Multi-Class Sentiment Analysis using CNN

Vashishtha Piyushkumar Master
Master of Science in Computer Science
Lakehead University
Thunder Bay, Canada
vmaster@lakeheadu.ca

***Abstract*—This paper focuses on implementation of Multi-Class Sentiment Analysis using convolutional neural network (CNN) for text based movie review.Here we have measured Accuracy, F1 Score, Precision and Recall for the model at the end. This paper gives in-depth knowledge about tpoics like Bag of Words (BoW)and TF-IDF Vectorizer. At the end of implementation I was able to acheive 61.53% accuracy.**

I. Introduction

Sentiment analysis (also known as opinion mining or emotion AI) refers to the use of natural language processing (NLP), text analysis, computational linguistics, and biometrics to systematically identify, extract, quantify, and study affective states and subjective information. Sentiment analysis is widely applied to voice of the customer materials such as reviews and survey responses, online and social media, and healthcare materials for applications that range from marketing to customer service to clinical medicine.

II. Literature Review

A. Dataset

The Rotten Tomatoes movie review dataset is a corpus of movie reviews used for sentiment analysis, originally collected by Pang and Lee. In their work on sentiment treebanks, Socher et al. used Amazon's Mechanical Turk to create fine-grained labels for all parsed phrases in the corpus. This competition presents a chance to benchmark your sentiment-analysis ideas on the Rotten Tomatoes dataset. You are asked to label phrases on a scale of five values: negative, somewhat negative, neutral, somewhat positive, positive. Obstacles like sentence negation, sarcasm, terseness, language ambiguity, and many others make this task very challenging. It has folling attributes in it:

1) PharseId
2) SentenceId
3) Pharse
4) Sentiment

B. Libraries

In order to run my model I have utilized following libraries:

- keras: Keras is an open-source neural-network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, R, Theano, or PlaidML. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible
- pandas: In computer programming, pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license.
- sklearn: Scikit-learn (formerly scikits.learn and also known as sklearn) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.
- numpy: NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.
- nltk: The Natural Language Toolkit, or more commonly NLTK, is a suite of libraries and programs for symbolic and statistical natural language processing (NLP) for English written in the Python programming language.
- matplotlib: Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy.

C. Bag of Words (BoW)

A bag-of-words model, or BoW for short, is a way of extracting features from text for use in modeling, such as with machine learning algorithms.The approach is very simple and flexible, and can be used in a myriad of ways for extracting features from documents.A bag-of-words is a representation of text that describes the occurrence of words within a document. It involves two things:

- A vocabulary of known words.
- A measure of the presence of known words.

It is called a "bag" of words, because any information about the order or structure of words in the document is discarded. The model is only concerned with whether known words occur in the document, not where in the document.

D. Word2vec

Word2vec is a group of related models that are used to produce word embeddings. These models are shallow, two-

layer neural networks that are trained to reconstruct linguistic contexts of words. Word2vec takes as its input a large corpus of text and produces a vector space, typically of several hundred dimensions, with each unique word in the corpus being assigned a corresponding vector in the space. Word vectors are positioned in the vector space such that words that share common contexts in the corpus are located close to one another in the space.[1]

E. TF-IDF

In information retrieval, tf–idf or TFIDF, short for term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus.[2] It is often used as a weighting factor in searches of information retrieval, text mining, and user modeling. The tf–idf value increases proportionally to the number of times a word appears in the document and is offset by the number of documents in the corpus that contain the word, which helps to adjust for the fact that some words appear more frequently in general. tf–idf is one of the most popular term-weighting schemes today.

$$TF(t) = \frac{Number\_of\_times\_term\_t\_appears\_in\_a\_doc}{Total\_number\_of\_terms\_in\_the\_doc}$$

$$IDF(t) = \log_e(\frac{Total\_Number\_of\_Docs}{Number\_of\_docs\_with\_term\_t\_in\_it})$$

III. Data preprocessing

In order to perform some task on the given data we need to do some preprocessing namely:

- Stop words Removal: Words like articles and some verbs are generally considered stop words because they do not help us in finding the context or the true meaning of a sentence. So, these words that can be removed without any negative impact on the final model that we are training.
- Removal of Punctuation's: We need to remove punctuation's as a part of preprocessing.
- Lemmaitization: Lemmatization means to do things properly with the use of a vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the lemma.

IV. implementation

For implementing our code, we are using google Collaboratory which has in built Jupyter notebook and performs each computation on cloud. By using this technology, it enables us to explore and perform programming tasks on cloud itself which helps user in keeping track of the work done easily, it provides with higher computational power than local computer so high end tasks can be done quickly.

