NLP Assignment 2

1st Khanam K.Zainab
dept. of Computer Science
Lakehead University
Ontario, Canada
kkhanam@lakeheadu.ca

***Abstract*—The main aim of this assignment was to design a efficient Convolutional Neural Network-based solution for the problem of text-based movie review multi-class sentiment analysis.The dataset was splitted with a ratio of 70:30 with a random state of 2003. As, the dataset is highly imbalanced, re-sampling method has been used to address this issue and also to increase the performance of the model spatial dropout has been used to overcome the overfitting issue. I have obtained an Accuracy of 0.605, F1 score of 0.433, precision score of 0.392 and recall score of 0.483.**

I. Introduction

The model designed from the rotten tomatoes movie rating dataset is a multi-class sentimental classification problem. The dataset has 5 features which consists of Phrase_ID, Sentence_ID, Phrase and the Sentiment score from 0-5. The table I shows the score of the sentiments and II shows the first 5 rows of the dataset.

Sentiment	Label
negative	0
somewhat negative	1
neutral	2
somewhat positive	3
positive	4

TABLE I
SHOWS THE SENTIMENT AND ITS SCORE RESPECTIVELY

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

TABLE II
SHOWS THE FIRST 5 ROWS OF THE DATASET

The dataset is highly imbalanced since there are not equal number of each classes and the figure 1 shows the distribution of each classes throughout the dataset. The 1 shows that the neural class (2) is highest compare to the rest of the classes and also the quantity of negative (class 0) and positive (classs 4) is quite low. Since, the dataset is very imbalanced, the performance of the model will not be high as the quantity of each of the classes vary immensely. Thus, in order to address this issue, resampling method is used which takes randomly equal number of classes. So for this dataset after splitting the dataset in the ratio of 70:30. As, the quantity of class 0 and class 4 is very low, by oversampling the training data, the class 0 and class 4 features the dataset was balanced. So, by using random state of 2003(as stated in the question), I have taken equal number of each classes so I have taken 75000 number of samples randomly for each of the classes in order to have a balanced training dataset.

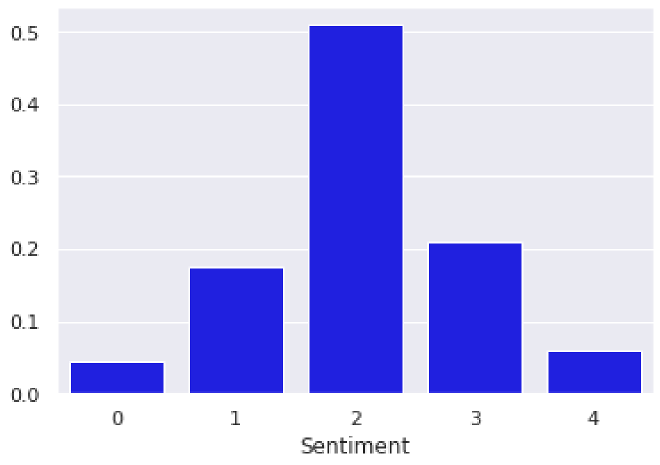

Fig. 1. shows the distribution of each of the classes

In order, to prevent overfitting from occuring I have used a dropout function which drops out the redundant features by ignoring the neurons which takes into acccount of such features. As fully connected layers occupies most of the parameters, the neurons in each of the layers tend to become co-dependent among each other during the training phase which leads to over-fitting of training data. Hence, dropping out the redundant features which are highly correlated with each other, helps to solve the over-fitting issue. However, regular dropout does not tend to regularize the activations and instead will cause the learning rate to decrease thus reducing the performance of the model. Hence, I have used an upgraded drop out function known as SpatialDropout1D which helps to promote independence between feature maps that are highly correlated with one another, and thus enables the learning rate

https://github.com/Zainab-Lakeheadu/NLPassignment2 to rather increase.

