Multi-class Sentiment Analysis using Deep Learning

Komal Barge
Computer Science
Lakehead University
Thunder Bay, Canada
bargek@lakeheadu.ca

***Abstract*—This paper describes a language-independent, multi-class sentiment analysis model using a basic six-layer neural network architecture (Embedding, Conv1D, Dropout, Global-MaxPooling, and two Fully-Connected). I have analyzed The Rotten Tomatoes movie review corpus which was collected by Pang and Lee [1]. This corpus has been analysed in [2] where each sentence is parsed into its tree structure and each node is assigned a fine-grained sentiment label ranging from 0-4 where the numbers represent very negative, negative, neutral, positive and very positive respectively. I have predicted the classes of these sentiments by considering reviews given in the corpus. Here, I have performed multi-class classification using deep convolutional neural network(DCNN) which has given the accuracy of around 68% in training and 65% in testing. This model architecture is combination of an embedding and convolutional layers. I will present the details of the model design, experimental analysis and performance metrics in this paper.**

***Index Terms*—Convolutional layer, Neural Network, classification, review**

I. Introduction

Over the last few years, Artificial Neural Networks has become essential and one of the most dominant tools as it facilitates in handling massive amounts of data. Convolutional Neural Network (CNN), a class of deep neural networks is designed to recognize, analyze and classify images enabling deep learning for computer vision. CNN can be utilized for classification as well as regression learning methods. CNN is broadly classified into main parts: a convolution/pooling layer and a fully connected (FC) layer. In image classification, the former is responsible for breaking up the image into tiny chunks and analyzing them and the results of these are fed as an input to a FC layer in predicting the image. Whereas, in regression problems, the layers act in same way but results in a real value at the end instead of giving probabilities like classification models.

Classification models have been out there for several years and have proved to be very useful in modeling real-world problems and providing useful classification models, in both science and industrial environments and industry. At the same time, neural networks and deep learning are that in popularity, and can compute complicated problems and provide results that mimic the human brain's process of learning. A standard CNN relies on breaking up the data into smaller intricate details and analyze them in isolation helping in making an informed decision about the instance in its entirety. Classification models are divided mainly into two factors that is binary classification and multi-class classification. Convolution layer acts as a filter passing over the data matrix, creating a feature map aiding in feature extraction. This is then down sampled in the pooling layer, extracting only the most important and relevant information. Fully connected layer (Linear layer) takes the output of the previous layers, flattens them and turns them into a single vector, applies weights to predict class value.

Multi-class sentiment classification has extensive application backgrounds although studies on this topic are still relatively limited. In this paper, a multi-class sentiment classification system is proposed, which includes two parts: 1) the selection of important text features using the word embeddings, and 2) the training of multi-class sentiment classifier using machine learning algorithm.

II. Literature Review

The different approaches to sentiment analysis share the common concept of mapping a piece of text from a predefined collection onto a given label [1]. It in turn is similar to certain other NLP operations, such as categorizing text (a.k.a. document classification) and language identification. Some work on the use of unsupervised methods does exist. Sentiment analysis work is widely classified into two paradigms, information-based and statistical data-based, depending on whether or not external language-dependent information and organized information (such as POS taggers and polarity lexicons) are included in the model as features. Machine learning methods are applied in the first model to a classification of sentiments aided by knowledge-based features. Research may cover a large amount of prior linguistic expertise and feature engineering within this area.

Yang Liu et al. proposed a framework for multi-class sentiment classification which mainly included feature extraction and machine learning algorithms. They performed sentiment analysis on three public datasets containing twelve data subsets, and a 10-fold cross validation is used to achieve classification precision for each combination of feature selection algorithm, machine learning algorithm, feature set size and data subset. In their paper work, they compared the performance of four popular feature selection algorithms (document frequency, CHI statistics, information gain and gain ratio) and five popular machine learning algorithms (decision tree, naïve Bayes, support vector machine, radial base neural network

function and K-nearest neighbour) in multi-class sentiment classification [3].

