CS230

Useful Review Prediction for Yelp Reviews

Christina Pan
capan@stanford.edu

John Martin Poothokaran
johnmp@stanford.edu

Abstract

Yelp ratings and recommendations help users choose which restaurants and businesses to visit. However, since not all Yelp reviews are useful, identifying the most helpful reviews would add to the customer experience. Since deep learning and natural language processing (NLP) techniques have not been applied on this domain yet, this paper will focus on applying NLP techniques used with deep learning. After comparing various NLP-specific architectures, the Long Short Term Memory (LSTM) architecture obtained the best result, with an average mean squared error (MSE) loss of 2.67.

1 Introduction

Yelp ratings and recommendations help users search for and pick restaurants and stores to visit. However, not all Yelp reviews are useful, and identifying the most helpful reviews would add to the customer experience and prevent the website from being overly cluttered.

While there has been a plethora of work upon this problem, none have applied deep learning, specifically natural language processing (NLP) techniques, upon this domain. In addition, because there is a significant difference in the usefulness of a review that received 1 useful vote compared to 200, we plan on doing a regression task.

Therefore, we will focus upon utilizing deep learning and NLP techniques upon the task of predicting the number of useful votes a given review will receive. The input to our algorithm(s) will be the text of a review in the form of 50 dimensional word embeddings. Then, we will feed these embeddings into a one of the various neural networks architectures, featuring architectures used in NLP. These models will then output a continuous number predicting the number of useful votes that review will receive.

2 Related work

Many have worked on the Yelp dataset to classify whether reviews are useful or not. Most of this work has been done on traditional machine learning techniques. For instance, Ghenai [1] describes applying SVMs, a naïve bayes classifier and a random forest classifier to a classification task with multiple classes, based on the majority class among the useful, funny, and cool votes received by a review. Zhang et al [2] apply SVMs, random forests and a number of K nearest neighbor experiments to predict the usefulness of a review using features from the user and business data provided by Yelp. Looking more generally at predicting the usefulness of reviews, Bhargava [3] modeled the usefulness of Amazon reviews, also as a classification problem. He, like Ghenai, used ensemble tree-based methods and obtained a high accuracy of around 85 percent. While classifying a review as useful or not is definitely helpful, knowing the gradation in which a review is helpful can be even more useful, especially when comparing between useful reviews.

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

In addition, NLP has made numerous significant breakthroughs in the most recent years with the help of deep learning. One of the state-of-the-art architectures is the bi-attentive classification network (BCN), which uses recent NLP developments, such as encoder-decoder LSTMs, attention, and convolution neural networks [4]. In addition, given that we have only one input - the text - for this problem, self-matching attention is one way to implement attention, which helps models focus on what part of the input to focus upon when generating output [5]. However, this form of attention is more finicky and requires more hyper-parameter tuning to work effectively.

3 Dataset and Features

The original Yelp dataset includes about 5 million reviews along with other user and business data [6]. Due to memory issues, we decided to include only reviews with at most 50 words and only look at the text to predict the number of useful votes a given review will obtain. This smaller dataset contains 1.5 million reviews, 1.35 million of which are in the train set, 75k in the dev set, and 75k in the test set.

In terms of the distribution of useful votes, the majority of reviews (75 percent) have 0 useful votes. 15 percent have 1 useful votes, and the maximum number of votes for a given review is 1341.

When cleaning the data, all of the text was stripped of any punctuation and turned into their lower case forms. However, any misspellings (e.g., "chrismas" instead of "christmas") or slang words that do not typically appear in the dictionary were kept as is. This new, stripped text then was turned into an embedding with pre-trained GLoVe vectors of 50 dimensions [7].

Ān example of the pre-processed data set (before being turned into its embedding form) is:
input: "excellent indian food i eat here all the time the atmosphere is excellent and everyone is extremely friendly"
output: 1

The cleaned data set contains 170 thousand unique vocabulary words. In Figure 1, we see the most frequent words (aside from words such as 'the' or 'a') based on how many reviews they appear in. These words touch upon the common themes, which include the food, service, and location along with the quality of these themes. Some of these words are associated with specific themes (e.g., "friendly"), while other characteristics can be applied on all categories (e.g., "great").

