Sentiment Analysis using BERT

Ahmed Komaira - 6421655
Ben Spooner - 6432380
Robert Bacon - 6416585

Abstract

This paper will investigate the use of Natural Language Processing (NLP) for sentiment analysis as a way to classify text based on their content and context. Specifically, the use of the state-of-the-art Bidirectional Encoder Representations from Transformers (BERT). For this paper we explore a variety of datasets available to train our model against, deciding on the Amazon Reviews dataset. We then investigated comparable models that could be used to validate our BERT model, deciding on ELMo. Finally, we compared the accuracy and precision of our BERT model to a suitably trained ELMo model.

1. Introduction

Sentiment analysis is the computational interpretation of positive, neutral and negative opinions using text data analysis. Understanding people's opinions is essential especially in this day and age where the commercial side of the world has seen a huge increase in e-commerce.

For businesses like Amazon, it can be a huge challenge to analyse and go through the thousands of reviews being posted on its platform every day therefore It is in the businesses best interest to analyse the data provided by the customer.

This is where Machine learning algorithms and data analysis techniques such as BERT can help by automatically classifying and analysing customer feedback. This has also seen an increase in the amount of research and published papers being put into sentiment analysis which has seen an increase over the last 10 years as seen in Figure 1.

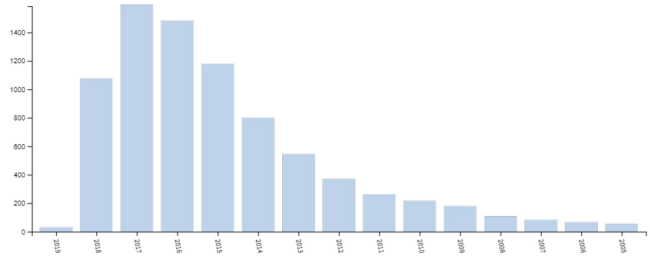

Figure 1. Published Papers on Sentiment Analysis

Creating project objectives will help to critically analyse the success of this project later on:

Objective Number	Objective
1	Critically analyse the current state of NLP sentiment analysis.
2	Find a suitable dataset that relates to the problem.
3	Build a BERT model to predict the sentiment level of sentences.
4	Evaluate the effectiveness of BERT against a similar architecture.

2. Related Work

Sentiment analysis and classification is one of the most popular areas of Natural Language Processing (NLP). A large proportion of sentiment analysis tends to classify based on a positive or negative classification. This may be a result of the lack of datasets available that provide a scale, whereas there are multiple large datasets that provide a suitable database for this style. This section will cover and discuss recent approaches to sentiment analysis.

Sentiment Analysis is not a new area of NLP and therefore there are multiple studies that have been undertaken. A paper in 2019 has explored sentiment analysis using Bidirectional

Encoder Representations from Transformers (BERT) in conjunction with a variety of classifier models to analyse their accuracies [1]. The dataset used in this paper contained 11,855 one sentence move reviews from Rotten Tomatoes provided by Stanford Sentiment Treebank. This dataset has each sentence labelled by at least 3 humans. This is an interesting dataset to use as single sentences will be able to provide a certain level of detail and sentiment, however, multiple sentence reviews which contain a deeper level of sentiment could lead the trained model to have issues. BERT is commonly used for a vast proportion of sentiment analysis. Additionally, BERT has also been used to perform sentiment analysis in 2016 using review data and a set of classifications [2].

Alternatively there is also the use of other word embeddings such as Word2vec as demonstrated in this paper on sentiment analysis in Convolutional Neural Networks (CNN) [4]. Similar to above, this paper made use of the Rotten tomatoes dataset for testing and training. While the use of Word2vec was instead investigated, the difference between BERT and Word2vec being that while BERT is able to generate the different word embedding such that a word is able to have its context in the sentence captured. Word2vec on the other hand generates the word embeddings independent of the context to just output a single vector per word.

CNNs are not the only network used to compute the representation of word vectors to sentiment analysis. There has also been the use of Long Short-Term Memory (LSTM), a special variant of a Recurrent Neural Network (RNN) [3]. Using a layered LSTM allows the data to be used both forward and backwards in the network and thus have a greater level of accuracy than a standard LSTM.

3. Architectures

This section will aim to investigate the current NLP architectures.

