Sentiment Analysis: Predicting Yelp Scores with BERT

Lichen Zhang (lichenz)
Carnegie Mellon University
Pittsburgh, PA 15213
lichenz@andrew.cmu.edu

Jinyao Zhang (jinyaoz)
Carnegie Mellon University
Pittsburgh, PA 15213
jinyaoz@andrew.cmu.edu

Pengtao Ni (pni)
Carnegie Mellon University
Pittsburgh, PA 15213
pni@andrew.cmu.edu

Abstract

Understanding the sentiment behind a text is a challenging task for machines. Given the user review data from Yelp, we explore multiple ways to build models that predict the score of the reviews and understand the factors that make a review positive or negative. By using the state-of-art natural language understanding model **BERT**, we are able to achieve a testing accuracy of 62% which is 50% higher than a baseline L_1 regression approach, which is only around 40% accuracy.

1 Introduction

Natural language processing is one of the most important applications of machine learning. There has been multiple pre-trained language models that are able to perform different natural language understanding text, including the word-level interpretation models such as Word2vec (Tomas Mikolov, et al. 2013)[1] and Glove (Jeffrey Pennington, et al. 2014)[2] and the sentence-level recognition models such as BERT, which stands for Bidirectional Encoder Representations from Transformers (Jacob Devlin, et al. 2018)[3].

Understanding the sentiment behind a text has been a particularly challenging task, since there are no explicit metrics that determine or classify human sentiment. However, it is crucial to understand the meaning conveyed by the text, which requires an understanding of not only words and tokens but also the underlying structures and tones of the text.

Our goal essentially consists of two tasks: the first one is to train a model that predicts the ratings (ranging for 1 to 5) of individual yelp reviews based on the numerical features and the review texts; and the second task is to identify the strongest indicators among all the features that leads to either positive or negative ratings. Our training and testing data both include the review text and classified features such as date, usefulness score (ranging from 1 to 5), restaurant name, restaurant location (represented by longitude and latitude), etc. Readers may find more information about our dataset in the dataset section.

Preprint. Work in progress.

2 Background and related works

2.1 L_1 Regularization

The intuition behind regularization is to do model selection by restricting the parameter of models from being too large. There are many types of regularization, the most popular family is the ℓ_p norm regularization family. In this work, L_1 regularization has been applied, which guarantees a better convergence rate than L_2 regularization or non-penalization setting, if the given model assumption is correct. Moreover, L_1 regularization can be used as a tool of feature selection —- in first-order based convex optimization, adding L_1 regularization encourages the model to "shrink" insignificant weights to 0, which can be viewed as a tool that combines training and feature selection. (Andrew Ng, 2004)[4].

2.2 BERT

BERT, which is the abstraction of Bidirectional Encoder Representation from Transformers, is the state-of-the-art sentence-level language understanding model . It is designed to "pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers" (Jacob Devlin, et al. 2018)[3].

There are two steps in the BERT framework: pre-training and fine-tuning. In the pre-training step, BERT first masks some tokens at random and then trains a bidirectional language model to predict these masked tokens with a deep learning architecture of 12 layers and 768 hidden states. Then it strengthens the model with the *next sentence prediction*(NSP) task. To perform downstream language processing tasks such as question-answering and language classification, BERT plugs in the task-specific inputs and outputs and fine-tune all the parameters end to end. Only the last layer of the model is task-specific, thus fine-tuning pre-trained BERT is not as costly.[3]

2.3 GloVe

Glove is a pre-trained vector space representations of words combining the advantages of both global matrix factorization and local context window methods. The representation model is trained only on the nonzero elements in a word-word covariance matrix, which leads to a high performance on word analogy tasks. It also provides an accurate prediction on word similarity tests. (Jeffrey Pennington, et al. 2014)[2].

3 Dataset

Each of our input data contains a variety of features:

- Stars - Stars of the review (1=worst, 5=best). The value only takes integers between 1 and 5. This is only available in the training data.
- ID - Id number of the review. This is only available in the testing and validation data.
- Name - Name of the business
- Text - Review Text
- Date - Date of Review
- Useful - Number of Yelp users who thought the review was useful. This value takes whole numbers (i.e. 0,1,2,...)
- Funny - Number of Yelp users who thought the review was funny. This value takes whole numbers (i.e. 0,1,2,...)
- Cool - Number of Yelp users who thought the review was cool. This value takes whole numbers (i.e. 0,1,2,...)
- City - City in which the business is located
- Longitude - The business's longitude location