Nakul Dawra et al. have used this same movie reviews dataset on an assortment of machine learning algorithms in an attempt to train a computer to accurately perform sentiment analysis. The best known fine-grained sentiment analysis algorithms are able to perform with an accuracy of 80.7% on this dataset. But, unfortunately, this paper has not achieved close to this with any of the approaches. In this paper, they have compared the performance with most of the algorithms such as Bag of Words, Random Forests, Support Vector Machines, Logistic Regression, Adaboost Classifier, Gradient Boosting (GB). This paper mainly focused on trying to improve bag of words accuracy by applying linguistic concepts, while incurring limited overhead computation [4].

III. Convolutional Neural Networks

Convolutional Neural Networks (CNNs) is an effective technique in deep learning because they maintain the spatial structure of the data. They have been shown to produce state-of-the-art results in image processing, computer vision, and speech recognition. CNNs have been widely applied in recent years to problems related to NLP and document classification.

CNNs input is a feature map that maps the pixels in an image, or words in a sentence or text, or characters in words. This feature map is scanned by filters in one region at a time in CNNs, assuming filters move across the feature map, or convolve. The way CNNs adjust their filter weights is by backpropagation, which means the network will look at the loss function after the forward pass, and make a backward pass to correct the weights.

A pooling layer which compresses or generalizes over the CNN representations is followed by the CNN layer. It reduces the CNN layer's dimensionality by downsampling the output and taking the maximum value as the feature corresponding to each filter. Usually, the pooling layer is followed by a fully connected feed-forward layer which takes the features from the pooling layer and allows new combinations for further learning or final predictions.

TABLE I
Rotten Tomatoes movie review dataset view.

PhraseId	SentenceId	Phrase	Sentiment
1	1	This movie is very funny.	3
2	1	This movie is very	2
3	1	This movie is	1
4	1	This movie	1
5	1	This	0

IV. Data Description

The Rotten Tomatoes movie review corpus is a collection of movie reviews collected by Pang and Lee in [1]. The corpus consists of approximately 150,000 phrases and all the methods used in this paper are evaluated by training on a random subset of phrases (and their sub-phrases) of approximately 70% of the data set and testing using the remaining 30%. The dataset consists of four feature columns: PhraseId, SentenceId, Phrase and Sentiment which can be seen in Table I.

There are 5 types of sentiments with which reviews have been labelled ranging from 0 to 4 where the numbers represent very negative, negative, neutral, positive and very positive respectively. The distribution of the sentiments can be seen in Fig. 1. From the figure you can see that most of the reviews have been labelled as a 'Neutral' sentiment.

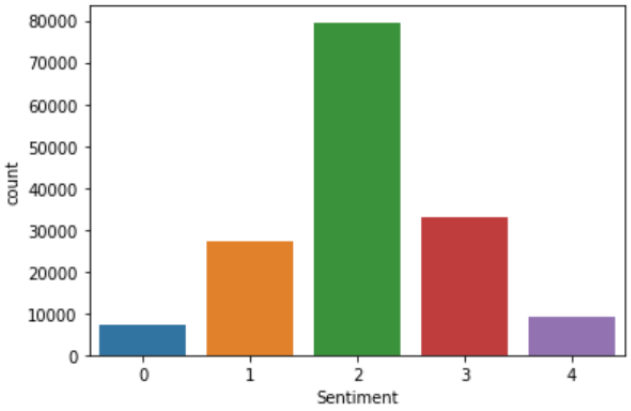

Fig. 1. Distribution of sentiments over the dataset.

As from Fig.1, there is a definite class imbalance in the dataset. To resolve this issue, we have to removed the incomplete sentence from the dataset and considered only first full review sentence. For example, as shown in Table I only the first row will be considered for training the model, as the remaining rows are nothing but the incomplete repeated version of the same review sentence.

V. Model Design

To predict polarity of sentiments, we use a deep neural network model. Our model's configuration, as shown in Table II, is straight forward. The first layer in our model is a randomly initialized word embedding layer which transforms words into a feature map in sentences and preserves the spatial (contextual) information for each word. It is accompanied by a layer of neural network convolution (CNN), which scans the feature map. This CNN layer has 128 filters and a width of 5 which means that each filter is trained in a 5-gram word window to detect a specific pattern.