3.1. Word2vec & GloVe

These are both examples of context independent word embedders. This means that the position of the word and the surrounding context is not used when calculating the word vectors in a sentence. As such they are able to provide semantic knowledge about known dictionary words. This does, however, have disadvantages due to the fact that words used in different contexts result in the same embedding being used for all instances. For example “That bird is a crane” and “They had to use a crane to lift the object” would both produce the same embedding for ‘crane’.

3.2. ELMo

ELMo is similar to BERT in that they are both context inclusive word embedders. However, ELMo is considered the older way to do this and BERT state-of-the-art. ELMo makes use of a concatenation right-to-left and left-to right LSTMs which means one layer reads in one direction and another layer reads in the other. Additionally, ELMo makes use of a character based input, providing vectors for each character that are usually combined through a model.

3.3. Bidirectional Encoder Representations from Transformers (BERT)

BERT [7] is a natural language processing (NLP) technique developed by Google which pre-trains a large model using a huge corpus, which can be used and adapted to match specific needs by fine-tuning the model with specific data. BERT is a Masked Language Model (MLM) instead of left-to-right or right-to-left unidirectional language models. It randomly selects 15% of all tokens and places a [MASK] token in its place which ‘enables the representation to fuse the left and the right context’ [2]. BERT is bidirectional, it considers the entire sequence instead of only the tokens before or after which is why it is so accurate in terms of context recognition.

Pre-training vs Fine-tuning

BERT’s framework is split into 2 similar parts, pre-training and fine-tuning shown in Figure 2. The pre-trained model is trained on a massive unlabelled dataset, BERT publishers stating ‘we use the BooksCorpus (800M words) [9] and English Wikipedia (2,500M words)’ [2]. Users can now download these models, freeze all layers previous to the output layer (to keep the pre-trained weights the same) and run a small amount of training for the users specific dataset. It is recommended that about 2-4 epochs are

run when fine-tuning as most of the bottom layers have already been perfectly weighted from the previous pre-trained step.

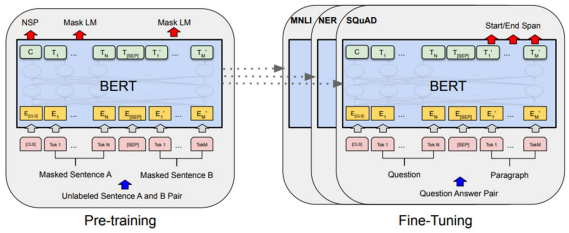

Figure 2

There are many different pre-trained models that are available, varying from multilingual support and cased/uncased. There are 2 main models that the creators recommend to use: BERTBASE (L=12, H=768, A=12, Total Parameters=110M) and BERTLARGE (L=24, H=1024, A=16, Total Parameters=340M) where 'we denote the number of layers (i.e., Transformer blocks) as L, the hidden size as H, and the number of self-attention heads as A' [BERT]. To load in a custom dataset to these models, a tokenizer must first be used. The BERT Tokenizer is created with the WordPiece model, which contains around 30,000 common english words, sub words and characters. It tries to match each token from a sentence with its dictionary, if none are found then it tries to match sub-words and if none are found it tries to match individual characters. The output vector, even if a word is not matched to any valid vocabulary, can be attained whilst keeping contextual meaning. Added to the beginning of each sequence is a classification token '[CLS]' and '[SEP]' to separate or finish a sequence.

Max sequence length is an important parameter for BERT. Capping at 512, the lower the max length the quicker the network can be trained so it is important to set this as low as possible while not affecting the dataset since sequences over the maximum value are truncated.

4. Dataset

The datasets available for us to use were the Amazon Review and IMDB Review dataset.

IMDB: 50,000 Movies reviews which was split evenly with 25,000 reviews for training and 25,000 reviews for testing the classifier. The range of the ratings are on a scale of 1 to 10.
Amazon: 130+ Million reviews and comes in multiple formats such as TSV and JSON. The dataset also provides reviews in multiple languages which will be helpful should we want to expand into other languages.

We had chosen to use the Amazon Reviews dataset as it was more reliable as each review was verified by Amazon for validity as well as the ability to expand into other languages.

The dataset chosen is derived from the consumer reviews publicly shared by Amazon to encourage further research to be done on understanding customer product experiences. The dataset contains 130 million customer reviews which contain metadata such as star_rating, review_body, the date the review was given and other related fields presented in a csv type format which will be used to train and test the model. An example review has been shown in Figure 3.