II. Background

For my assignment I have used neural network model which includes the Convolutional Neural network (CNN) which is defined as linear operation that uses general matrix multiplication in atleast one of the layers [1]. CNN has been used widely in many real life applications such as for image classification datasets [2]. Also, in my neural network model I have used Adamax as my optimizer. Optimizers are algorithms that can aid to change the attributes of neural network such as weights and learning rate in order to decrease the losses and thus helps to get results faster, some of the optimizers are - Adam, Adamax,Stochastic gradient [3].Artificial neural networks also have activation function of a node in artificial neural networks and it is defined as the output of that node given an input or set of inputs [4].Furthermore, in machine learning, over-fitting is an issue found in training artificial neural networks where the performance of the training data is higher than that of the testing data. In order to overcome this problem dropout layer is used dropout refers to ignoring units (i.e. neurons) during the training phase of certain set of neurons which is chosen at random. As a fully connected layer is very much occupied with most of the parameters, the neurons of the layers start to become co-dependent amongst each other during the training phase which leads to over-fitting of training data. Hence, dropping out redundant features helps to solve the over-fitting issue.

III. Steps carried out for Completing the Assignment

A. Steps for pre-processing the Dataset

The phrase column of the dataset was pre-processed by lowering all the cases, removing punctuation, stopwords and lemmatizing all the words. Thus, after cleaning the phrases the new column has been named as pre-processed text which is shown below in Table III.

Phrase	Pre-processed text
A series of escapades	a series of escapades
A series of escape	a series of escape
A series	a series
A	a

TABLE III
SHOWS THE PHRASE AND ITS PRE-PROCESSED CLEAN TEXT

The Table III shows that after pre-processing all the text has been converted to lowercase and also the punctuation and stopwords have been removed. In the new clean pre-processed text lemmatization has been used, which groups the different forms of the similar word. Then, the dataset was spilted at a ratio of 70:30 where 70% was used in the training dataset and 30% of the dataset was used as the testing dataset and the dataset was splitted with the aid of sklearn library.

B. Steps for the neural network model

After creating the training and testing data, total number of features for the pre-processed column was calculated, which is the total number of unique words in the corpus.Then I calculated the maximum length of the number of words in each of the reviews.I have used the keras platform for creating my model, the neural network cannot run text-strings directly so the text strings are converted into integers using tokenizer which converts words to integers and this is fed as an input to the neural network. In CNN models, the data must be transformed such that each sequence has the same length, this operation is done by sequential padding.The text of each review is formatted as a sequence of 48 index term vectors as 48 was the largest number of tokens collected from each review and if a review is less than 48 tokens , index/indices of pad will be appended to the sequence. Each token is represented as a 100 dimensional vector of the index of the corresponding term in the review text. Thus, for each review, a sequence of 48 vectors of 100 dimensions were considered.Deep learning libraries assume a vectorized representation of the data. The first layer of the CNN model is the embedding-layer which converts each integer-token into a vector of values. The tokenized integers are the values that refers to the index of vocabulary (the collection of the unique words). The embedding-layer will learn to map words with similar semantic meanings to similar embedding-vectors. Maximum 100 features which were previously calculated has been taken as input for the embedding layer. After this layer I have added a spatial dropout layer which drops out the redundant features by ignoring the neurons which takes into acccount of these features. Drop out is mainly used to prevent over-fitting from occuring. As a fully connected layer occupies most of the parameters, the neurons in each of the layers start to develop co-dependency amongst each other during the training phase which leads to over-fitting of training data. Hence, dropping out redundant features helps to solve the over-fitting issue. Regular dropout does not tend to regularize the activations and instead will cause the learning rate to decrease thus reducing the performance of the model. SpatialDropout1D helps to promote independence between feature maps and and thus enables the learning rate to rather increase. That is why in the first layer I have used spatial dropout at a rate of 0.5. Next, I have added convolutional layer with leaky relu layer as the activation function and a Max pooling layer to extract the maximum number of features. Similarly, I have added two more such convolutional layers with a two max pooling layers and in the second and third layer I have user PRelu activation function. Then, for reducing the dimension I have used a Flatten layer, Followed by a flatten layer I finally added a dense layer which is a regular layer consisting of 5 neurons in a neural network. Each neuron receives input from all the neurons from the previous flatten layer and it is densely connected. I have also used Adamax optimzer and kept the padding same in both the convolutional layers so that the model does not leave any features from the first and second layer. I have run my model for 40 number of epochs since the performnace of the model decrease from 40 epoch onwards. The code listing for each of the steos are showed in the Appendix section.

IV. Experimental Results

After pre-processing, splitting the dataset and re sampling the training data, the performance of the model increased. I have also added three convolutional layers and I have used leaky relu and PRelu activation function in order to obtain a better result. The Table IV shows the precision, recall, f1 score as well as the accuracy scores.The F1 score of 0.433 is lower than the accuracy score of 0.605. The recall score of 0.433 is higher than the precision score of 0.392.