In my program firstly I have imported all the libraries stated above. For the next step we are importing dataset using url provided. Then in order to train our model we firstly perform the data preprocessing steps which we discussed above. After the data preprocessing we split the data into 70:30 for training and testing respectively. Now when all the things are done the we provide our model with the training data while training I have set 20 epochs, batch size was set to 128 and I am using Adamax as optimizer for my model. While training my model I was receiving accuracy, F1 score, Precision and Recall as upwards of 67%, 66%, 72% and 61% respectively. While at the end in Testing I received accuracy, F1 score, Precision and Recall of 61.53%, 60.12%, 64.90% and 56.12% respectively. After getting these I have saved my model with 1100901_1dconv_reg name. And at the end I have ploted graphs for Training and Testing Accuracy with respect to epochs, Training and Testing F1 Score with respect to epochs, Training and Testing Precision with respect to epochs and lastly Training and Testing Recall with respect to epochs.

V. Performance of Model

In order to train my model I had set different parameters as follows:

- Epochs: 20
- Batch Size: 128
- Loss Function: categorical_crossentropy
- Optimizer: Adamax

After setting parameters as above I have achieved following performance for my model.

Model Performance	
Accuracy	61.53
F1 Score	60.12
Precision	64.90
Recall	56.12

VI. Graphs

Fig. 1. Accuracy

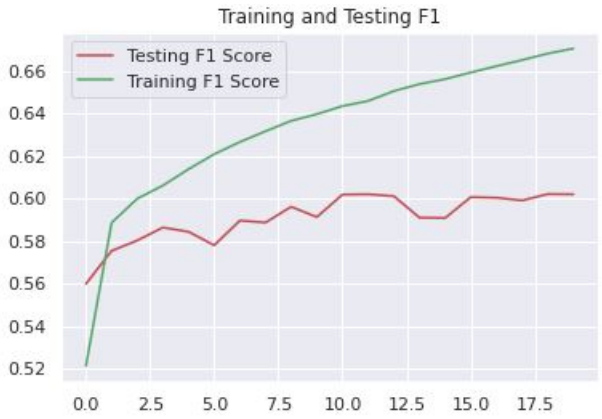

Fig. 2. F1 Score

Fig. 3. Precision

Fig. 4. Recall

VII. Conclusion

This assignment has helped and improved my knowledge about Multi-class Sentiment Analysis using CNN. Here I got know how to implement bag of word (BoW), TF-IDF, punctuation and stop word removal. Besides all these I got to learn more in depth about different optimizer, normalization technique so that we can improve our model's accuracy. And at the completion of the assignment I got better idea about natural language processing(NLP) and mainly Multi-class Sentiment Analysis. After applying all of my knowledge about sentiment analysis I was able to produce Accuracy of 61.53% while testing.

VIII. Acknowledgment

I would like to acknowledge that the efforts which were made by Professor Dr. Thangarajah Akilan was remarkable not only he introduced the topic in detail it has created huge interest for this subject in my mind. Also the work done by Andrew and Purandeep who were always available for help made by progress of learning the concept really easy.

IX. Link for my Github Repository

https://github.com/vashishtha18/Multiclass_Sentiment_Analysis

References

[1] Mikolov, Tomas; et al. (2013). "Efficient Estimation of Word Representations in Vector Space". arXiv:1301.3781
[2] Breitinger, Corinna; Gipp, Bela; Langer, Stefan (2015-07-26). "Research-paper recommender systems: a literature survey". International Journal on Digital Libraries.
[3] https://en.wikipedia.org
[4] https://keras.io/

Appendix

A. Imported Libraries in the Code

import tensorflow.compat.v1 as tf tf.disable$_v2_behavior()$
import pandas as pd
from sklearn.model$_selectionimporttrain_test_split$
from sklearn.model$_selectionimporttrain_test_split$
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt sns.set(color$_codes = True)$
import csv
import urllib.request as urllib2
from nltk import FreqDist
import nltk
import random
from nltk.tokenize import word$_tokenize$
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv1D, MaxPooling1D
from keras import backend as K
from keras.preprocessing import sequence,text
from keras.preprocessing.text import Tokenizer
from keras.layers import Input, Dense, Embedding, Flatten
from keras.layers import SpatialDropout1D
from keras.layers.convolutional import Conv1D, MaxPooling1D
from sklearn.feature$_extraction.textimportCountVectorizer, TfidfV$
from keras.utils import to$_categorical$
import time
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')

B. Model in the Code

model = Sequential()
model.add(Conv1D(filters=64, kernel$_size = 5$,
activation='relu', input$_shape = (2500, 1)$))
model.add(Conv1D(128, kernel$_size = 5$,
activation='relu'))
model.add(Conv1D(128, kernel$_size = 5$,
activation='relu'))
model.add(MaxPooling1D(pool$_size = 1$))
model.add(Flatten())
model.add(Dense(num$_classes, activation = ' softmax'$))