Multi-class Sentiment Analysis using Deep Learning

Umang Mehta
Student Id: 1117269
Department of Science
Lakehead University
Thunder Bay, Canada

***Abstract*—As Machine Learning is developing day by day and recently the deep learning has grabbed the attention with analysis in various fields without respect to specific domain. Sentiment analysis is one of the most known and useful task of natural language processing. The report describes Implementation, result, of a one-dimensional convolution (Conv1D)-based neural network for the problem of text-based movie review multi-class sentiment analysis. The sentiments are consisting in 5 classes based on the movie review is very negative, somewhat negative, neutral, somewhat positive, and very positive. The designed model is not using advanced sequence handling components such as GRU, RNN, LSTM and only stick to the Convolution and dense layers and the associated operations like max pooling, Ave pooling and non- linear activation functions such as ReLu, Sigmoid, tanh, SoftMax, Leaky ReLU. The performance of model is evaluated based on Accuracy, Recall, Precision, and Figure-of-Merit (f-1) score.**

***Index Terms*—Sentiment analysis, NLP, 1D CNN, Recall, precision, figure-of-merit (f-1) score.**

I. Introduction

In this fast-growing world of internet and social media, people love to speak their view or opinions by writing it over such platform. Understanding the Sentiment of the such sentences help companies to improve their services and help to enhance the marketing techniques, politics, and security. On the other hand, this analysis can not be done by any human manually because there are higher changes to make mistakes and cannot do it for millions of sentences as well as it's a time-consuming process on the same time.

To overcome such issue and save money and time, the advanced world came up with sentiment analysis also known as opinion mining. There are mainly two classification as required by the application, first, Binary classification, second, Multi-class classification. The binary classifier identify text that expresses sentiment either positive or negative. On the other hand, multi-class classifier identifies more than two classes such as very negative, somewhat negative, neutral, somewhat positive, and very positive.

The model is trained on 70 percentage of Rotten Tomatoes movie review dataset and also tested on the same dataset but with the rest 30 percentage. The dataset is cleaned first and then split into train and test set. To clean the data This report will describe the required dataset, implementation using 1D CNN, cleaning data, vectorization, testing and evaluation of model.

II. Background/ Literature review

In the paper, they used two different databases in which one was with binary labels and another one was with multi-class labels. The vectorization method used for the binary labeled database was bag of words and skip gram word2vec models which were followed by various classifiers including SVM, random forest and logistic regression. Alternatively, for the multi class database, they implemented the recursive neural networks. The Recursive neural networks (RNN) can not express some relations likewise negated positives or negated negatives. The extensions that are capable of dealing with negation errors is resent area of research in deep learning and natural language processing. The performance of the classifier on the movie reviews here is evaluated by checking the fraction of correctly classified reviews from the test sets they chose. For the binary classification, the baseline for the accuracy was at 84 percent and using random forest, SVM and logistic regression classifiers were 84, 85.8 and 86.6 percentage respectively. By comparing these results to the RNN method or with multi-class method, they found that the accuracy was found to be less than comparably.

III. Proposed Model

A. Work Environment

To build the application, I use Google Colab, which allows arbitrary python code to be written and executed over the web, and is especially well suited for machine learning, data analysis and education. More technically, Colab is a host Jupyter notebook service that needs no setup to be used, thus providing free access to computing resources like GPUs. For easily store and access data, we need to mount the google drive with Colab.

```
from google.colab import drive
drive.mount('/content/gdrive')
import os
os.chdir("gdrive/My Drive/")
```

Now, the code will return the link for authenticate and we need to provide that authorization code to the required field then our drive will be mounted successfully.

