Sentiment Analysis: Predicting Yelp Scores with BERT

Anonymous Author(s)
Affiliation
Address
email

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 L1 regression approach.

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)and Glove (Jeffrey Pennington, et al. 2014)and the sentence-level recognition models such as BERT, which stands for Bidirectional Encoder Representations from Transformers (Jacob Devlin, et al. 2018).

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.

2 Background and related works

2.1 L1 Regularization

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

Submitted to 32nd Conference on Neural Information Processing Systems (NIPS 2018). Do not distribute.

Figure 1: Methods Outline

bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers" (Jacob Devlin, et al. 2018).

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.

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)

3 Methods

We adapt slightly different methods for the two tasks: feature significance analysis and predictor model training. However, BERT is involved in both tasks as we try to improve on the baseline approach.

3.1 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 L1 regularization. We expect the weight of insignificant features to shrink to 0 by L1 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 L1 regularization.

3.2 Score Predictor

Baseline model is a simple logistic regression using again numerical features, and our final model is a fine-tuned BERT. More specifically, we first load the pre-trained BERT model, then add up a final linear classification layer, this final layer is what have trained for. We have strong confidence in this learning framework, since in the original paper of BERT, authors achieve very high accuracy using this fine-tuned strategy.

4 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
- 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.

5 Experiments and Results

5.1 Baseline: L1 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 L1 weight decay, this combines training together with feature selection. Intuitively, L1 regularization will pull the weights of useless features to 0, therefore, we can select features based on the weights of their norms.

Our baseline for this problem is a simple linear regression with L1 weight decay, using only numerical features. 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.

5.2 Fine-tuning with 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.

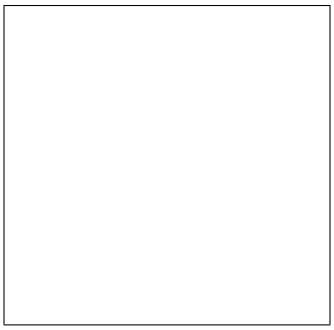

Figure 2: Sample figure caption.

5.3 Analysis and Adjustments

6 Conclusion and Future Work

6.1 Conclusion

Sentiment analysis and further classification and prediction is a non-trivial task. One key component is how to feature engineer the text into features, in order 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. Also, L1 weight decay 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 is switch to some more advanced generative model, such as the SMOTE algorithm. 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.

6.3 Figures

All artwork must be neat, clean, and legible. Lines should be dark enough for purposes of reproduction. The figure number and caption always appear after the figure. Place one line space before the figure caption and one line space after the figure. The figure caption should be lower case (except for first word and proper nouns); figures are numbered consecutively.

You may use color figures. However, it is best for the figure captions and the paper body to be legible if the paper is printed in either black/white or in color.

Table 1: Sample table title

	Part	
Name	Description	Size (μm)
Dendrite	Input terminal	$\sim$100
Axon	Output terminal	$\sim$10
Soma	Cell body	up to 10^6

6.4 Tables

All tables must be centered, neat, clean and legible. The table number and title always appear before the table. See Table 1.

Place one line space before the table title, one line space after the table title, and one line space after the table. The table title must be lower case (except for first word and proper nouns); tables are numbered consecutively.

Note that publication-quality tables *do not contain vertical rules.* We strongly suggest the use of the `booktabs` package, which allows for typesetting high-quality, professional tables:

`https://www.ctan.org/pkg/booktabs`

This package was used to typeset Table 1.

References

References follow the acknowledgments. Use unnumbered first-level heading for the references. Any choice of citation style is acceptable as long as you are consistent. It is permissible to reduce the font size to `small` (9 point) when listing the references. **Remember that you can use more than eight pages as long as the additional pages contain *only* cited references.**

[1] Alexander, J.A. & Mozer, M.C. (1995) Template-based algorithms for connectionist rule extraction. In G. Tesauro, D.S. Touretzky and T.K. Leen (eds.), *Advances in Neural Information Processing Systems 7*, pp. 609–616. Cambridge, MA: MIT Press.

[2] Bower, J.M. & Beeman, D. (1995) *The Book of GENESIS: Exploring Realistic Neural Models with the GEneral NEural SImulation System.* New York: TELOS/Springer–Verlag.

[3] Hasselmo, M.E., Schnell, E. & Barkai, E. (1995) Dynamics of learning and recall at excitatory recurrent synapses and cholinergic modulation in rat hippocampal region CA3. *Journal of Neuroscience* **15**(7):5249-5262.