TABLE II
Deep learning Model Configuration.

Layer	Output Shape	Params
embedding	(None, 100, 100)	1510300
conv1d	(None, 96, 128)	64128
dropout	(None, 96, 128)	0
globalmaxpooling1d	(None, 128)	0
dense1	(None, 64)	8256
dense2	(None, 5)	325

This output is then processed through Dropout layer to get rid of an overfitting issue which generally overfits the model on training data and doesn't perform better while testing this

model. Global maxpooling is applied to each filter's output to take the maximum score of any pattern we search for through the text. The primary role of the pooling layer is to minimize the dimensionality of the CNN representations by downsampling the output and taking the maximum value as the parameters for each filter. Those score are then supplied to 64 size layer of feed-forward (full-connected) and Sigmoid activation to allow further learning. Finally, the layer's output passes through a Softmax layer which predicts the output classes. Table II shows the layer configuration of our model.

In this paper, I tried to compare the performance by training a new embedding and by using pretrained GloVe (The Global Vectors for Word Representation) embedding on the movie review dataset. A word embedding is a learned representation of text where words with the same meaning are expressed in a similar representation. In fact, word embedding is a class of techniques in which individual words in a predefined vector space are represented as real-valued vectors. Each word is mapped to a single vector, and the vector values are learned in a manner that resembles a neural network, and thus the technique is often lumped into the deep learning domain.

An embedding layer is a word embedding that is learned along with a neural network model on a task of processing the natural language, such as language modeling or document classification. It requires cleaning and preparing document text in such a way that each word is one-hot encoded. The vector space size is defined as part of the model, such as dimensions 50, 100, or 300. In this paper, vector space dimension used is 100. Small random numbers initialize the vectors. Using the backpropagation algorithm, the embedding layer is placed on the front end of a neural network and works in a supervised fashion.

GloVe, algorithm is an extension to the word2vec method developed by Pennington, et al. at Stanford for the efficient learning of word vectors. GloVe is an approach for combining the global statistics of matrix factorization techniques such as LSA in word2vec with local context-based learning. Rather than using a window to define a local meaning, GloVe uses statistics across the entire text corpus to construct an explicit word-context or word-co-occurrence matrix. The outcome is a learning model which can usually lead to better embedding of words.

VI. Experiments and Results

Multi-class sentiment analysis for the rotten tomatoes movie reviews dataset has been reported in a number of papers. For example, Nakul Dawra et. al (2015) attempted Bag of Words approach by using different algorithms such as Random Forests, SVM, logistic, Gradient boost ML classifiers and achieved maximum accuracy around 52.44%.

In this paper, by using the pre-trained word embeddings in deep learning model, I achieved around 65% accuracy. Before passing the dataset to the training model, I preprossed it by removing punctuations, stop words, html tags, tokenized the dataset and updated the internal vocabulary based on the list of texts. After tokenization, transformed each text in texts to sequence of integers, then applied padding of 100 to all the squences. While using the GloVe embeddings, we updated the weights feature matrix according to the pre-trained word embeddings for our dataset and passed this matrix to the embedding layer in convolutional model. I have tuned the hyperparameters for the model as shown in Table III

TABLE III
Deep learning Model hyperparameter tuning.

Hyperparameters	Values
loss	category crossentropy
epochs	100
optimizer	adam
batch size	200

TABLE IV
Confusion Matrix without pre-trained word embeddings.

Class	Precision	Recall	F1-score
0	0.51	0.20	0.29
1	0.53	0.31	0.39
2	0.66	0.89	0.76
3	0.56	0.43	0.49
4	0.57	0.23	0.33
Accuracy			0.62
Macro Average	0.57	0.41	0.45

TABLE V
Confusion Matrix with pre-trained GloVe word embeddings.