B. Dataset

I used Rotten Tomatoes movie review dataset to train and test data for movie reviews. The model will be trained after

cleaning with 70 percentage of data and will be tested on rest 30 percentage. To give reference or bind the dataset into our variable, there are tow methods. One is providing path from google drive:

```
data = pd.read_csv('train.tsv', sep='\t')
```

second is providing web link:

```
data =
    pd.read_csv('https://raw.githubusercontent
.com/cacoderquan/Sentiment-Analysis-on-the
-Rotten-Tomatoes-movie-review-dataset/master/
train.tsv', sep='\t')
```

TABLE I
DATASET FIRST 5 ROWS

PhraseId	SentenceId	Phrase	Sentiment
1	1	A series of escapades demonstrating...	1
2	1	A series of escapades...	2
3	1	A series	2
4	1	A	2
5	1	series	2

Table 1 is depicting first five tuples of the dataset. The phraseId is a unique id for each sentence, the sentenceId is some tokens of the same sentence and remained same for same sentence, Phrase is the review sentence or some review tokens, and sentiment is the sentiment class from 0 to 4 as re feared as(very negative, somewhat negative, neutral, somewhat positive, and very positive).

In table 2, the data is categorised by the five sentiment classes and related sentences.

TABLE II
CATEGORISED DATA BY SENTIMENT CLASSES

Sentiment class	number of phrases
2	79582
3	32927
1	27273
4	9206
0	7072

The full Sentences available in dataset : 8544

C. Cleaning data

Data cleaning is the method of preparing data for analysis by removing or altering data that is inaccurate, incomplete, irrelevant, duplicated or formatted improperly. The basic need in the data science is to get form data for analysis from the raw data and thus the data scientists usually spend 80 percentage of their time in cleaning and rest 20 percentage of time in analyzing. The main goal to do data cleaning is that the structured data can maximize the accuracy of the network. There are several steps to clean data in preprocessing step.

Tokenization: Word tokenization is the important task in Natural language processing. It split every word from the given text as each word is really useful to classify, counting, and understanding of particular sentiment. To achieve this we use word_tokenize method from the Natural Language Tool kit(NLTK) library. Moreover, tokens are basic need for some methods like stemming and lemmatization. Natural Language toolkit has main module that is tokenize, which is further divided in sub-modules: 1. sentence tokenize: Sentence tokenization is useful when we want to count average word per sentence.For example,

```
text = "God is Great! I won a lottery."
Output: ['God is Great!', 'I won a lottery ']
```

2. word tokenize: To split the word we use the word_tokenize() method. The generated tokens can also be provided as input to do further cleaning steps such as punctuation removal, numeric character removal or stemming. Any Machine learning network is trained on numerical data so tokenized words are really important in such conversation.

```
text = "God is Great! I won a lottery."
Output: ['God', 'is', 'Great', '!', 'I',
    'won', 'a', 'lottery', '.']
```

Stop words: The main pre-processing task is to filter not useful data and such words are referred as stop words. So, the stop word is a commonly used word such as “the”, “a”,”an”,”in”. These words can decrease the sentiment accuracy and waste the processing time. To remove such words, we can store a list of such words and we can easily remove those. NLTK (Natural Language Toolkit) in python has a list of stop words stored in 16 different languages. The application areas where stop words are in use: 1. Supervised machine learning – removing stop words from the feature space 2. Clustering – removing stop words prior to generating clusters 3. Information retrieval – preventing stop words from being indexed 4. Text summarization– excluding stop words from contributing to summarization scores and removing stop words when computing ROUGE scores

Types of stop words: Determiners – Determiners tend to mark nouns where a determiner usually will be followed by a noun examples: the, a, an, another Coordinating conjunctions – Coordinating conjunctions connect words, phrases, and clauses examples: for, an, nor, but, or, yet, so Prepositions – Prepositions express temporal or spatial relations examples: in, under, towards, before

Punctuation: Same as the stop words, punctuation are not useful at some extinct. So, we are removing punctuation by looping through our data are keeping those data which is not in string.punctuation list.