Figure 3

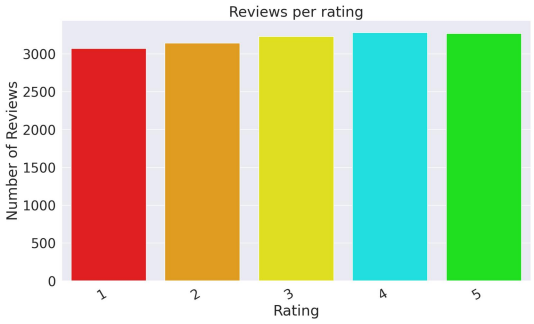

Figure 4.

The dataset contains 5 classes which are ratings from 1 - 5. Figure 4 shows the classes are not imbalanced which helps to deliver accurate results. In section 5, we will be showing how the data will be converted and tokenized in order for the data to be processed.

5. Method

For the model, the lowercase BERT-Base was chosen due to case sensitivity not impacting the review context and the large model being excessive for this type of text sentiment.

Loading the Dataset

After loading the .csv dataset, 20,000 reviews were taken from it and split into training and testing data using a 80% / 20% split. Using the python package 'Seaborn' we can plot the review data by star rating to make sure the training data is evenly split.

Pre-processing Data

Both training and testing data need to be pre-processed to turn review sentences into the required format for BERT to use.

Firstly each sentence needs to be tokenized. This step is done by normalizing text (convert whitespace characters to spaces, convert to lowercase and remove accent markers) and punctuation splitting which adds whitespace around punctuation. To finish the tokenization, WordPiece tokenization is run which converts longer versions of words to simpler subwords eg. playing -> play + ##ing. This is done so similar versions of words can be treated the same eg. run, runner, running. After tokenization, the next step is to apply [CLS] and [SEP] to the start and end of the tokenized sentence, respectively. Lastly we need to convert tokens to ids and pad data with 0s.

BERT Model

In figure 5 is a summary of the finished model.

Layer (type)	Output Shape	Param #
input_ids (InputLayer)	[(None, 196)]	0
bert (BertModelLayer)	(None, 196, 768)	108890112
lambda (Lambda)	(None, 768)	0
dropout (Dropout)	(None, 768)	0
dense (Dense)	(None, 768)	590592
dropout_1 (Dropout)	(None, 768)	0
dense_1 (Dense)	(None, 5)	3845

Total params: 109,484,549
Trainable params: 109,484,549
Non-trainable params: 0

Figure 5.

The first layer takes the pre-processed ids as an input (notice the shape is set to the max sequence length) and feeds them into the BERT layer. After this, to reduce the dimensionality of the layer, a Lambda layer is used to flatten the middle dimension (max sequence length) whilst ignoring the other dimensions. The resulting tensor is fed through dropout and dense layers finally ending with a dense layer of shape that represents the classes the model can predict, in this case 5. The last layer has a softmax activation function which should be used when doing classification and outputs probabilities for each class that all sum up to 1. Dropout layers are used in this model to prevent overfitting of data.

Results

BERT has a trade-off between max sequence length and batch size. A lower sequence length requires less memory so larger batch sizes can be used. After a few training sessions while tweaking parameters, it was found that using a max sequence length of 196 was suitable for the review data alongside a batch size of 16. In the BERT Github page [7], it states that, when using BERT-Base, a sequence length of 256 caps the batch size at 16. The same batch size and lower sequence size ensures that the model will not run out of memory when training. Figures 6 and 7 plot the results when 2000 review samples were used with a batch size of 12. It shows that the model is overfitting data, training accuracy increases whilst validation accuracy decreases. The model is learning too well from the training data whilst actually getting worse at predicting data it has never seen before.

Figure 6.

Figure 7.

After training successfully over 4 epochs, the results can be plotted. For both graphs in figures 8 and 9, overfitting does not occur as the validation accuracy increases alongside training accuracy.

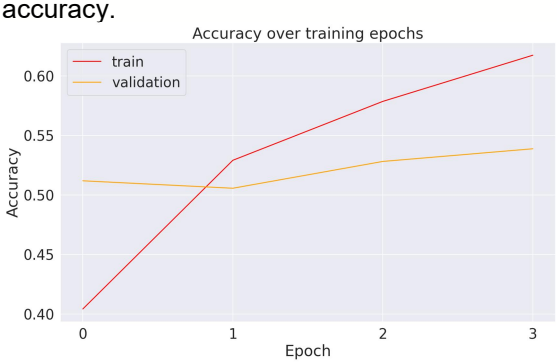

Figure 8.