Class	Precision	Recall	F1-score
0	0.51	0.32	0.39
1	0.52	0.44	0.48
2	0.71	0.82	0.76
3	0.57	0.53	0.55
4	0.56	0.35	0.43
Accuracy			0.65
Macro Average	0.57	0.49	0.52

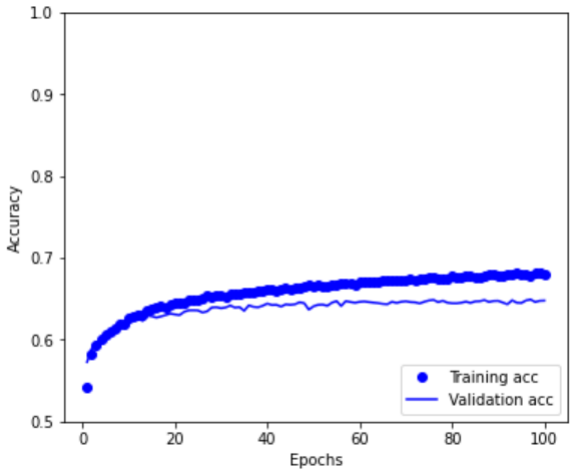

Fig. 2. Training and Validation accuracy over epochs.

After training a model with pretrained GloVe word embeddings, it performed 3% better than the performance I achieved through training a model with new embeddings.

Tables IV and V show the confusion matrix for the model's predictions without training it with word embeddings and with pre-trained GloVe word embeddings respectively.

The pre-trained word embeddings technique works in improving the performance raising both the accuracy (from 62% to 65%) and the macro F-measure scores (from 45% to 52%). The training accuracy curve with GloVe embeddings over the 100 epochs can be seen in Fig. 2. This figure shows that we have removed the overfitting issue with the help of Dropout layer and training curve has a slight edge on accuracy compared to the validation curve. Confusion matrix Table V shows the precision around 57% which implies that an algorithm returned substantially more relevant results than irrelevant ones which is same for training model with new embeddings. Whereas, we got the 8% higher recall in pre-trained word embeddings model than new embeddings model, which indicates that an algorithm with pre-trained GloVe word embeddings returned most of the relevant results, which made a difference with new word embeddings in F1 score by 7%.

VII. Conclusion

In this paper, I presented a framework for classification of multi-class sentiments and found that the model of deep neural networks can outperform traditional methods [4] that rely on language-specific feature design. This design model has performed around 65% with categorical cross-entropy loss around 0.95. In this model, dropout layer has performed major role in resolving overfitting issue which gave better performance on testing dataset by 8%. Experimental analysis shows that 5-gram model performs better than n-gram model with n greater than 5. This analysis aims on comparing the performance in training a deep learning model with new versus using pre-trained word embeddings, where GloVe word embeddings has performed 3% better than new word embeddings on movie reviews dataset.

VIII. Appendix

The dataset used for this project can be downloaded from this link:

https://github.com/cacoderquan/Sentiment-Analysis-on-the-Rotten-Tomatoes-movie-review-dataset/blob/master/train.tsv

The github link for the code can be found at this below link:

https://github.com/komalbarge45/NaturalLanguageProcessing/

The given model files on github are in JSON and h5 format. Json file is an acutal saved keras model which you can load and with the h5 file you can load the weights into the model. Without h5 file, you will not be able to run the loaded json model.

References

[1] Bo Pang , Lillian Lee. Seeing stars: exploiting class relationships for sentiment categorization with respect to rating scales, Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, p.115-124, June 25-30, 2005, Ann Arbor, Michigan.

[2] Socher, Richard, Perelygin, Alex, Wu, Jean Y., Chuang, Jason, Manning, Christopher D., Ng, Andrew Y., and Potts, Christopher. Recursive deep models for semantic com- positionality over a sentiment treebank. In Conference on Empirical Methods in Natural Language Processing , 2013b.

[3] Yang Liu, Jian-Wu Bi, Zhi-Ping Fan. A method for multi-class sentiment classification based on an improved one-vs-one (OVO) strategy and the support vector machine (SVM) algorithm. Expert Systems with Applications, Volume 80, 1 September 2017, Pages 323-339.

[4] Nakul Dawra, Nikola Kovachki, Alex Lew, Walker Mills, Xiang Ni. Sentiment Analysis, April 20, 2015.