Assignment 2 - Multi-class Sentiment Analysis using Deep Learning

https://github.com/imrohu/NLP/blob/master/Assignment2NLP.ipynb
Speaker : Dr. T. Akhilan
Author : Rapelli, Sai Rohit
Student ID: 1107345
Email: srapelli@lakeheadu.ca
20 March 2020

***Abstract*—Sentiment Analysis defines the viewpoint of the sentence by considering word polarity as either positive or negative. This paper introduces Convolution neural networks being applied to predict a successful solution. In this lab, we are going Implement a scalable and robust convolutional neural network-based solution for the problem of text-based movie review multi-class sentiment analysis which helps to achieve the best possible accuracy, precision, F-1 score. We use the raw train data of Rotten Tomatoes movie reviews dataset[3]. We run the output through the Soft-Max layer for multi-class classification to get the integer values between range 0 and 1 for accuracy and f1-score[6].**

I. Introduction

A. Sentiment Analysis

It uses Natural Language processing techniques, Text Analysis, Computational Linguistics to systematically classify the information, and to analyse successful information states.

B. Multi Class Sentiment Analysis

Multi-class sentiment analysis is an extension to binary sentiment analysis which only classifies the given text as positive or negative.

Firstly, we need to connect to a GPU run time from Google Co-lab and process our housing dataset. Second, we will create a custom model, from scratch, using 1D convolution layers Third, we will create a custom method to train our new model in batches and let it run for a set number of epochs. Finally, we will evaluate our model on the testing dataset to see how it performs [2].

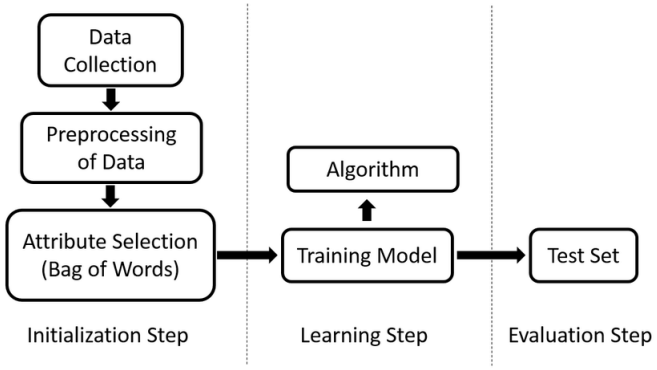

Fig. 1. Sentiment Analysis Flowchart [7]

II. Background

Sentiment analysis refers to the automatic collection, aggregation, and classification of data collected online into different emotion classes. Although much of the research pertaining to text sentiment analysis focuses on the binary and ternary classification of such results, less attention has been paid to the task of multi-class classification. 1.First, connect to a GPU run time (Run time - Change run time type - Hardware accelerator - GPU - Save, then connect.Next, download the 'train.tsv' dataset. 3. Upload it to your co-lab notebook. 4.Finally, let's import the libraries to work with our dataset[3].

III. Literature Review

Multilayer perceptron, or other neural networks are networks that are entirely connected. In this network every neuron in one layer is connected in the next layer to all other neurons. Whereas in CNN the neuron is related to the neighbor's nearest neuron. CNN makes networking complex easy. We want to classify movie reviews (obtained from nltk corpus) as either pos or neg based on their text, We vectorize the review text first and convert the labels to binary. We also need to preprocess our text for improved model performance. Punctuations and common English words (stop-words) don't add anything to our sentiment frequencies. Also, we should consider treating same base words as same words.[1]

IV. Proposed Model Steps

Step 1: First,let's read and clean the dataset by using dropna command.
Step 2: Next,we will split the dataset into the input (X) and output (Y) data-sets.
Step 3: Finally, we need to split it into training and testing portions.
Step 4: Firstly,let's import the libraries to build our network.
Step 5: Next,let's define our model.
Step 6: let's also import the optimizer and performance measure libraries.
Step 7: Next, let's create a method for running the batches of data through our model.
Step 8: Finally, let's train the model and We will test the model to see its performance on the testing dataset[4].

V. Experimental Analysis

We need to perform the following visualization using the Python and Pandas libraries in order to print the first ten records of the dataset. Here we import the pandas library to read our dataset, get the train/test package from *sklearn* library next we are importing *numpy* library to manipulate the data. We need to import the optimizer Adam and SGD, cuda package for performing optimisation and import the L1 Loss and Cross Entropy Loss package for measuring the performance. We Import accuracy, precision recall score package from sklearn for measuring the score[5].

VI. Methods/Approach

A. Import Dataset

Use the raw train data of Rotten Tomatoes movie reviews available at:
https://raw.githubusercontent.com/cacoderquan/Sentiment-Analysis-on-the-Rotten-Tomatoes-movie-review-dataset/master/train.tsv into a Colab notebook.

We retrieve the given dataset and put it in a list of tuples called data. We shuffle our data to ensure an even mix of category representation when we do the train/test split later Set the seed value as a constant to ensure the same random shuffle every time.

B. Text Normalization

1.First we import libraries for stop words removal, stemming and lemmatization.
2.We will see the impact of setting the True/False flags on the model performance later.
3.Then we clean and normalize the review text (next slide) based on the flags set.
4.We also change the label of review to binary and combine all words into a review sentence.

