**Sentiment Analysis on Yelp Reviews using BERT**

**Abstract**

With the advancement of technology, we now check the internet for reviews of businesses before going to a restaurant, purchasing a product, or utilizing any services from a service provider. As a result, there are numerous platforms where customers may write reviews, and other customers can trust them sufficiently to make decisions based on them. Sentiment analysis is commonly used on textual data to help businesses analyze brand and product sentiment in customer feedback and better understand client expectations. The reviews are beneficial in a variety of ways, including improving operations, decision-making, and customer satisfaction. Our goal is to predict whether the reviews will be positive or negative based on the user's comments using BERT.

**Introduction**

Sentiment analysis is the process of determining the emotional tone behind a series of words, used to gain an understanding of the attitudes, opinions and emotions expressed within an online mention. Sentiment analysis on social media acts as a way of monitoring the people as it allows us to gain an overview of the wider public opinion behind certain topics. The ability to extract insights from social data is a practice that is being widely adopted by organizations across the world. In fact, shifts in sentiment on social media have been shown to correlate with shifts in the stock market! Due to the importance of sentiment analysis on social media we decided to perform sentiment analysis on one of the popular social media websites that we have today -Yelp. For our project we will be using a subset of the Yelp Reviews Dataset which originally consists of ~8m reviews. We will be implementing sentiment analysis using BERT. We will also be experimenting with different hyperparameters and report the best result. We will be using the evaluation metrics- accuracy, f1-score, precision, recall and computational efficiency(time taken to run) to evaluate the performances of our model. When evaluating the sentiment (positive, negative) of a given text document, research shows that human analysts tend to agree around 80-85% of the time. This is the baseline we try to meet or beat when we’re training our sentiment scoring system [1].

**Dataset - Yelp Reviews**

The review dataset contains review\_id in string format which represents unique id of the review posted, business\_id (String) contains unique id of the business for which the review is posted, text ia a string column containing reviews posted by user for the respective business. Star is an int column containing the stars given by the user along with the review. Date is a string, date format YYYY-MM-DD column containing the date of the post. Useful is an integer column of the number of useful votes received. Funny is an integer column of the number of funny votes received. Cool is an integer column of the number of cool votes received.

The Review dataset contains the review\_id, business\_id, review and rating given by the user.It consists of 8,635,403 reviews.

**Exploratory Data Analysis**

The business ratings are in the scale 1 to 5 where 1 is the lowest rating and 5 is the highest rating. There are no missing values in the ratings dataset. 1 and 5 are the most popular ratings. We are considering 3 to 5 ratings as positive sentiment (denoted by 1) and 1 & 2 as negative sentiment(denoted by 0) for analysis.

Due to our computational limitations, we decided to randomly sample 20,000 rows from our dataset. As you can see in the graph below, the two classes were balanced with respect to the positive and negative reviews.

Figure 1 (a): Distribution of Ratings across the dataset. Figure 1 (b): Distribution of sentiments across training dataset after converting 1,2 rating to negative (0) sentiment & 3,4,5 to positive (1) sentiment and then sampling 20,000 reviews.

Since sentiment in text is often affected by geographical location, we wanted to explore the regions this text belonged to as well in case it would help us in further analysis. The most number of reviewers are from Austin, Portland, Atlanta, Boston and Orlando. We can see that top users are from ​​Austin but Portland and Boston are in the Top 5 list.

Figure 2: Distribution of reviewers on the basis of their geographical location (State)

We observe that the length of token i.e., number of words in a review reaches upto 1200. Due to computational constraint of running BERT on google colab we removed the reviews containing word count more than 100.

The first plot represents the token length i.e., Word counts in reviews before removing the longer reviews and the second plot is after removing the reviews with review word counts more than 100.

Figure 3(a): Distribution of number of words in reviews. Figure 3(b): Distribution of number of words in reviews after removing reviews with greater than 100 word count.

Since the dataset was huge and inorder to understand the Yelp data to determine the pre-processing techniques we need to use we did the word analysis

We constructed a word cloud of reviews from the reviews dataset. Larger font size represents higher frequency. We are using the word cloud module built by Andreas Mueller. Mueller uses scikit-learn's CountVectorizer and extracts the 200 most frequent words, after filtering for 'stopwords' (common English words). More information on his approach is available in his blog post.

Figure 4: Word cloud generated on the basis of frequency of term/word used in reviews.

After removing the stop words when we construct the most frequently used words based in all reviews. We can see that food, place, great, good are most frequent words

Figure 5: Distribution of most frequent word in the description of reviews

We are plotting the 'most important' words in the Review dataset. 'Most important' means a term that occurs frequently in an individual review but is not used too frequently across the entire corpus of review descriptions. This uses tf-idf (Term Frequency - Inverse Document Frequency) and TfidfVectorizer.

