**Review Classification using Convolution Neural nets**

**Master’s Project**

Varun Negandhi

vsngndhi@memphis.edu

1. **Abstract:**

In the age of social media, online reviews are critical in decision making. Therefore, predicting if a user is going to review a product favorably or unfavorably is an important task for businesses. The aim of this project is to build a model that predicts the sentiment of user reviews. That is, we want to know whether a review is written with a positive or a negative sentiment. Further, we want to use information from a “neighborhood” of similar reviews to make this prediction more robust. To do this, we develop a model based on Convolution Neural Networks, a model that has traditionally been used for computer vision problems. We evaluate our proposed model on the dataset of IMDB movie reviews and show that it is more accurate than other Machine learning models.

1. **Introduction:**

Online reviews play an extremely important role when one makes decisions regarding almost any product or service [1]. Therefore, businesses are always looking for novel ways to understand the types of reviews a user is likely to write. Specifically, in this project, we consider a binary classification problem where we need to classify a review of a user as that having a positive or a negative sentiment. Sentiment classification is well-known to be an extremely challenging problem due to the complexity of natural language [2]. That is, just like it is difficult for humans to read or understand an unknown language, it is very difficult for the machine learning models to interpret text directly. It is required to translate this unknown language into a form that is easily understood. For typical machine learning models, the language they understand is numerical features. So, these texts have to be converted into a vector of numerical features. However, deriving this vector is not simple. One way to derive vectors is to assign a unique vector to each word and combine the words together using vector operations. In general, this is referred to as vectorization of texts. However, we cannot assign these vectors randomly. We need to assign vectors such that similar vectors represent words with similar meaning. A major advancement in text processing was the development of word embeddings [3]. The idea here is to use a neural network to compute a low-dimensional representation for words. Using this, words with similar meaning have similar embeddings.

Going from word embedding to sentence or document embedding is challenging. Simple approaches include averaging the word vectors, summing the word vectors, etc. However, these do not take the sequence of words in the text into account. That is, for any sequence of the same words the same vector is generated. More recently, Google developed an advanced model that uses sequence information to convert sequences of word vectors to a sentence or a document vector. This model is called as the Universal Sentence Encoder [4].

In this project, the embedding for the text reviews is done using Google’s Universal Sentence Encoder. The main idea of using Universal Sentence Encoder is that we can transform the entire review into a low-dimensional fixed-length vector which is further used for text similarity, text classification, etc. All the review vectors will have same dimensions of length 512. Google’s universal sentence encoder is a pretrained deep learning model freely available on tensorflow hub. This model is trained from various sources by Google and hence it is a very powerful tool for text embedding. It has 2 types of architectures – Transformer model and the Deep Averaging model. Transformer model provides very high accuracy, but it also consumes memory. Deep Averaging model provides high accuracy and also consumes less memory than the previous model discussed. Considering this case, this project uses Deep Averaging model for text embedding [4].

Once the embeddings are obtained, we can use the embedding vector for a review to predict its sentiment using Machine learning models. However, this approach may have errors if the embedding is not perfect. That is, there may be cases where the embeddings of two sentences are similar, but they are of opposite sentiment. To smooth such errors, we develop a new model that is based on the concept of K-nearest neighbors. In K-Nearest Neighbors (K-NN), the idea is to use similarity between data points to make predictions. The most popular recommendation system algorithms such as collaborative filtering that are very popular in predicting user ratings are based on K-Nearest Neighbors. However, the disadvantage of K-NN is that they do not really learn a model that combines the predictions of the neighbors, it is simply based on majority voting (if all neighbors are positive then positive sentiment, else negative). We develop a more advanced deep learning based model that can take advantage of the review neighborhood. In this project, we develop a Convolutional Neural Network [5] based architecture that combines the neighbors using more complex functions called CNN kernels (or filters). CNNs have been extremely successful in image understanding tasks such as object detection, segmentation, etc. because they learn kernels over image neighborhoods. Here, we borrow the same concept for reviews.