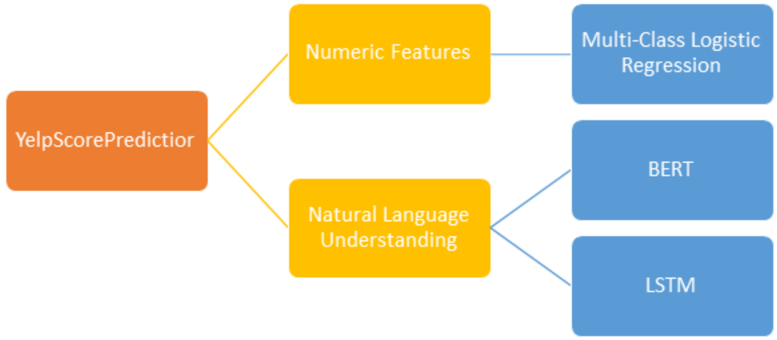

Figure 1: Method Outline

- Latitude - The business's latitude location
- Categories - An un-parsed string that categorizes the businesses into Yelp's categories.
- nchar - Number of characters in the review
- nword - Number of words in the review. The word count only considers words that are not part of the list in stop words (e.g. "a","I","the",etc.), which is part of the R package tidytext.
- Sentiment Score - A score of the text's sentiment using the AFINN lexicon, which ranges from -5 (very negative) to 5 (very positive).
- Gem - Number of times the word "gem" appears in the review text.
- Incredible - Number of times the word "incredible" appears in the review text.
- Perfection - Number of times the word "perfection" appears in the review text.
- BLANK - Number of times the word "BLANK" appears in the review text.

We partitioned the features into two classes - we call all the features besides the Text "numerical features" as they could be represented by numbers or discrete classes. For those features that are not represented by numbers, such as Category, we pre-process the data by assigning a number to each category and mapping the features to their numerical representations. In our baseline approach, we only used the numerical features to train the models. When we incorporated the text into the feature, we experimented with two pre-processing methods: using GloVe to represent each word in the text, and using BERT to generate an embedded model of the text.

4 Methods

We adapt slightly different methods for the two tasks: feature significance analysis and predictor model training. However, a similar outline is adapted by both tasks as shown in figure 1 above: in the baseline model, we use only numerical features to perform a linear or logistic regression with L_1 regularization. To improve the performance, we introduce more advanced language processing models such as BERT and LSTM to perform text analysis.

4.1 Feature Significance Analysis

Our baseline model is a linear regression with numerical features, which is everything except the comment text of the review data. The way we perform feature selection is via L_1 regularization. We expect the weight of insignificant features to shrink to 0 by L_1 regularization. Therefore, we can sort weights based on their norms, to get a ranking of features. To improve on that, we use BERT to generate a word-embedding for sentences. Then combining with numerical features we have used before, we train a final linear layer using this embedding with L_1 regularization.

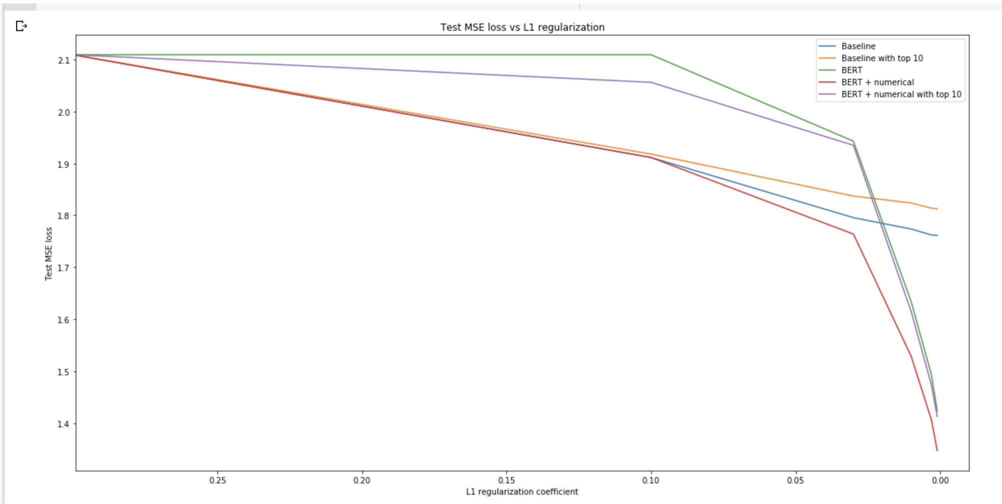

Figure 2: Mean-squared error with respect to L_1 regularization coefficient

4.2 Score Predictor

The baseline model is a simple logistic regression using again only the numerical features, and our final model is a fine-tuned BERT with the texts. More specifically, we first load the pre-trained BERT model, then adding up a final linear classification layer as the training target. We have strong confidence in this learning framework, since in the original publications of BERT, the developers achieved very high accuracy using this task-specific fine-tuning strategy.