Figure 6: Distribution of most important terms in reviews based on frequency of occurance in an individual review

Similarly, If we observe the top categories of business with top rating counts, we can see that top businesses are Mexican restaurants, Pizza restaurants, Chinese Restaurants, Coffee & Tea places and Salons. Which shows that most of the businesses rated at Yelp are food places. We can see redundancy in names because of changes in order of words. 15 out of 20 Top business rated are food businesses.

Figure 7: Top categories of business based on ratings given by users

**Data Pre-Processing**

For pre-processing of dataset we perform following steps :

1. We dropped the following columns review\_id, user\_id, business\_id, date, useful, funny and cool. Since these columns will not be used for sentiment analysis and due to computational restraints we had to remove some data to reduce resource consumption.

2. We checked for NA/Null values in all columns. The Review dataset didn’t contain any NA values in any column.

3. Next, we check for duplicate rows. There are no duplicates in any dataset

4. Add a new column ‘Sentiment’ on the basis of ratings given by the user, if rating 3 or more then we consider it as positive review(1) and negative review(0) otherwise.

5. Sentiment analysis is very sensitive towards words that are feeded into the model hence removing the stop words might change the meaning of the sentences. Example removing “not” from a review containing “not nice”, will provide wrong information to the model. Hence, we are not removing the stop words from the corpus.

6. Removal of non-alphanumeric characters using string functions along with some special characters like "&amp;amp;", “\n”

7. We checked for spelling mistakes using SpellChecker module and applied SpellChecker.correction function on the review column which checks and corrects all the spelling mistakes in the column. For example someone wrote “gooooood food” which will be treated as new token but after spell correction, it will just be treated as “good” token.

8. We are using the Tokenizer from keras module to tokenize the review column and generate tokens of words and save it into a new column named “cleaned\_tokens\_uncased”. We get the token counts per review by counting the length of tokens generated.

9. Since the BERT model works with a fixed length of input, we figured that a length of 100 is appropriate for the computational limitation, we have of running BERT on Google Collab. Hence, we removed the reviews which have token count more than 100.

10. Converting text column into lower case and saving it into a new column named “cleaned\_text\_cased”. When we tested how BERT will perform with all lower-case reviews as compared to original reviews, the results did not give us a good accuracy so this column wasn’t used further.

11. Renaming “review” column to “text” column and preprocessed review column to “cleaned\_text\_uncased”

**Methodology**

# **The Human Baseline**

We wanted to understand what kind of accuracy we should be aiming for. From reading up from online sources[1] that when evaluating the sentiment (positive, negative) of a given text document, research shows that human analysts tend to agree around 80-85% of the time. This is the baseline we will try to meet or beat when we’re training our models. Note that this means that you’ll always find some text documents that even two humans can’t agree on, even with their wealth of experience and knowledge. But when we’re running automated sentiment analysis through natural language processing, we want to be certain that the results are reliable. So, we evaluate and look at how accurate we can get and we aim at getting at least 80-85% of accuracy.

# **BERT**

## **About BERT**

BERT[2] stands for Bidirectional Encoder Representations from Transformers.

Bidirectional means, to understand the text you’re looking at you'll have to look back (at the previous words) and forward (at the next words). ‘Attention Is All You Need’ paper[3] presented the Transformer model. The Transformer reads entire sequences of tokens at once. In a sense, the model is non-directional, while LSTMs read sequentially (left-to-right or right-to-left). The attention mechanism allows for learning contextual relations between words (e.g. his in a sentence refers to Jim). BERT also uses (Pre-trained) contextualized word embeddings - The ELMO paper[4] introduced a way to encode words based on their meaning/context for e.g Nails has multiple meanings - fingernails and metal nails. BERT was trained by masking 15% of the tokens with the goal to guess them. An additional objective it had was to predict the next sentence. The training corpus for BERT consisted of two entries: Toronto Book Corpus (800M words) and English Wikipedia (2,500M words). While the original Transformer has an encoder (for reading the input) and a decoder (that makes the prediction), BERT uses only the decoder.BERT is simply a pre-trained stack of Transformer Encoders. We have two versions of BERT - with 12 encoders (BERT base) and 24 encoders (BERT Large). We are working with the BERT base model.Additionally we can use a cased and uncased version of BERT and tokenizer. We’ve experimented with both & reported the results.The BERT paper was released along with the source code and pre-trained models. The whole reason we picked BERT is because you can do Transfer Learning with BERT for many NLP tasks such as ours and we can train with small amounts of data and achieve great performance!

## **Data Preparation for BERT**

We checked for NA Values in every column there were no NA values. Then we removed some frequently used signs such as ‘\n’ from the review column to get more clarity of words else the algorithm would have considered it one word.Then removed special characters from the review.Check spellings and abbreviations in the review column. We corrected the spelling using the in-build library.