Figure 9.

A confusion matrix, as shown in figure 10, can now plot the model's predictions using the test set. It shows the true class labels on the y axis and the models predicted values on the x axis. There is a trend with the graph, most of the time predictions are made correctly with the rest falling either side of the true positive. Only a handful of the testing set items have incorrectly predicted the correct result shown with dark blue in the matrix.

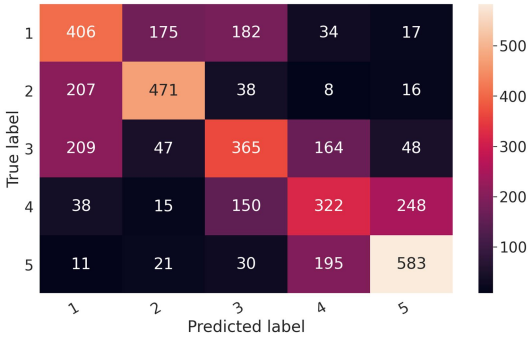

Figure 10.

Testing the results on real data is equally important to analysing model metrics. After creating some test review sentences, we could then perform pre-processing to convert it to the required input shape. The output layer on the model has a softmax activation function which outputs the class probabilities. To ensure the model gives a single predicted rating value, armax is used when predicting to select the label that has the highest classification probability. The results are shown in figure 11.

Sentence	Predicted Rating
"This product is rubbish, would not recommend."	★
"Amazing! Loved this so much! Very Happy."	★★★★★
"This product was average."	★★
“this product was ok, I enjoyed some aspects however, other parts were not up to standard."	
“I had only problems.”	

Figure 11.

ELMo

To evaluate BERT, alongside using metrics for loss/accuracy and predicting against a test set, we must compare it to a similar architecture like Word2Vec, ELMo and GloVe which were researched earlier in this paper. ELMo was chosen because it uses LSTMs instead of Transformers whilst being similar to BERT e.g. shallowly bidirectional. Comparing the results of both models will indicate if the chosen model was correct to use for the task and the dataset.

To keep things fair, the same batch size and input data was used. The number of epochs however were unique to the model, epochs of 5 and 6 lead to overfitting data so the best candidates were 3 and 4 as any less than this would have resulted in underfitting. When training ELMo, the data was consistently being overfit regardless of hyperparameters. Even

adding a dropout layer between the last dense layers had no impact. Changing the optimizer from RMSprop to stochastic gradient descent (recommended on the keras-team github [10]) helped fix the issue of predicting a single class for all true labels. The model architecture shown in figure 12 was used to compare against BERT.

Figure 12.

After training, the metrics could be plotted and predictions could be done against the same test sentences like the previous BERT model. As seen in figures 13 and 14.

Figure 13.

Figure 14.

The confusion matrix in figure 15 shows that, like BERT, it predicts the same or near to the true label. Prediction seems almost random which could be due to context of phrases not being learned well enough.

Figure 15.

Now we need to test the new model on the test sentences again as shown in Figure 16.

Sentence	Predicted Rating
"This product is rubbish, would not recommend."	
"Amazing! Loved this so much! Very Happy."	
"This product was average."	
“this product was ok, I enjoyed some aspects however, other parts were not up to standard."	
“I had only problems.”	

Figure 16.

6. Evaluation

After comparing the results from the previous step, it is quite clear that BERT outperforms ELMo for sentiment analysis on review text. Comparing the confusion matrices, it is clear that while both models are accurate and can predict roughly the true sentiment BERT’s predicted values are much more precise. ELMo’s matrix shows a lot of predictions on neighbouring labels showing it is going in the right direction.

When analysing the predicted sentences, ELMo has learned quite well, however some sentences used were completely misclassified. “This product was average” should be around 3 stars

whilst “I had only problems” should be 1. BERT on the other hand predicted well, apart from “I had only problems” which was rated 3 stars, 1 less than ELMo.

The final models accuracy is shown below which can confirm our earlier analysis:
BERT: 60.04%
ELMO: 42.90%

Although many other sentiment analysis projects using BERT achieve a higher accuracy (>80%), these are using only positive or negative reviews. This project has 5 different classes all of which could overlap each other.