We evaluate our architecture on a dataset consisting of IMDB reviews. We show that our proposed approach shows gives better performance than using Machine learning methods directly on the embedding.

1. **Dataset:**

The data is a movie review dataset containing just 2 columns – the SentimentText and the Sentiment. It contains 50000 such reviews [6].

1. **Embedding:**

Embedding can be defined as the mapping of a discrete/categorical variable to a vector of continuous numbers. As explained before, a machine learning model cannot process text data directly. It understands the language of numerical features which we obtain through the process of embeddings. There are many traditional methods of embedding available for example like the one hot encoding. It works well but it does have an issue. The embeddings in one hot encoding do not have a relational meaning. It does not consider the fact that 2 words have similar meaning. For example, the words ‘brilliant’ and ‘excellent’ have similar meanings but their embeddings are random for one hot encoding. This means we cannot identify these as similar words with the current embedding. This drawback is covered by the Universal Sentence Encoder.

Universal Sentence Encoder is a pretrained deep learning model which is already trained on huge dataset from different sources. The embeddings through this process have relational meaning. This embedding helps identify similar words or sentences in the document. The image below shows similar words which have close word embeddings:

A screenshot of a cell phone

Description automatically generated

Fig 1. Similar words having close embeddings

Similarly, the two sentences “I will cook the food today” and “The food is cooked by me” gives a high value of cosine similarity when embedded using Universal Sentence encoder. Whereas, the sentences “I love swimming” and “Please add 2 spoons sugar in my coffee” have low value of cosine similarity. With this advantage of relational meaning between the texts, we move forward with our problem of movie review classification. This is referred to as sentence embeddings. When embedded using the Universal sentence encoder, each review text is a numerical vector with 512 dimensions. The similar reviews are then calculated using the cosine similarity between these embedded numerical vectors.

1. **Baseline models:**

Before getting into the deep learning model, we describe Machine Learning algorithms we use as a baseline in our comparison - K Nearest Neighbors and Decision Trees.

* 1. **K Nearest Neighbor:**

K-Nearest Neighbor(K-NN) algorithm is a very popular unsupervised machine learning algorithm. It is used in classification problems. It is based on a vote-based approach from the nearest K neighbors where K is specified by the user. Depending on the majority of the labels obtained from its neighbors, the given datapoint is classified. For this model execution, the data is first embedded with the Universal sentence encoder. Now all the review text are vectors of size 512 each. This data is then divided into train and test. Next step is applying the nearest neighbor algorithm with 25 nearest neighbors. Here 25 is the value of K in K-Nearest Neighbor algorithm.

**4.2 Decision Tree:**

A decision tree is a supervised machine learning algorithm. It can be considered as a tree type representation of the dataset wherein each node is a class label and the branches are the outcomes of the event. For this experiment, the text data is converted to 512 dimensional embeddings using Universal Sentence Encoder. This is the same approach as explained previously.

1. **Similarity and Neighborhood Concept:**

We use the idea behind K-NNs and collaborative filtering to identify neighborhoods based on similarity. We embed each review into a 512-dimension vector. For each review, the neighborhood set is made up of 25 closest reviews including the review itself. In order to identify the reviews similar to a given review, the similarity measure used here is the cosine similarity.

An example of finding similar sentences using the Universal Sentence Encoder is as:

Great for kids. I love that I can monitor my kids

This is my first tablet. It's very user friendly. Like the parental controls on it. So my kids can use it

This product so far has not disappointed. My children love to use it and I like the ability to monitor control what content they see with ease.

Cosine Similarity

I love this table because I can monitor everything my kids download!!! Watch!!! Use!!!! I can control everything!!! Best investment!!!

Fig 2. Example of identifying similar text reviews by cosine similarity

In the fig 2, we take an example text of a product review and identify the 3 most similar product reviews from the dataset based on cosine similarity.