Accuracy	Precision	Recall	F1 score
0.605	0.392	0.483	0.433

TABLE IV
COMPARISON OF THE OF THE PARAMETER AND RESULT OF THE OLD AND NEW EXPERIMENT

V. Conclusion

To sum up, the assignment was completed based on the rotten tomatoes movie dataset in which my aim was to enhance the performance of the model where I have achieved an accuracy of score of 0.605 with my testing dataset. I have splitted the training and testing data at a ratio of 70:30 with a random state of 2003. I also addressed the overfitting and imbalanced issue by using re-sampling technique and dropout layer when I was training the model. Although, the accuracy score has increased, the F1 score of 0.433 is lower than the accuracy score of 0.605. The recall score of 0.433 is higher than the precision score of 0.392 which proves that the model can correctly identify more true positives and fewer false negatives from the data.

References

[1] Hubel, David H., and Torsten N. Wiesel. "Receptive fields and functional architecture of monkey striate cortex." The Journal of physiology 195.1 (1968): 215-243.

[2] Wang, Jiang, et al. "Cnn-rnn: A unified framework for multi-label image classification." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[3] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).

[4] Haykin, Simon. Neural networks: a comprehensive foundation. Prentice Hall PTR, 1994.

[5] Hochreiter, Sepp. "The vanishing gradient problem during learning recurrent neural nets and problem solutions." International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 6.02 (1998): 107-116.

Appendix A
Importing the libraries for pre-processing the text

```
import numpy as np
import pandas as pd
from collections import
Counter
from nltk.corpus import
stopwords
from nltk.tokenize import
word_tokenize
from nltk import FreqDist
from nltk.stem import
WordNetLemmatizer
lemma=WordNetLemmatizer()
from string import punctuation
import re
import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('wordnet')
```

Appendix B
Uploading and Reading the dataset

```
url = "https://raw.githubusercontent
.com/cacoderquan
/Sentiment-Analysis
-on-the-Rotten-Tomatoes-
movie-review-dataset
/master/train.tsv"
df = pd.read_csv(url, sep='\t')
df.head(10)
```

Appendix C
Method for pre- processing the text

```
def clean_text(each_col):
    documents=[]
    for i in range(0,len(each_col)):
        document=str(each_col[i])
        document=re.sub
        ('[^a-zA-Z]',' ',document)
        document=[lemma.lemmatize(w)
        for w in word_tokenize
        (str(document).lower())]
        document=' '.join(document)
        documents.append(document)
    return documents
df['pre_processed']=
clean_text(df.Phrase.values)
```

Appendix D
Analyzing the distribution of classes and that the dataset is imbalanced

```
dist = df.groupby(["Sentiment"]).size()
dist = dist / dist.sum()
fig, ax = plt.subplots()
tips = sns.load_dataset("tips")
sns.barplot(dist.keys(),
dist.values, color = "blue");
```

Appendix E
Initializing the X (pre-processed text) and Y (Sentiment as the target variable)

```
from keras.utils
import to_categorical
X = df['pre_processed']
Y = (df['Sentiment'].values)
print(Y)
```

APPENDIX F
SPLITTING THE DATASET

```
from sklearn.model_selection
import train_test_split
X_train, X_test,
Y_train, Y_test =
train_test_split(X, Y,
test_size=0.3,
random_state=2003)
Y_test = to_categorical
(Y_test)
print(Y_test)
```

APPENDIX G
FOR RESAMPLING

```
x_train = X_train.tolist()
l = []
for i in range(len(x_train)):
    join = (x_train[i],Y_train[i])
    l.append(join)
D = pd.DataFrame
(l,columns=[
'pre_processed',
'Sentiment'])
from sklearn.utils import resample
category_0 = D[D['Sentiment']==0]
category_1 = D[D['Sentiment']==1]
category_2 = D[D['Sentiment']==2]
category_3 = D[D['Sentiment']==3]
category_4 = D[D['Sentiment']==4]

category_0_sample = resample(
category_0,replace=True,
n_samples=75000,
random_state=2003)
category_1_sample = resample(
category_1,replace=True,
n_samples=75000,
random_state=2003)
category_2_sample =resample(
category_2,replace=True,
n_samples=75000,
random_state=2003)
category_3_sample =resample(
category_3,replace=True,
n_samples=75000,
random_state=2003)
category_4_sample = resample(
category_4,replace=True,
n_samples=75000,
random_state=2003)
df_resampled = pd.concat
([category_0_sample,
category_1_sample,
category_2_sample,
category_3_sample,
category_4_sample])
X_train = df_resampled
['pre_processed']
Y_train= to_categorical
(df_resampled['Sentiment'].values)
```