C. Processing our dataset

1.Split dataset 70/30 and place data into dataframes to get numpy arrays out later, for our vectorizers and Pytorch models.
2.We use sklearn's built in vectorizers to convert our text reviews into vectors the size of our overall vocabulary.
3.We can specify removal of stop words and use of n-grams within the vectorizer.
4.We first fit the vectorizer on all text so that it contains a list of all words, and then transform our training and testing data, to be used by our model later. Train/test split ratio is 70:30 using sklearn.model selection library and random state is set to 2003.

```
x_train,x_test,y_train,y_test =
train_test_split(X,Y,test_size=0.3,
random_state=2003)
```

```
epochs=10
batch_size=128
```

Importing required PyTorch libraries:

```
import torch
from torch.nn import Conv1d
from torch.nn import MaxPool1d
from torch.nn import Flatten
from torch.nn import Linear
from torch.nn.functional import relu
from torch.utils.data import DataLoader,
TensorDataset
from torch.nn.functional import relu, softmax,
sigmoid
from torch.utils.data import DataLoader,
TensorDataset
from torch.optim import SGD, Adam
from torch.nn import L1Loss, CrossEntropyLoss
```

Our Trained Model snippet code:

```
def 1107345(data):

model2 = Sequential()

model2.add(Embedding(max_features,
150, input_length=max_words))

model2.add(SpatialDropout1D(0.2))

model2.add(Conv1D(32, kernel_size=3,
padding='same', activation='relu'))
model2.add(MaxPooling1D(pool_size=2))

model2.add(Conv1D(64, kernel_size=3,
padding='same', activation='relu'))
model2.add(MaxPooling1D(pool_size=2))

model2.add(Flatten())
model2.add(Dense(5, activation='softmax'))
```

Recall code:

```
def recall_nlp(true, pred):
    t_p = Ke.sum(Ke.round(Ke.clip(true *
    pred, 0, 1)))
    p_p = Ke.sum(Ke.round(Ke.clip(true,
    0, 1)))
    recall = t_p/ (p_p + Ke.epsilon())
    return recall
```

Precision Code:

```
def precision_nlp(true, pred):
    t_p = Ke.sum(Ke.round(Ke.clip(true
    * pred, 0, 1)))
    p_p = Ke.sum(Ke.round(Ke.clip(pred,
    0,1)))
    precision = t_p / (p_p +
    Ke.epsilon())
    return precision
```

F1 Score Code:

```
def f1_nlp(true, pred):
    precision = precision_nlp(true, pred)
    recall = recall_nlp(true, pred)
    return 2*((precision*recall)/
    (precision+recall+Ke.epsilon()))
```

Save Model:

```
model.save("1107345.pt")
```

VII. Features

A. BOW - Bag Of Words

The simplest text vectorization technique.
Step I: Cleantext
Step II: Tokenize
Step III: Create a vocabulary.
Step IV: Word count. For a given input text, it outputs a numerical vector, which is simply the vector of word counts for each word in the vocabulary.

B. TF-IDF

TF-IDF focuses on finding the 'valuable' classification key words for each document. TF of a word in a document increases its value, and DF decreases it. The vector of these key words are then fed to deep learning models as features in order to determine the topic of the documents that contains the key word.

C. Word2Vec

BoW and TF-IDF are different from Word2vec. Word2vec produces one vector per word, whereas BoW and TF-IDF produce one number(a word count or normalized count). Word2vec is great for digging into documents and identifying content and subsets of content. Word2vec represents each word's context, the n-grams of which it is a part. BoW is a good, simple method for classifying documents as a whole.

VIII. Applications

Multi class CNNs have recently been proposed and immediately achieved the state-of-the-art performance levels in several applications such as personalized biomedical data classification and early diagnosis, structural health monitoring, anomaly detection and identification in power electronics and motor-fault detection. Another major advantage is that a real-time and low-cost hardware implementation is feasible due to the simple and compact configuration of 1D CNNs that perform only 1D convolutions[2].

Conclusion

Multi-class classification has always been a challenging task given the complexity of natural languages and the difficulty of understanding and mathematically "quantifying" how humans express their feelings. Upon completing the above experiment are:

Accuracy	0.6403092827544962
F1 Score	0.6355485329665064
Precision	0.6508624560708008
Recall	0.6214276560297322

Table.1 Results

References

[1] 1D Convolutional Neural Networks and Applications – A Survey 123456 Serkan Kiranyaz , Onur Avci , Osama Abdeljaber , Turker Ince , Moncef Gabbouj , Daniel J. Inman.
[2] S. Kiranyaz, T. Ince, A. Iosifidis, M. Gabbouj, Operational Neural Networks, (2019). http://arxiv.org/abs/1902.11106.
[3] Multilingual Multi-class Sentiment Classification Using Convolutional Neural Networks, Mohammed Attia, Younes Samih, Ali Elkahky, Laura Kallmeyer.
[4] A. Balahur, R. Steinberger, M. Kabadjov, V. Zavarella, E. Goot, M. Halkia, B. Pouliquen, and J. Belyaeva, "Sentiment Analysis in the News", in Proceedings of the Seventh International Conference on Language Resources and Evaluation, European Language Resources Association (ELRA), 2010.
[5] S. Gebremeskel, "Sentiment Mining Model for Opinionated Amharic Texts," Unpublished Masters Thesis, Addis Ababa University, Addis Ababa, 2010.
[6] https://github.com/imrohu/NLP/blob/master/Assignment2NLP.ipynb
[7] https://images.app.goo.gl/dzMri3DAkZSLJpHi9