We encode the dataset as a 3D matrix where the first dimension refers to the number of reviews, second dimension is the size of the neighborhood we consider and third dimension is the size of the embedding. Thus, each review is now encoded as a 2D matrix, where each row corresponds to an embedding that has closest cosine similarity to the review whose sentiment is to be predicted. Since each review is converted to vector with dimension 512 using the Universal Sentence Encoder, the new set formed with the similar reviews will now have 25 rows and 512 columns. This means each review text is now converted into a matrix of dimension (25, 512) which is referred to as a neighborhood for each review. As the original dataset contains 50000 reviews, the shape of our dataset in matrix form is now (50000, 25, 512).

1. **CNN Model:**

The model used in this project is a Convolutional Neural Network where the neighborhood matrix is considered like an image. The input data for this model is the training data having dimensions (Number of training data rows, 25, 512, 1). The model consists of Conv2D layers of first 32 filters and then 2 layers of 64 filters and 80 filters, wherein each convolutional layer is followed by Max-Pool layer followed by 2 Dense layers of 100 neurons and a final output layer containing 2 neurons for the output. The model summary is shown below

A picture containing screenshot

Description automatically generated

Fig 3. Model Summary

This is just to provide the general architecture of the model. The parameters like the number of hidden layers and the number of neurons in each layer are varied during the experiment phase of this project.

1. **Experiment and Results:**

The tools used in these experiment are keras and tensorflow for deep learning model. Inorder to use the Universal Sentence Encoder from Google, the deep learning module tensorflow-hub is utilized in this project. The hardware used is laptop with 2.3 GHz Quad-Core Intel Core i5 processor.

The project is evaluated using the metrics accuracy, precision, recall and F1-Score.

The table below gives the classification report for the K-NN algorithm.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1-Score | Support |
| Negative | 0.83 | 0.80 | 0.82 | 8208 |
| Positive | 0.81 | 0.84 | 0.83 | 8292 |
| Accuracy |  |  | 0.82 | 16500 |
| Macro avg | 0.82 | 0.82 | 0.82 | 16500 |
| Weighted avg | 0.82 | 0.82 | 0.82 | 16500 |

Table 1. K-NN Classification Report

The next table shows the classification report for the decision tree algorithm.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1-Score | Support |
| Negative | 0.74 | 0.75 | 0.74 | 8208 |
| Positive | 0.75 | 0.74 | 0.74 | 8292 |
| Accuracy |  |  | 0.74 | 16500 |
| Macro avg | 0.74 | 0.74 | 0.74 | 16500 |
| Weighted avg | 0.74 | 0.74 | 0.74 | 16500 |

Table 2. Decision Tree Classification Report

These are the results from simple machine learning algorithms. Now, we implement the concept of neighborhood for reviews as explained before.

The next experiment in this project is with the number of convolutions and max pool layer parameters needed in the model. These parameters are the number of filters and the kernel size used in each convolution. So, the experiment starts with using 2 pairs of convolution2D layer and maxpool layer.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| (filters, kernel\_size) | Precision | Recall | F1-Score | Support |
| (32, 2) | 0.83 | 0.83 | 0.83 | 16500 |
| (32, 3) | 0.83 | 0.83 | 0.83 | 16500 |
| (32, 5) | 0.83 | 0.83 | 0.83 | 16500 |
| (32, 10) | 0.82 | 0.82 | 0.82 | 16500 |
| (32, 15) | 0.83 | 0.83 | 0.83 | 16500 |
| (64, 3) | 0.82 | 0.82 | 0.82 | 16500 |
| (64, 5) | 0.82 | 0.82 | 0.82 | 16500 |

Table 3. Filters and kernel size experiment

This project then focused on testing the performance of the model based on the number of neurons in each layer. The table below gives a details about the accuracy and the time taken for each epoch to train.

|  |  |  |
| --- | --- | --- |
| Neurons in each layer | F1-Score | Average time per epoch(sec) |
| 50 | 0.83 | |  | | --- | | 554.2 | |
| 100 | 0.84 | 560.1 |
| 150 | 0.83 | 709.1 |
| 200 | 0.83 | 527.3 |
| 256 | 0.83 | 524.8 |
| 300 | 0.83 | 534.6 |

Table 4. Neurons in each layer experiment

In this experiment, the convolutional layer has 32, 64 and 80 filters in the first 3 layers and kernel size 3. As the filters are increased or the hidden layers are increased the training takes more time.

