Predicting the Quality of a Comment, Measured by Public Reaction

R. Dijkstra and **A.K.J.G. de Wit**
Maastricht University / Minderbroedersweg 5-8, Maastricht

Abstract

An online comment can range from serious to humorous and from acclaiming to deriding. All comments are subjective to the content they reply to and the general context. In this article we will describe the programme that predicts public reaction to an online comment, a Reddit comment in particular. Also, we will provide results and analysis of the conducted research. Reddit is an online platform with a lot of user freedom. Its topics range from social media to political discussions and sports commentary. Any user can post or respond to almost anything. The comments of Reddit therefore differ in content and hopefully provide general constraints for a good or bad comment.

1 Introduction

Any piece of text is arguably ambiguous to some extent, which is the reason text is difficult to interpret mechanically. However, with the techniques available in the present day mechanical prediction knows little boundaries. Using a large database of comments we will apply machine learning techniques to train a programme to predict the quality of a comment, measured by the public reaction to a comment. The public reaction can either be positive or negative, represented as a 0 for negative or a 1 for positive. By interpreting this database of existing data, a classifier is made which, when trained, provides a reasonable prediction.

As previously mentioned, comments, like any piece of text, are subjective and even more freely interpretable when intonation and other aspects of speech are lacking. Text comments are either appreciated or disliked by the public and the same comment could be appreciated differently in a different context. Hence, predicting the response to a comment by examining only the text is no trivial task. However, there could be something else that defines a comment that is likely to be appreciated and if not there is at least a probability to be found by interpreting an entry with a classifier trained by the large database.

2 Data

Online platform 'Reddit' calls itself 'the front page of the internet'. As it is almost 14 years old the amount of content that can be found on the platform is colossal. Currently it has 1.2 million 'subreddits' (topics), which are frequented by a large amount of users. Additionally, these users comment on existing posts. Generally, one post contains multiple comments. Due to the vast amount of data and the versatility of the content, which ranges from political debates to humorous content, Reddit is an ideal platform to base our research on. The database used contains two million positively and two million negatively received comments. As well as score, parent comments are available in the database.

Inherently, the task at hand is a simple classification task; learning whether or not a comment will be received positive or not. However, the use of language is connected to context. Consequently, the relation between the parent comment and the entry is very important when deciding whether the response to a comment will be acclaiming or oppositely so.

3 Model

With the use of Tensorflow, a high end programming language for neural networks, a classifier was created that is able to predict the public reaction to a reddit comment with 68% accuracy. We chose to work with a neural network because neural networks are able to handle more complex classification problems. A neural network also tries to

lower the loss instead of only increasing the accuracy, preventing a high variance in the model.

3.1 Implementation

Our dataset of four million entries is split into a fraction of one hundred thousand entries. Each fraction contains an equal amount of positive and negative entries and is mixed randomly. This makes training more efficient. Changing parameters for better results can be done at quicker intervals.

We use the tokenizer from the Keras library to be able to feed the text of both the comment and its parent to a model. This tokenizer first learns from all the text we have and assigns an integer to each word (`tokenizer.fit(text_data)`). This allows us to change every comment into a sequence of integers which is easier to learn to our model.

First we implemented a simple random forest with the most basic hyper paremeters to see wether a model could make sense of the data and respective scores. When the random forest reached 63% accuracy, we began exploring other machine learning options and TensorFlow deep learning looked promising and interesting for our problem. At first, the neural network consisted of six intermediate layers. The first layer is an Embedding layer which turns the positive integers from the tokenizer into dense vectors of fixed size. After that is a 'GlobalAveragePooling1D' layer which averages the input and returns a list of tensors. Then we have three 'Dense' layers which are normal neural network layers with nodes and weights. The last layer is our output layer which has a sigmoid function to give a probability between 0 and 1, allowing the network to predict a 0 (negative comment score) or a 1 (positive comment score).

4 Results

Whilst training, it soon became apparent that this approach resulted in overfitting the classifier to the data (Refer to Appendix A for chronological progression of the achieved accuracy). The accuracy on the training data was 84% and the validation accuracy only 65%. After the necessary research we found several methods for generalizing the neural network. First and foremost, we removed several dense layers. Progressively, the training and validation accuracy converged. Secondly, the amount of neurons in each layer can be altered. With trial and error we found that the ideal numbers of neurons for the layers, were 4 neurons for the Embedding layer and just 2 for the hidden layer.

When the neural network was created it started with training parameters that some tutorials on the internet suggested. The first test ran used a batch size of 512 and ran 40 epochs. The amount of epochs can have an influence on the measure of overfitting. When using too many or too little epochs, the accuracy of the training and validation set diverge. However, we found the ideal number of epochs to be close to our initial number. Fifty epochs seemed to minimise overfitting. With 50 epochs and batch size of 512, and with the new structure of our neural network we were able to achieve an accuracy 73% on our training data and 68% on our validation data. This is the best result we experienced with the variations mentioned above.

5 Analysis

The way accuracy and loss behaved over the course of the experiments is mostly explained within the results. However an accuracy of 68% is not very high and this can be explained due to the lack of contextual information. As previously mentioned, identical comments can be received in a different way when responding to different content. Profanity in reply to an offensive post could trigger a positive response whereas profanity in reply to a post that has been well received could be looked down upon.

Through the project we tried to implement sentiment analysis and profanity checks. We attempted to stack both attributes to the sequence created from the text. However, the neural network has to use the global average pooling layer for the token sequences, because, they are padded with zeros. This is necessary to keep them the same size. When adding the attributes for sentiment and profanity we noticed that these values also got averaged and had little to no impact to the improvement of the model.