Stemming and Lemmatization: We come across situations

in the Natural Language Processing areas where two or more words have a similar origin. The three terms-agreed, agreed and acceptable, for example, have the same root word. A search that contains both of these words will consider them as the same word that is the source. So, linking all the words into their root word becomes important. The NLTK library has methods to connect this and to give the output that shows the word root. Stemming effectively eliminates a word's suffix, and reduces it to its root word. For example: "Flying" is a word and its suffix is 'ing', then it removes the 'ing' so we get root word 'fly'. The main goal is to reduce each word's inflectional forms into a specific base word, or root word or stem word. Inflection is a process of word forming in which a word is changed to convey different categories of grammar such as stress, case, speech, aspect, person, number, gender, mood, animation, and meaning.There are majorly 2 errors: 1. Over Stemming: It referred same root word for two or more different stem words, this is also called as false positive.For Example, universal, university, universe are stemmed to universe which is wrong. 2. Under Stemming: Under-stemming is when there are no two words to the same root that should be stemmed in. This is sometimes called a false negative.For Enample, alumnus, alumni, alumnae.

Lemmatization :This takes the morphological study of the words into consideration. Extracting the appropriate lemma requires dictionaries for each language to provide this form of analysis and to catch the root term.

After performing data cleaning we are dividing our dataset into training and testing dataset with 70 and 30 percentage ration.

D. Vectorization:

The features for machine learning is basically numerical attributes. But Sometime database does not contain only numeric data same like our dataset. So we need to do conversation of such data and its called featurization. And process of converting text into vector is called vectorization. To convert string data into numerical data we can use many methods as follow: Bag of Words: It is basic model used in natural language processing. Every word order in the document is discarded and informs us only whether the weather term is in the document or not. If consider n word per time is then it is called n gram. Same as we have bigram that is using two words at a time , trigram – using three word at a time. To convert the sting to the vector I am using CountVectorizer function we can convert text document to matrix of word count. Matrix which is produced here is sparse matrix. TF-IDF: TF-IDF stands for Term Frequency-Inverse Document Frequency. It basically show important word from the corpus ordataset. TF-IDF has two concept Term Frequency(TF) and Inverse Document Frequency(IDF) Term Frequency: It Shows how frequently the word appear in the dataset or corpus. Since each sentence is not the same length, it could be likely that a word occurs more often in the long sentence than in the sorter sentence. Term frequency can be defined as:

$$\text{TF} = \frac{Number of Time Word appear in the document}{Total Number of a word in the document} \quad (1)$$

Inverse Document Frequency: It is also used for searching importance of the word. It is focused on the fact that words which are less frequent are more insightful and significant. IDF is represented by formula:

$$\text{IDF} = \log_{10} \frac{Number of Documents}{Number of Document in which word apper} \quad (2)$$

TF-IDF: TF-IDF is a multiplication of TF table and IDF table. So it reduces the common words' value which are used in different documents.

E. Building our network

To solve any problem through machines by machine learning method it mainly depends on the finely formed model. A CNN works well to recognise simple patterns within the data, which can then be used in higher layers to shape more complex patterns. CNN is working same way for any number of dimensions. The only change within these is the structure and input data and how it is filtering it. A 1D CNN is very useful when you intend to extract interesting features from the overall data set's shorter (fixed-length) segments.

The figure 1 depicting example of 1D CNN network example. The 1D CNN model extracts features from sequences data and maps the internal features of the sequence. As show in figure 2, the layers are connected with each other sequentially and they are sharing parameters just because they have local connectivity. Thus, the CNN's functions are able to learn by its own the useful feature and able to classify in particular category.

I am using sequential API from keras, which is high level neural network API, for our dataset to go through one by one to each layer. Here, I am also using dense, dropout, flatten, Convolutional 1Dimentional and max pooling layers, where the dense layer is to receive an input form the previous layer. The Dropout layer is used to prevent a model from overfitting as shown in figure 3. Dropout works by setting the outgoing edges of hidden units randomly to 0 at each change of the training process. Based on prediction error on the training data and the evaluation data, we can find if it is underfits or overfits the training data.