The next step was to experiment with the batch size and the activation functions used in the project. The batch size used initially was 100 and now it is 1000 just to manage the time constraint. Higher batch sizes were also experimented, but it takes very long time for single epoch. The activation function used is ‘relu’ for the hidden layers. For the output layer, the activation function experimented were sigmoid and ‘softmax’. Also, the epoch value was increased from 10 to 20. The below table gives final values for both activation functions in output layer:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Activation function | Precision | Recall | F1-Score | Support |
| Sigmoid | 0.83 | 0.83 | 0.83 | 16500 |
| Softmax | 0.84 | 0.84 | 0.84 | 16500 |

Table 5. Activation function values

During the training process, the most common problem of model overfitting on the training data occurred. Overfitting is avoided for machine learning models in order to generalize on any unseen data of that type. To handle this issue, Dropout layer values in the model were experimented with. Starting with values from 0.1 to 0.5. To determine whether the model is overfitting or not, 20 percent of the training data was considered validation data set. If the validation loss is much more than training loss, then the model seems to overfit. With the value of dropout layers 0.1, 0.2, 0.25 the training data accuracy of 98% was achieved but the validation accuracy was 82%. There is a big difference in the two loss. Finally, the value of 0.5 reduced the gap between. these loss.

A screenshot of a cell phone

Description automatically generated

Fig4. Loss – Train vs Validation data

**A picture containing pencil

Description automatically generated**

Fig5. Accuracy – Train vs Validation data

These graphs show that the accuracy for training and validation data are approximately same which ensures that the model is not overfitting the training data.

As a result of all these extensive experiments, the final evaluation of the test data was 84% in F1-Score. The final model is trained for 20 epochs with batch size 1000. The detailed classification report of the final result is shown in the table 6. (More results are presented in appendix)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1-Score | Support |
| Negative | 0.83 | 0.85 | 0.84 | 8208 |
| Positive | 0.84 | 0.81 | 0.84 | 8292 |
| Accuracy |  |  | 0.84 | 16500 |
| Macro avg | 0.84 | 0.84 | 0.84 | 16500 |
| Weighted avg | 0.84 | 0.84 | 0.84 | 16500 |

Table 6. Classification report for the model

1. **Future Work:**

This same concept of neighborhood and the model can be extended to a multiclass classification problem. The model explained above was used to predict product ratings for Amazon review data set with five classes[7]. The metrics used for evaluation were the same. the test data in this case give F1 score of 67%. the performance of this model is not as high as expected because it has very few examples of review ratings between 1 and 3. So this unbalanced dataset does not work well [2]. Hence more experimentation is needed in this case.

1. **Conclusion:**

The project focused on classifying a user review based on the similar reviews which are referred to as neighborhood. This system is very useful in user recommendation systems since it has to figure out the likes and dislikes of the user based on previous reviews. The model was also tested with different parameters of neural network like the Convolutional layer and Max pool layer. The number of neurons in the hidden layers were also experimented. It is important for the dataset to be a balanced dataset for target values in order to get a good result through this approach.

1. **Appendix:**

Here are some of the other experimental results obtained for the dataset. These were tested using the same resources as mentioned above.