Our model only learns based on the text of comments and their parent comments, it doesn't learn anything about context or sentiment which is quite important to understand someones intention with a comment, and therefore, whether it is a likable comment or not. Consequently, it is noteworthy that based on text alone our model can predict, with some degree of certainty, whether a comment

will be received positively or not.

6 Conclusion

With very little information, a complex deep learning model can make noteworthy predictions in seemingly simple classification problems. To the human brain these tasks are often much easier due to the ability to notice patterns and connections. A machine, on the other hand, needs specific instructions to perform any task. Despite the fact that a computer can execute binary operations incredibly fast, some tasks are challenging to script. Interpreting a complex concept such as language is one of those tasks.

Considering the lack of context in our database and the limited power of interpretation of a machine, an accuracy of 68% based on a mere sequence of words is impressive. Be that as it may, there is still room for improvement. With a more extensive database, which also saves the public reaction to the post a comment replies to, a deep neural network could probably tell with more certainty how a comment would be received. Furthermore, if we continue working on this project, we would seek to implement the semantics and profanity properties in a different format. And, consequently, avoid unwanted smoothing steps which, in this particular case, pollute the data.

A Appendix

and Validation accuracy.png

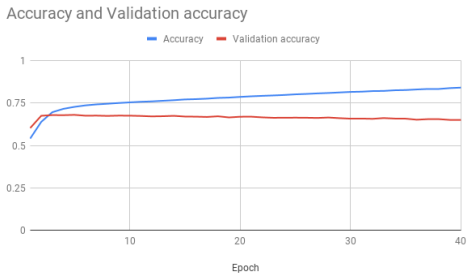

Figure 1: Accuracy and Validation accuracy

and Validation Loss.png

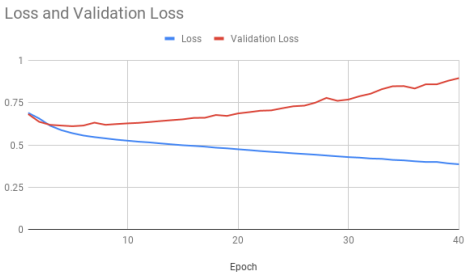

Figure 2: Accuracy and Validation accuracy

and Validation accuracy 1.png

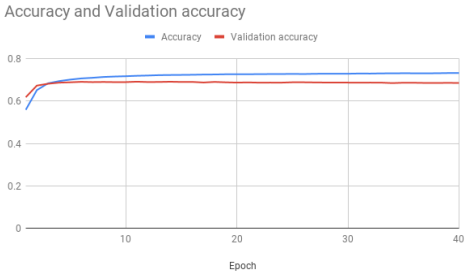

Figure 3: Accuracy and Validation accuracy

and Validation Loss 1.png

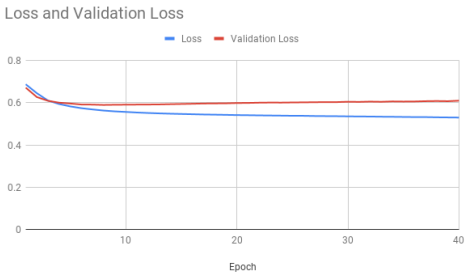

Figure 4: Accuracy and Validation accuracy

and Validation accuracy 2.png

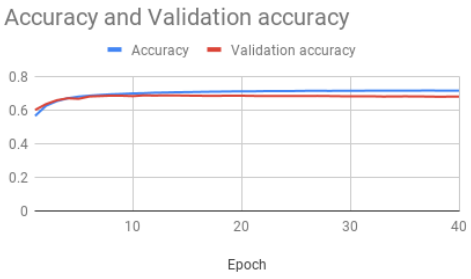

Figure 5: Accuracy and Validation accuracy

and Validation Loss 2.png

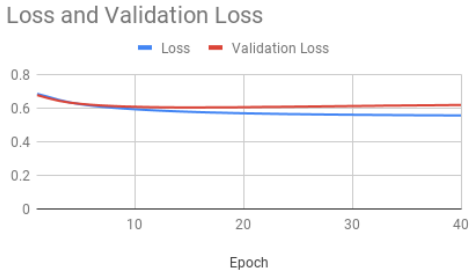

Figure 6: Accuracy and Validation accuracy

and Validation accuracy 3.png

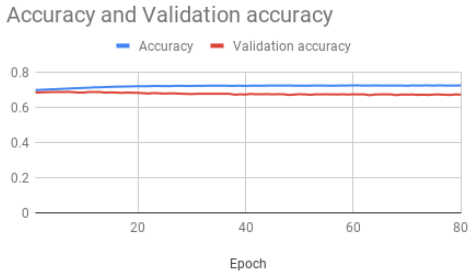

Figure 7: Accuracy and Validation accuracy

and Validation Loss 3.png

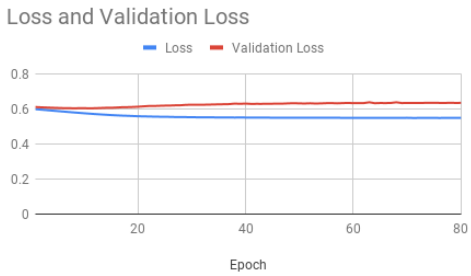

Figure 8: Accuracy and Validation accuracy

and Validation accuracy 4.png

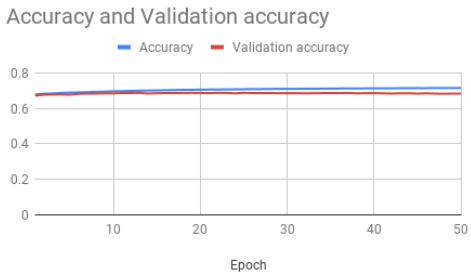

Figure 9: Accuracy and Validation accuracy

and Validation Loss 4.png

Figure 10: Accuracy and Validation accuracy