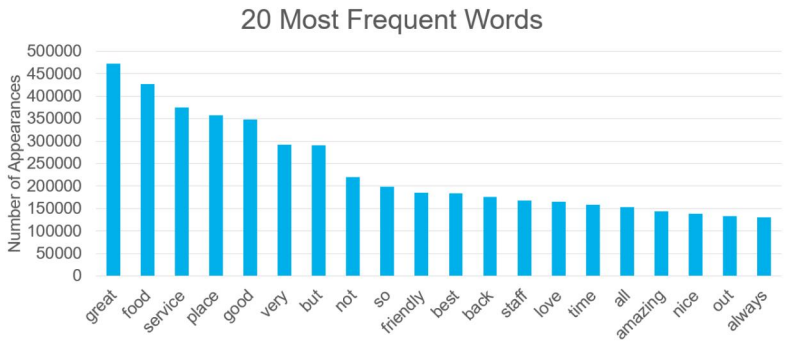

Figure 1: The 20 most frequent words in the processed dataset, excluding words like 'the' or 'a'. Note that the number of appearances calculated here counts the number of reviews in which a given word appears.

4 Methods

4.1 General Approach

In this paper, we tried various neural networks of varying complexity utilizing NLP techniques associated with deep learning. In particular, all models used an embedding layer as the first layer, with the embeddings being initialized with the 50 dimensional GLoVe word vector associated with the word.

All models use the Adam optimizer. In additional, all models utilize Mean Square Error (MSE) as their loss function (i.e., $MSE = 1/N \sum_{i=1}^{N} (\hat{y}_i - y_i)^2$ where N is the number of examples in the dataset and $\hat{y}_i$, y_i are the ith prediction and actual output in the set respectively).

4.2 Dense

This model contains an embedding layer that feeds into a single dense layer with a ReLu activation (i.e., $ReLu(x) = max(0, x)$) that outputs the predictions.

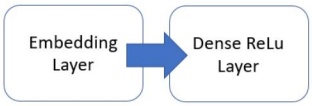

Figure 2: The architecture of the Dense model involving embeddings.

4.3 LSTM

This model consists of an embedding layer, a single feed-forward Long Short Term Memory network (LSTM) layer with 50 units, and a dense layer with ReLU activation which outputs the predictions. LSTMs are a special type of Recurrent Neural Network (RNN) which can track long term dependencies. RNNs are a type of neural network that retain previously inputted information by not only taking the new word (in the form of an embedding) but also the output of the previous word/input. However, retaining information from long ago (or long after) in a regular RNN is difficult (due to either exploding or vanishing gradients). Therefore, the LSTM was created to learn and retain long term dependencies within the data (e.g., a word referring to something 10 words prior). In addition, a feed-forward LSTM will only use information inputted before the current embedding.

Figure 3: The architecture of the basic LSTM model.

4.4 Deep LSTM

This neural network is similar to the simpler LSTM, except that it has additional layers. It consists of an embedding layer, 5 layers of LSTMs with 50 units per each layer, and a dense layer with a ReLU activation before a dense layer which outputs the predictions.

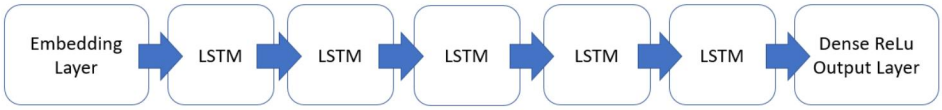

Figure 4: The architecture of the deep LSTM model.

4.5 Modified Bi-attentive Classification Network (BCN)

We decided to emulate the state-of-the-art bi-attentive classification network (BCN) used by McCann et al. The bi-attentive classification network is a model architecture that was first implemented in the source paper for single and double sentence classification tasks.