Challenges faced

Preprocessing data was a difficult step in this project. This is due to each model having a unique set of inputs requiring a certain format. Following guides listed under official model resources helped to understand the step by step process of turning a review sentence into the correct tokenized or embedded format.

Objectives

Objective 1 was to critically analyse the current state of sentiment analysis within NLP. As investigated during Section 2: Related Work, there have been some studies and papers on sentiment analysis. We can see that the current state of sentiment analysis is through making use of BERT as a state-of-the-art model in comparison to older models such as ELMo.
Objective 2 was to find a suitable database that relates to the problem. During Section 4: Dataset, we identified two datasets that could be used for training and testing, Amazon and IMDB. We decided to use the Amazon reviews dataset as this provided us with a more reliable set of data as each review is verified by Amazon for validity and content ensuring our training would be the best possible. The data was also available in multiple different languages which gives us the opportunity to expand.
Objective 3 was to build a BERT model to predict the sentiment level of sentences. As shown in Section 5: Method, the BERT model was implemented. Shown in this section are some sample sentences provided to our model as well as the predicted rating. As context is inclusive within BERT the first sentence “This product is rubbish, would not recommend.” becomes a single star review. Without context this would be higher as ‘recommend’ would be taken positively.
Objective 4 was to evaluate the effectiveness of this BERT model against a similar architecture. The similar architecture was determined during Section 3: Models, the best comparison was found to be ELMo as it is also a context inclusive embedder. In contrast to this, Word2vec and GloVe were also investigated but as they were both context independent embedders were deemed not comparable. Once this was decided, we implemented a version of the ELMo model using the same dataset as our BERT model, this is shown in Section 5: Method – ELMo. Finally, as shown during this section, the accuracy of our BERT model when compared to our ELMo model was roughly 20% better showing the clear effectiveness of BERT.

7. Conclusion

This project was a success due to the effectiveness of the sentiment analysis model, the evaluation of it and the objectives that were met. Even though Word2Vec and GloVe were not tested, we can estimate that, like ELMo, BERT will outperform them due to a superior understanding of context.

8. Individual Contributions

Ahmed Komaira: Dataset investigation, performed data pre-processing and assisted with the analysis of the models using confusion matrices and graphs.
Ben Spooner: Carried out BERT research, performed data pre-processing and worked on implementing both BERT and ELMo. Evaluation of each model was also carried out
Robert Bacon: Investigated related sentiment analysis work. Carried out ELMo research and worked on implementing ELMo. Helped analyse finished models.

References

[1] Wanliang Tan, Xinyu Wang, Xinyu Xu. (2018) Sentiment Analysis for Amazon Reviews. http://cs229.stanford.edu/proj2018/report/122.pdf

[2] Munikar, Manish & Shakya, Sushil & Shrestha, Aakash. (2019). Fine-grained Sentiment Classification using BERT. 10.1109/AITB48515.2019.8947435.

[3] Fang, X., Zhan, J. Sentiment analysis using

product review data. Journal of Big Data 2, 5 (2015). https://doi.org/10.1186/s40537-015-0015-2

[4] X. Ouyang, P. Zhou, C. H. Li and L. Liu, "Sentiment Analysis Using Convolutional Neural Network," *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, Liverpool, 2015, pp. 2359-2364, doi: 10.1109/CIT/IUCC/DASC/PICOM.2015.349.

[5] Pal, Subarno & Ghosh, Soumadip & Nag, Amitava. (2018). Sentiment Analysis in the Light of LSTM Recurrent Neural Networks. International Journal of Synthetic Emotions. 9. 33-39. 10.4018/IJSE.2018010103.

[6] Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Well-read students learn better:The impact of student initialization on knowledge distillation.arXiv preprint arXiv:1908.08962,2019.

[7] Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *ArXiv, abs/1810.04805*.

[8] Google Research. (2020) TensorFlow code and pre-trained models for BERT. https://github.com/google-research/bert

[9] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In Proceedings of the IEEE international conference on computer vision, pages 19–27.

[10] Ashia C. Wilson, Rebecca Roelofs, Mitchell Stern, Nathan Srebro, Benjamin Recht. (2017) The Marginal Value of Adaptive Gradient Methods in Machine Learning. https://arxiv.org/abs/1705.08292