5 Experiments and Results

5.1 Feature Analysis

5.1.1 Baseline: L_1 regularization

Since BERT is a strong and robust pre-trained model, we won't add a lot more complexity to our final model, as using a simpler model would drastically increase the interpretability of the results. Thus, we propose using linear model for ease of interpretation. For feature selection, we use linear regression with L_1 regularization, this combines training together with feature selection, using only the numerical features. Intuitively, L_1 regularization will pull the weights of useless features to 0, therefore, we can select features based on the weights of their norms.[4]

This model achieves a 1.77 mean squared error on test dataset. We then use only top 10 features ranked by the norm of weights, this gives a 1.81 test loss.

5.1.2 Using Pre-trained BERT Model

We feed review text into pre-trained BERT and use its last hidden states as our representation of text. Using purely BERT processed text, we get loss down to 1.42. Combining features in baseline and BERT, we finally get shrink loss to 1.35, which beats all our previous models. We validate the effectiveness by selecting top 100 features, and this model gives a loss of 1.41. See 5.1.2 for more details.

5.1.3 Analysis and Adjustments

In the top 10 features selected from our baseline model shown in figure 3 (5.1.2), it is worth noting that `nchar` and `nword` are ranked highest by their weights, which is in contrast of what we have thought about. We expect words that indicate strong positive or negative sentiment to have higher

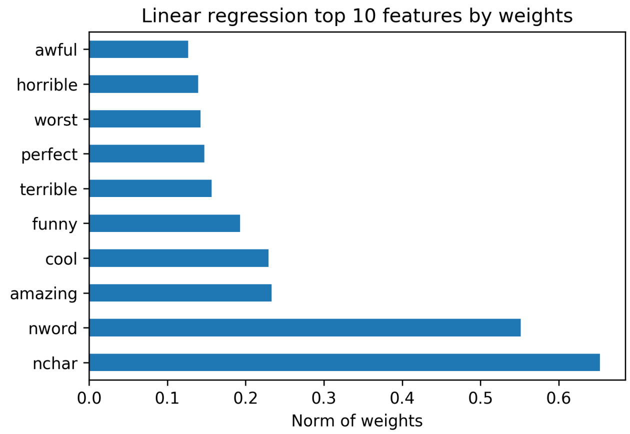

Figure 3: Top 10 features selected by baseline

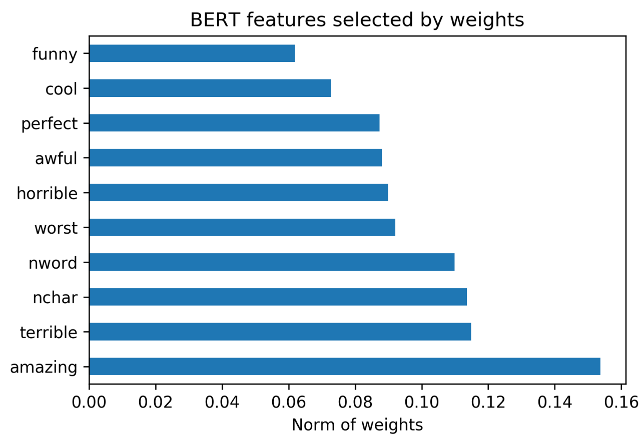

Figure 4: Top 10 numerical features selected by BERT-based model

weights, which is not the case in this model.

When using BERT, we expect the expressiveness of BERT will give a good representation of words and sentences, therefore, some features in that have high rank in baseline will be "washed out". See figure 4(5.1.3) for a comparison. The first observation is the norm of weights have decreased drastically. This meets our expectation, since BERT is more expressive than these numerical features, the contribution of numerical features have been significantly reduced. In our final BERT model, we have $768 + 35 = 803$ features, among the top 100 features, only 1 numerical feature appears, namely, the occurrence of word `amazing`. On the other hand, for the top 200 features, 33 numerical features have appeared, which means we haven't washed them out entirely via BERT embedding.

5.2 Predicting Scores

5.2.1 Baseline: Logistic regression with Cross-Validation

In score-predicting task, our baseline model consists of a simple logistic regression using only numerical features. Since linear regression is not computationally expensive, we can perform more comprehensive and sophisticated model selection by cross-validation, which is intrinsically hard for more complex models. With a 10-fold cross-validation, we get an accuracy of around 40%.