The pooling Layer is always used in the end of convolutional layer to get those input feature maps and combine it with the activation features. To reduce the amount of parameters and computation of the network, The pooling layer is reduce the spatial size of the representation. It performs in each feature

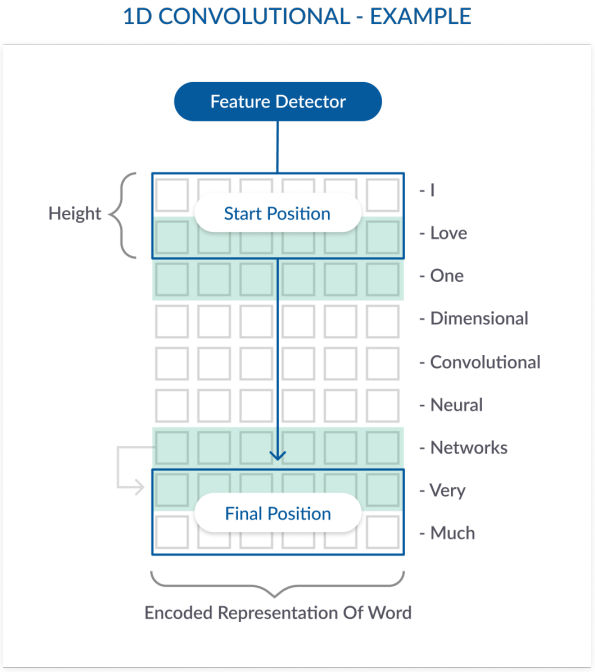

Fig. 1. 1D CNN Example

Source: https://missinglink.ai

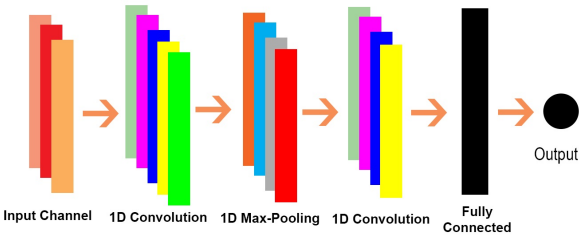

Fig. 2. CNN Architecture

Source: https://i.stack.imgur.com/Lbdyv.png

map independently. Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network. Pooling layer operates on each feature map independently. The 1D CNN is generally used for Analysis of a time series of sensor, Analysis of signal data over a fixed-length period, for example, an audio recording. In the below listing I have shown the code of my designed network and the summary of the model is shown in the table 3.

```
model = Sequential()
model.add(Conv1D(filters=64, kernel_size=3,
                 activation='relu',
                 input_shape=(2500,1)))

model.add(Conv1D(filters=64, kernel_size=3,
    activation='relu'))
model.add(Conv1D(filters=64, kernel_size=3,
    activation='relu'))
model.add(MaxPooling1D(pool_size=1))
model.add(Dropout(rate = 0.25))
model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dense(num_classes,
    activation='softmax'))
```

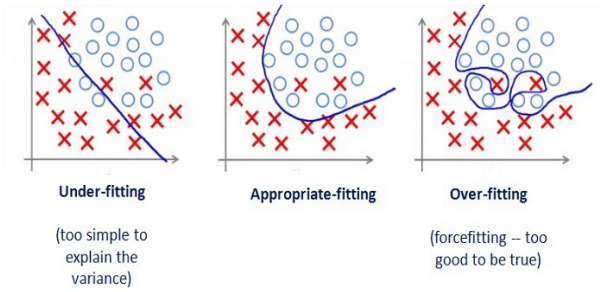

Fig. 3. Fitting issues

Source: https://miro.medium.com

TABLE III
MODEL SUMMARY

Layer (type)	Output Shape	Param
conv1d_1 (Conv1D)	(None, 2498, 64)	256
conv1d_2 (Conv1D)	(None, 2496, 64)	12352
conv1d_3 (Conv1D)	(None, 2494, 64)	12352
max_pooling1d_1	(MaxPooling1 (None, 2494, 64)	0
dropout_1 (Dropout)	(None, 2494, 64)	0
flatten_1 (Flatten)	(None, 159616)	0
dense_1 (Dense)	(None, 64)	10215488
dense_2 (Dense)	(None, 5)	325