2 pairs of conv2d and maxpool layer

filters = 32, kernel\_size = 2

precision recall f1-score support

negative 0.82 0.84 0.83 8208

positive 0.84 0.81 0.83 8292

accuracy 0.83 16500

macro avg 0.83 0.83 0.83 16500

weighted avg 0.83 0.83 0.83 16500

filters= 32, kernel\_size = 3

precision recall f1-score support

negative 0.83 0.82 0.82 8208

positive 0.82 0.83 0.83 8292

accuracy 0.83 16500

macro avg 0.83 0.83 0.83 16500

weighted avg 0.83 0.83 0.83 16500

filters=32, kernel\_size = 5

precision recall f1-score support

negative 0.84 0.82 0.83 8208

positive 0.83 0.84 0.84 8292

accuracy 0.83 16500

macro avg 0.83 0.83 0.83 16500

weighted avg 0.83 0.83 0.83 16500

filters=32, kernel\_size = 10

precision recall f1-score support

negative 0.84 0.79 0.82 8208

positive 0.81 0.85 0.83 8292

accuracy 0.82 16500

macro avg 0.82 0.82 0.82 16500

weighted avg 0.82 0.82 0.82 16500

filters=32, kernel\_size = 15

precision recall f1-score support

negative 0.84 0.81 0.82 8208

positive 0.82 0.85 0.83 8292

accuracy 0.83 16500

macro avg 0.83 0.83 0.83 16500

weighted avg 0.83 0.83 0.83 16500

extra dense layer 256 neurons:

filters=32, kernel\_size = 2

precision recall f1-score support

negative 0.84 0.80 0.82 8208

positive 0.81 0.85 0.83 8292

accuracy 0.83 16500

macro avg 0.83 0.83 0.83 16500

weighted avg 0.83 0.83 0.83 16500

filters=32, kernel\_size = 3

precision recall f1-score support

negative 0.83 0.81 0.82 8208

positive 0.82 0.84 0.83 8292

accuracy 0.83 16500

macro avg 0.83 0.83 0.83 16500

weighted avg 0.83 0.83 0.83 16500

1 conv2d, 1 maxpool:

filters = 64, kernel\_size = 3

precision recall f1-score support

negative 0.81 0.84 0.82 8208

positive 0.84 0.80 0.82 8292

accuracy 0.82 16500

macro avg 0.82 0.82 0.82 16500

weighted avg 0.82 0.82 0.82 16500

filters = 64, kernel\_size = 5

precision recall f1-score support

negative 0.81 0.84 0.83 8208

positive 0.84 0.81 0.82 8292

accuracy 0.82 16500

macro avg 0.82 0.82 0.82 16500

weighted avg 0.82 0.82 0.82 16500

1. **References:**

[1] Owen Eagan, Emerson College - *The Influence of Film Critics on Movie Outcomes*, December 2018 in International Journal of Interdisciplinary Research

[2] Edward Collins, Nikolai Rozanov, Bingbing Zhang , *Evolutionary Data Measures: Understanding the Difficulty of Text Classification Tasks*, December 2018 in Proceedings of the 22nd Conference on Computational Natural Language Learning

[3]Tomas Mikolov, [Ilya Sutskever](https://arxiv.org/search/cs?searchtype=author&query=Sutskever%2C+I), [Kai Chen](https://arxiv.org/search/cs?searchtype=author&query=Chen%2C+K), [Greg Corrado](https://arxiv.org/search/cs?searchtype=author&query=Corrado%2C+G), [Jeffrey Dean](https://arxiv.org/search/cs?searchtype=author&query=Dean%2C+J) - *Distributed Representations of Words and Phrases and their Compositionality,* October 2013 in Advances in neural information processing systems.

[4] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Ce ́spedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, Ray Kurzweil - *Universal Sentence Encoder*, March 2018 in *EMNLP demonstration*, Association for Computational Linguistics, Brussels, Belgium

[5] Alex Krizhevsky Ilya Sutskever Geoffrey E. Hinton - *ImageNet Classification with Deep Convolutional Neural Networks,* May 2017 in Communications of the ACM

[6] Dataset download:

<https://www.kaggle.com/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews>

[7] Amazon Product Review dataset for multiclass classification.

<https://www.kaggle.com/datafiniti/consumer-reviews-of-amazon-products>