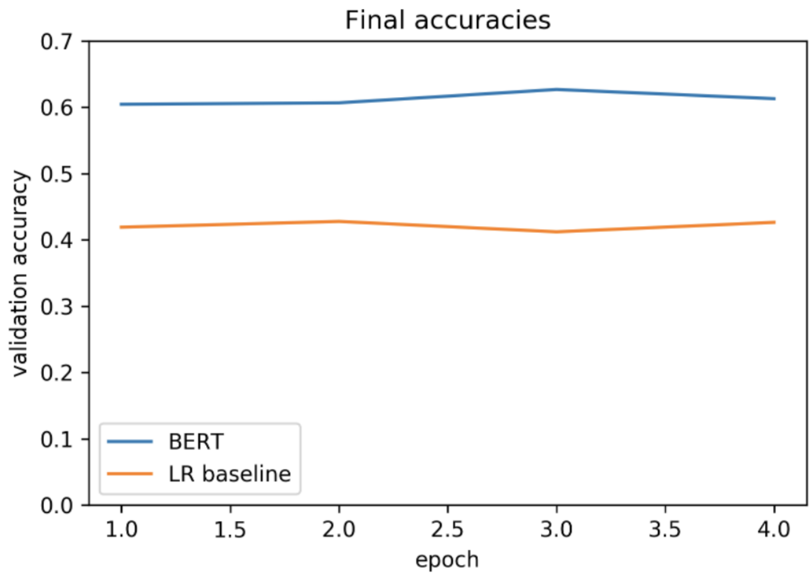

Figure 5: Final accuracy of predicting scores

5.2.2 Fine-tuning Pre-trained BERT model

We adapt the strategy described in [3], by adding an extra linear layer to the pre-trained BERT model, and train the parameters of last layer. Errors are still getting propagated back to the entire model, but most of the weight and coefficient tuning happens only in the last layer, so fine-tuning process is much faster than training the entire model. In our experiment, it takes around 50 minutes to train an epoch of our dataset, which compromised of around 30000 sentences. Our final accuracy of this model is around 62%. We refer readers to Figure 5(5.2.2) for figures.

5.2.3 Analysis and Adjustments

As the state-of-the-art natural language understanding model, BERT outperforms baseline model to a significant extent. However, it is worth noting that such accuracy has not yet achieved the performance of BERT on other sentiment analysis task. For example, in Netflix review prediction, BERT achieves an accuracy over 95%. We observed that BERT generally performs well when output is binary, i.e., in task like classifying whether a review is positive or not. Additionally, we measure the accuracy by using a 0-1 loss, i.e., if prediction matches output label, we give a 1, otherwise 0. Due to the structure of this specific task, it might be better to use more sensitive measurement of loss.

We also experimented on a different approach for text processing with GloVe[2] and LSTM, which stands for Long Short-term Memory and is essentially a variance of vanilla RNN. However, since GloVe performs transformation on word level, we realized that we do not have a good model to combine the vectors of the words and represent the whole text. Otherwise, we would have to train on an input dimension that is too large to be reasonable.

6 Conclusion and Future Work

6.1 Conclusion

Sentiment analysis and further classification and prediction is a non-trivial task. The key challenge to feature engineer the text into features. To address this problem, we make use of the state-of-the-art natural language understanding model, BERT, which generates an embedding for sentence. This pre-trained fine-tuned model proves to be useful for both tasks. In sentiment score regression problem, it reduces 30% loss compared to baseline model. For classifying star ratings, it improves

50% accuracy compared to baseline. Meanwhile, L_1 regularization proves to be a useful feature selection tool.

6.2 Reflections and Future Work

One might notice that, even though we are able to achieve a significant improvement over the baseline approach, a final testing accuracy of 62% is still far from optimal. One possible reason is labels of data are very imbalanced: one third of them are 4, one third of them are 5, and only one third are 1,2 and 3. We haven't found a way to address this problem. One possibility is to augment the labels using some generative way: for example, interpolate the data points, then using KNN to generate some similar data. The other idea is to switch to a more advanced generative model, such as the SMOTE algorithm[5]. We believe that these pre-processing methods will be able to increase the testing performance.

Another issue that we are yet to address is the problem of combining BERT with all the numerical features. We reason that padding a dense layer right before the output layer might be a feasible approach. However, if we simply join the embedded word representations of the text from BERT to the other numeric features in such a linear setting, there might not be a significant difference between this method and our feature analysis model. We might also propagate error back to BERT pre-trained weights. Due to time constraint (training one epoch of BERT takes more than 50 minutes), we don't have a clear answer to this question.

References

[1] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[2] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv e-prints*, page arXiv:1810.04805, Oct 2018.

[4] Andrew Y. Ng. Feature selection, l1 vs. l2 regularization, and rotational invariance. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 78–, New York, NY, USA, 2004. ACM.

[5] Kevin W. Bowyer, Nitesh V. Chawla, Lawrence O. Hall, and W. Philip Kegelmeyer. SMOTE: synthetic minority over-sampling technique. *CoRR*, abs/1106.1813, 2011.