The model described by McCann et al [6] was modified to have a single 'review text' input, with the bi-attention mechanism replaced by self-matching attention. See Figure 5 below for the equations for

self-matching attention. The word embedding for a batch of sentences is passed through an initial ReLU network and then through a bidirectional LSTM encoder. A bidirectional LSTM can use inputs both before and after the given input, and an encoder converts the inputted text into meaning. This meaning can be used by the model to improve results. The attention layer - which helps the model focus on the most relevant parts of the inputted hidden layer - takes in a single sentence input, and follows implementation of self-matching attention from R-Net [5], which allows the attention scores to be calculated without a decoder vector.

$$s_j^t = \mathrm{v}^{\mathrm{T}} \tanh(W_v^P v_j^P + W_v^P v_t^P)$$
$$a_i^t = \exp(s_i^t) / \Sigma_{j=1}^n \exp(s_j^t)$$
$$c_t = \Sigma_{i=1}^n a_i^t v_i^P$$

Figure 5: These are the equations used for self-matching attention.

The softmax of the attention scores is then multiplied element-wise by the output of the LSTM encoder layer. The attention layer output is then passed through a bidirectional LSTM. The output is then passed through a layer that performs both max and average pooling, followed by three dense layers with dropout layers in between these layers during training.

Due to memory limitations, the parts of the model after the attention layer were further modified. The bidirectional LSTM layer after the attention layer was removed completely, and the number of dense layers at the end of the graph were reduced to 2, one with 5 units and the last one producing the prediction. See Figure 6 below to see the final architecture used in the modified BCN.

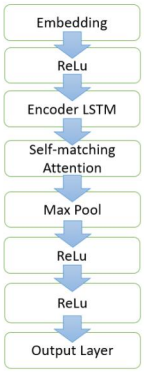

Figure 6: The final modified BCN architecture after overcoming memory and overfiting issues.

5 Experiments/Results/Discussion

5.1 Experiments and Hyperparameters

Given the large sets of data and the complexity of the later models, training every epoch takes a long time, limiting the number of experiments that could be performed. For instance, the deep LSTM architecture would take over an hour for each epoch, meaning that running that model for 10 epochs would take over 10 hours.

After a brief hyper-parameter search focusing especially upon the learning rate and the number of epochs, we found that the best hyper-parameters across the models are a learning rate of 0.001, β_1 of 0.9, and a β_2 of 0.999. In addition, the models each trained for 10 epochs with batches of 2048 to speed up training time. The modified BCN, due to memory issues however, trained with batches of 256.

5.2 Results and Discussion

Looking at the test results shown in Table 1, we can see that the basic LSTM architecture obtains the best test results, although the modified BCN architecture is quite close too. The raw test set error is significantly larger than the dev set error across the board (the dev set generally has a error of around 3 across the models). This is because the raw test set contains an outlier - the review with the maximum number of useful votes (i.e., 1341 compared to the typical range of 1-20). Once we process the data and remove this outlier, we see that the test set does about the same as the dev set results. Even with the processed test set, we see that the basic LSTM still performed the best, with a loss of 2.69. While there has been previous work on predicting the usefulness of Yelp reviews, those researchers decided to do classification, leaving no direct comparators for our regression results.

Model	Loss	Mean $\lvert y_i - \hat{y_i} \rvert$	Processed Loss	Processed Mean $\lvert y_i - \hat{y_i} \rvert$
Dense	27.72	5.26	2.93	1.71
Basic LSTM	27.46	5.24	2.67	1.63
Deep LSTM	27.53	5.25	2.74	1.66
Modified BCN	27.48	5.24	2.69	1.64

Table 1: This table contains the test set losses (MSE) and mean differences between the predicted and actual useful votes across the various models. Note that the first set contains the outlier with 1341 useful votes and the second does not.

Given that the deep LSTM and modified BCN architectures are more expressive than the basic LSTM's model, we would expect that these more powerful models would obtain better results. However, due to these models' complexities, these architectures are more sensitive to hyper-parameters. Therefore, more experiments and finer hyper-parameter tuning are necessary to fully utilize these architectures.

