: 1,075,807
Non-trainable params: 0
```

Fig. 2. Model summary

Below is the result table of the performed experiments. During this experiments the max feature was set to 2000, the keras default adam optimizer is used, number of epochs are 10, model architecture is same as displayed in Fig.2. The only variable parameters where batch size and the n-gram value. No overfitting issues where observed in any of the experiments.I perferomed the experiment using sgd optimizer but the accuracy equivalent to this was not achieved in 10 epochs. Thw drop out rate is set to 0.3 in every experiment.

Table:1 Result Comparison

result	$B_s = 128$ bigram	$B_s = 128$ trigram	$B_s = 64$ bigram	$B_s = 64$ trigram
Loss	0.981	0.984	0.978	0.97
Accuracy	0.612	0.613	0.608	0.61
Precision	0.664	0.683	0.672	0.67
Recall	0.520	0.484	0.502	0.50
F1 score	0.582	0.565	0.573	0.57

It is observed from the comparision table that the precision and recall are inversely proportional as when the precision is higher then the recall is lower. we can see that loss in model with batch size 128 is higher but the accuracy is not affected.

V. Conclusion

In this Literature review, we took a brief overview of sentimental analysis and convolutional neural network. we understood the factors which plays important role in text preprocessing. Given the rotten tomatoes movie review dataset, the proposed model has performed well on trigram compared to bigram. Their were also few hidden pillars that supported the model which are optimizer, activation function, and softmax function. As observed in the comparison table the accuracy of trigram is slightly higher than the bigram. However, the difference is not significant even. However, it can be observed that their is fine difference between the bi-gram models with different batch size. So, we can say batch size 128 performs better on bigram CNN model compared to batch size 64.

References

[1] X. Ouyang, P. Zhou, C. H. Li and L. Liu, "Sentiment Analysis Using Convolutional Neural Network," 2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing, Liverpool, 2015, pp. 2359-2364.

[2] J. Ramos. Using TF-IDF to Determine Word Relevance in Document Queries. Technical report, Department of Computer Science, Rutgers University, 2003

[3] challet, Franois and others,"Keras", challete2015keras, 2015, https://keras.io.

VI. Appendix

Stopwords and punctuation removal

```
for l in range(len(documents)):
  label = documents[l][1]
  tmpReview = []
  for w in documents[l][0]:
    newWord = w
    if remove_stopwords and (w in stopwords_en):
      continue
    if removePuncs and (w in punctuations):
      continue
    if useStemming:
      newWord = lancaster.stem(newWord)
    if useLemma:
      newWord = wordnet_lemmatizer.lemmatize(newWord
    )
    tmpReview.append(newWord)
  documents[l] = (tmpReview, label)
  documents[l] = (' '.join(tmpReview), label)
print(documents[0])
```

Train Test Split

```
X_train, X_test, Y_train, Y_test = train_test_split(
    df['Phrase'], df['Sentiment'], test_size=0.3,
    random_state=2003)
```

Vectorizer

```
from sklearn.feature_extraction.text import
    CountVectorizer , TfidfVectorizer

vectorizer = TfidfVectorizer(max_features = 2000,
    ngram_range=(1, 3))
```

CNN architecture

```
model = Sequential()
model.add(Conv1D(filters=64, kernel_size=3,
                 activation='relu',
```

```
                 input_shape=(2000,1)))
model.add(Conv1D(128, kernel_size=3, activation='
    relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(Conv1D(128, kernel_size=3, activation='
    relu'))
model.add(MaxPooling1D(pool_size=2))
# model.add(Conv1D(128, kernel_size=3, activation='
    relu'))
# model.add(MaxPooling1D(pool_size=1))
model.add(Dropout(rate = 0.3))
model.add(Flatten())
model.add(Dense(15))
model.add(Dense(num_classes, activation='softmax'))
```