For the BERT cased version we will be using data which is lower-cased while for the BERT uncased version we will be using data which is NOT lower-cased.Other than this all the methodologies are the same for both the models.

We had to perform additional data preparation before feeding the data into BERT for optimum results. We added [SEP] as a marker for ending a sentence. Also we must add [CLS] to the start of each sentence, so BERT knows we’re doing classification. BERT also uses a special token for padding i.e [PAD]'. Additionally BERT only understands tokens that were in the training set. Everything else should be encoded using the [UNK] (unknown) token. All of these data preparation methods are applied using the encode\_plus() method. BERT works with fixed-length sequences. From Fig 2 we know that almost all of our tokens have a sequence length less than 50. Just to be safe we’ve selected a sequence length of 100.

## **Classifier Design**

Our classifier SentimentClassifier() delegates most of the heavy lifting to the BertModel. We use a dropout layer for some regularization and a fully-connected layer for our output. We return the raw output of the last layer and use that for the cross-entropy loss function(requirement of Pytorch CE Loss function). The rest should work like any other PyTorch model. To get the predicted probabilities from our trained model, we’ll apply the softmax function to the outputs.

## **Training**

To reproduce the training procedure from the BERT paper, we have used the AdamW [5] optimizer provided by Hugging Face. It corrects weight decay, so it’s similar to the original paper. We use a linear scheduler with no warm up steps. The BERT authors have some recommendations for fine-tuning i.e

· Batch size: 16, 32

· Learning rate (Adam): 5e-5, 3e-5, 2e-5

· Number of epochs: 2, 3, 4

Figure 8: Training and validation accuracy obtained vs number of epochs

We ran our model for 4 epochs and the validation accuracy was the highest at epoch 2.

**Results & Conclusion**

Our results have been consolidated in the following table. We ran all the learning rates as recommended by the BERT authors to get the results for 3 models.

As you can see all the results are comparable. We picked the third model since the test accuracy was the highest. The hyperparameters used for the best model i.e BERT-3 were :

· Batch size = 16

· Learning Rate = 5e-5

· Epochs = 4

BERT is known to very quickly overfit to the data. However, as we can see we attained an extremely high level of test accuracy of 95.1% without really doing any further fine-tuning. It is also interesting to note that BERT was able to obtain 94-95% test accuracy with just 18,000 records of training. This is impressive because machine learning models are often known to require huge amounts of training data for a high level of accuracy. However, BERT being pre-trained on a humongous corpus works wonders with very little training data. This also means that BERT would train much more quickly - as we can see all the models ran in around ~13 minutes for 4 epochs. That’s definitely a very reasonable time. Another very important point to note is that sentiment analysis is quite context heavy. Algorithms for sentiment analysis that use the bag of word representation often lose out on the context. BERT on the other hand is revolutionary because it is able to process context-heavy text with high levels of accuracy due to it’s bidirectional training. Following our table is the classification report as well to further scrutinize the predictions from our model.

|  | BERT-1 | BERT-2 | BERT-3 |
| --- | --- | --- | --- |
| Precision | 0.957 | 0.950 | 0.958 |
| Recall | 0.936 | 0.942 | 0.940 |
| F1 | 0.947 | 0.946 | 0.949 |
| Test accuracy | 0.949 | 0.948 | 0.951 |
| Computational Efficiency | 13min 19s | 13min 10s | 13min 8s |

\*Model 1 has learning rate 2e-5,Model 2 has learning rate 3e-5, Model 3 has learning rate 5e-5.

Figure 9: The confusion matrix of the best BERT model obtained.

**Future Work**

In the future we would like to experiment with the hyperparameters further. In this project we only experimented with the number of epochs and learning rate. We would also like to experiment with batch size. Additionally, we would also want to see how the model performs with a number of epochs more than that suggested by BERT authors.

Also, we want to try the BERT uncased just to verify that it wouldn’t perform as well as the BERT cased model that we picked intuitively.

We would also experiment how BERT would perform with an extremely less amount of data for e.g 1000 records. Reducing the data also means lesser training time and if we can achieve high levels of accuracy with just that amount of data (even if it means some hyperparameter fine tuning) it would be quite a revelation.

Citations

[1]<https://www.lexalytics.com/lexablog/sentiment-accuracy-baseline-testing>

[2]<https://arxiv.org/abs/1810.04805>

[3]<https://arxiv.org/abs/1706.03762>

[4]<https://arxiv.org/abs/1802.05365v2>

[5]<https://huggingface.co/docs/transformers/main_classes/optimizer_schedules#adamw>

[6]<https://www.kaggle.com/sudhirnl7/basic-exploration-of-business-review-at-yelp-com>

[7]<https://github.com/ds5110/stinky/blob/main/text_analysis.ipynb>