In terms of qualitative analysis, looking at the basic LSTM predictions elucidates common themes in reviews people find useful. Strongly opinionated reviews tend to be labeled as useful, especially if the reviewer is willing to commit and say that they either plan on coming back or never plan to do so. In addition, comparative statements, especially about the authenticity of the food are often considered a quality of useful reviews. For instance, "Amazing, authentic Yucatan food, not your traditional Tex-Mex Americanized cuisine," was considered a useful review by both the model and people in real life.

In addition, looking at the test set reviews, we can see that the usefulness of a review does not just depend on its text. For instance, the review with 1341 votes states: "I have gotten 3 Private messages from Samy asking me to provide a reciept for this meal. I do not like to be bothered by the owner of an establishment telling me I lied. I did in fact eat here, and I didn't like it that much." Looking just at the text, it seems odd that this review is considered useful, until we realized that the review was about Amy's Baking Company, a restaurant which went viral for their owners' attitudes, poor service, and social media responses. Therefore, this review most likely got so many useful votes because it went viral [8].

6 Conclusion/Future Work

We constructed a variety of NLP-specific architectures that took the text of reviews in word embedding form and predicted the number of useful votes that review would receive. Ultimately, the basic LSTM model performed the best with a test set loss of 2.67, with the modified BCN close behind with a test loss of 2.69.

Given the limited time and long amounts of time necessary for training, the more complex models could be improved with extra hyper-parameter tuning. In addition, including other pertinent information about the review, which can range from the post's position on Yelp to other information on the business or reviewer, could be incorporated to help improve the model. This is especially true given that non-textual data can really impact the number of useful votes obtained, as we have seen in the review with the most useful votes. Finally, turning this task into a classification one would be another interesting area to explore.

7 Contributions

Christina pre-processed the data, implemented the Dense, basic LSTM, and deep LSTM architectures (in Keras), and ran experiments along with writing the report.
John wrote the various iterations of the modified BCN architecture (including another implementation of a LSTM) in Tensorflow and wrote the report.

References

[1] A. Ghenai *"Learning in Translation: Contextualized Word Vectors"*. ". University of Waterloo. CS 886 Project Report. December 2014. URL: `https://cs.uwaterloo.ca/~aghenai/assets/publications/Amira_Ghenai_Final_Report.pdf`

[2] H. Zhang, X. Liu, K. Ying. *"Reviews Usefulness Prediction for Yelp Dataset"*. University of California San Diego. March 2017 URL: `https://cseweb.ucsd.edu/classes/wi17/cse258-a/reports/a040.pdf`

[3] S. Bhargava. *"Predicting Amazon review helpfulness"* . University of California, San Diego. CSE 225 Assignment. 2015. URL: `https://cseweb.ucsd.edu/~jmcauley/cse255/reports/wi15/Shitij_Bhargava.pdf`

[4] B. McCann, J. Bradbury, C. Xiong, R. Socher. *"Learning in Translation: Contextualized Word Vectors"*. 17th March 2018. URL: `https://arxiv.org/pdf/1708.00107.pdf`

[5] Natural Language Computing Group, Microsoft Research Asia. *"R-Net: Machine Reading Comprehension with Self Matching Networks"*. URL: `https://www.microsoft.com/en-us/research/wp-content/uploads/2017/05/r-net.pdf`

[6] Yelp. *Yelp Dataset Challenge*. n.d. URL: `https://www.yelp.com/dataset/challenge`

[7] J. Pennington, R. Socher, C. Manning. *"GloVe: Global Vectors for Word Representations"*. n.d. URL: `https://nlp.stanford.edu/projects/glove/`

[8] K. Clay. *Lessons From Amy's Baking Company: Six Things You Should Never Do on Social Media*. 14th May 2013. URL: `https://www.forbes.com/sites/kellyclay/2013/05/14/lessons-from-amys-baking-company-six-things-you-should-never-do-on-social-media/#6c0e9eb42f1d`

[9] TensorFlow. *Tensorflow* n.d. URL: `https://www.tensorflow.org/`

[10] Keras. *Keras* n.d. URL: `https://keras.io/`

Github Repository Link: `https://github.com/cpan427/yelpReviewQualityPredictor`