APPENDIX H
PADDING AND SEQUENCING THE TEXT

```
from keras.preprocessing
import sequence,text
from keras.preprocessing.text
import Tokenizer
from keras.preprocessing.sequence
import pad_sequences
#Finding the maximum
#number of unique words

words=' '.join(X_train)
words=word_tokenize(words)
dist=FreqDist(words)
num_word=len(dist)

#Finding the maximum
#length of words
#for each document

doc_len=[]
for text in X_train:
    word=word_tokenize(text)
    l=len(word)
    doc_len.append(l)
max_doc_len=np.max(doc_len)
max_doc_len

#Initializing the number
#of epochs, batch size,
#maximum features and
#maximum words

max_features = num_word
max_words = max_doc_len
batch_size = 256
epochs = 40
num_classes=5

#tokenizing the text
#with keras tokenizer

tokenizer = Tokenizer
(num_words=max_features)
tokenizer.fit_on_texts(X_train)
X_train = tokenizer.
texts_to_sequences(X_train)
X_test = tokenizer.
texts_to_sequences(X_test)

#Initializing the padded
#sequence words in X_train
#and X_test

from keras.models import Sequential
from keras.preprocessing.sequence
import pad_sequences
X_train = sequence.pad_sequences
(X_train, maxlen=max_words)
X_test = sequence.pad_sequences
(X_test, maxlen=max_words)
```

APPENDIX I
FOR CREATING THE MODEL

```
from keras.layers import Dense,
Dropout,Embedding,
Conv1D,Flatten,MaxPooling1D,
SpatialDropout1D
from keras.losses import
categorical_crossentropy
from keras.optimizers
import Adamax
from keras.layers import
Input, Dense, Embedding, Flatten
from keras.layers
import SpatialDropout1D
from keras.layers.convolutional
import Conv1D, MaxPooling1D
from keras.models
import Sequential
from keras.layers
import LeakyReLU, PReLU
from keras.layers
import Dropout


model = Sequential()
# Input / Embdedding
model.add(Embedding(max_features,
100, input_length=max_words))
# CNN
model.add(SpatialDropout1D(0.5))
model.add(Conv1D(32, kernel_size=3,
padding='same'))
model.add(LeakyReLU(alpha=0.05))
model.add(MaxPooling1D
(pool_size=2))
model.add(Conv1D(128, kernel_size=3,
padding='same'))
model.add(PReLU(alpha_
initializer='zeros',
alpha_regularizer=None,
alpha_constraint=None,
shared_axes=None))
model.add(MaxPooling1D(pool_size=2))
model.add(Conv1D(128, kernel_size=3,
padding='same'))
model.add(PReLU(alpha_initializer='zeros',
alpha_regularizer=None,
alpha_constraint=None,
shared_axes=None))
model.add(MaxPooling1D(pool_size=2))

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

#Importing metrics for
#calculating precision
#and recall

!pip install keras-metrics
import keras_metrics

#compiling model

model.compile(loss=
'categorical_crossentropy',
optimizer='adamax',
metrics=['accuracy',
keras_metrics.precision(),
keras_metrics.recall()])

#fitting the model

model.fit(X_train, Y_train,
validation_data=(X_test,
Y_test), epochs=epochs,
batch_size=batch_size,
verbose=1)
```

APPENDIX J
SAVING & LOADING THE MODEL

```
model.save
('1110186_1dconv_reg.h5')
from keras.models import load_model
model = load_model(
'1110186_1dconv_reg.h5',
{'binary_precision':
keras_metrics.binary_precision(),
'binary_recall':
keras_metrics.binary_recall()})
```

APPENDIX K
EVALUATING THE MODEL

```
val_loss, val_acc,
val_prec, val_recall
= model.evaluate(X_test,
Y_test, batch_size=256)
print('accuracy', val_acc)
print('val_prec', val_prec)
print('val_recall',
val_recall)
f1_score = 2 *
((val_prec*val_recall)
/(val_prec+val_recall))
print('f1_score', f1_score)
```