Total params: 10,240,773
Trainable params: 10,240,773
Non-trainable params: 0

IV. TRAINING, ANALYSIS AND RESULTS

After defining the model, its time to train the model. The better accuracy and good performance of the model is based on certain parameters like batch size, kernel size, number of epochs, activation function, optimizer, dropout rate, and last but not least number of layers. The performance is analyzed based on the accuracy, loss, F1 Score, recall measure, and precision measure. Precision means the percentage of your results which are relevant, refer figure 4. Whereas recall is the percentage of total relevant results correctly classified by algorithm. I have tried different optimizers to test the model and improve the performance of model which is shown in table 4.

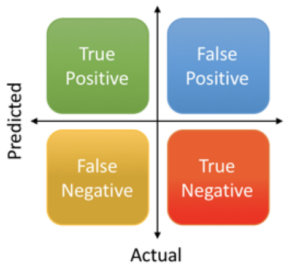

Fig. 4. Recall Precision

Source: https://miro.medium.com/

TABLE IV
RESULTS OF DIFFERENT OPTIMIZERS

Optimizer	loss	Accuracy	F1 score	Recall	Precision
Adam	1.093	0.64	0.58	0.60	0.58
Adamax	1.029	0.63	0.59	0.58	0.60
Adadelta	1.022	0.63	0.43	0.33	0.60
Nadam	1.112	0.63	0.60	0.69	0.59
RMSProp	1.102	0.46	0.35	0.56	0.46
SGD	1.19	0.50	0.50	0.49	0.50
Adagrad	5.089	0.50	0.45	0.71	0.31

V. CHALLENGES

The main concern is to get a satisfied result from the model and to do so we have to choose right parameters for model such as optimizer, batch size, number of epochs, number of layers, and activation function. Initially I tried with only layer and SGD optimizer but the result was disappointed. Then after I tried to shuffle data and changed some parameters like batch and activation function. As a result, I got somewhat improvement by 1 to 2 percentage. In a next chance, I decided to add 1 extra layer and I changed the optimizer to Adamax. This time I got a good hike. I keep changing parameters and the results is shown in the table 4. The Main issue to be taken care of in training is overfitting and under fitting. Underfitting means The model is not able to the structured data or trainable parameters are less and so it is not able to fit it whereas overfitting is happened when the trainable parameters are high and run for more epochs, refer figure 3.

REFERENCES

[1] Hadi Pouransari, Saman Ghili, Stanford University "Deep learning for sentiment analysis of movie reviews"
[2] kavita ganesan, “What are Stop Words?” [Online]. Available: https://kavita-ganesan.com/what-are-stop-words/.XnfGiWhKg2x
[3] “Python - Word Tokenization” [Online]. Available: https://www.tutorialspoint.com/python_data_science/python_word_tokenization.htm
[4] “Tokenize Words and Sentences with NLTK?” [Online]. Available: https://www.guru99.com/tokenize-words-sentences-nltk.html
[5] “Python - Stemming and Lemmatization” [Online]. Available: https://www.tutorialspoint.com/python_data_science/python_stemming_and_lemmatization.htm
[6] Tushar Srivastava ,“NLP: A quick guide to Stemming” [Online]. Accessed on: Aug 6, 2019 Available: https://medium.com/@tusharsri/nlp-a-quick-guide-to-stemming-60f1ca5db49e