Various Approaches to Sentiment Classification for Yelp Reviews

Joohyung Shin
joohyuns@andrew.cmu.edu

Guannan Tang
guannant@andrew.cmu.edu

Neil Mehta
neilashm@andrew.cmu.edu

Abstract

Modern day social media platforms allow users to express their sentiments about businesses and products online. It has become increasingly important to be able to classify this data for the purposes of market research and strategy. In this work, we use primarily Yelp review data text to predict review ratings using several machine learning models. We achieve adequate baseline results and improve them using novel methods.

1 Introduction

In the context of sentiment analysis, the art of predicting subjective information across various media, the internet has quickly become a source of vast amounts of data. In this report, we develop models to predict Yelp business review ratings based on the sentiment of the review text. To formally define the problem, let C be the set of class labels associated with the star rating of a review, $\{1, 2, 3, 4, 5\}$. y_{true} represents the true label from this set for a given sequence of review text, $\{x_i\}_{i=0}^{L}$ where L is not fixed. We are interested in learning a model, f that returns a probability distribution over the class labels for a given review. In other words we seek to minimize the classification error.

$$\epsilon_{classification} = \min |y_{true} - argmax f(\{x_i\}_{i=0}^{L})| \tag{1}$$

To solve this problem, we explore several models which achieve varying degrees of success. Throughout the experiments, we demonstrate that this non-trivial task is not easy to generalize using existing techniques and propose new methods to be explored.

2 Background and Related Works

2.1 Sentiment Analysis

Sentiment analysis is essentially a subjectivity analysis that works with words, sentences and any other expressions in language. One major problem remaining at the very center of this analysis is to categorize the sentiment polarity[3]. One direct research interest in this context is review rating prediction, which has been an active field of research for many years. Qu et al. (2010) has used sentiment analysis as a feature extraction technique to capture the sentiment orientation in Amazon reviews and has successfully related them to the review ratings[7]. Ganu et al (2009) proposed a better alternative to conventional recommendation systems by utilizing the sentiment information of the restaurant reviews together with their corresponding textual information[4].

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

2.2 Yelp Score Prediction

In this category of review rating prediction, the Yelp dataset challenge has been one of the top problems on which research has focused. Previous works were oriented toward analyzing review text with different feature engineering techniques. Xu et al.(2014) used an opinion lexicon to encode yelp review text and experimented with their data using various learning algorithms. The best result in their research uses Naïve Bayes with stop words removed and stemming[8]. Xue et al.(2015) on the other hand used a world cloud to visualize yelp review text and represented the content with Bigrams and Trigrams, which were then passed through different learning algorithms to make predictions of the rating. The best results were obtained by using Linear Regression and Random Forest Trees[11].

As opposed to the above, Yu et al. [10] focused on learning algorithms and implemented several complex neural networks to tackle the Yelp score prediction problem. The best result was achieved using a recurrent neural network with an LSTM[9]. In this report, we are interested in exploring algorithms with more complex structures than models already available in the literature. We believe that building models with higher complexity (e.g. hierarchy, multiple inputs) can help to improve the prediction precision in this difficult task.

3 Methods

3.1 Ensemble Logistic Regression

To form a feature matrix, we first separated review texts into five different categories according to their review ratings. Then, the top K most frequent words in each category were extracted and used as indices of the feature matrix. We compare these indices with each review text and count the number of occurrences of each of the most frequent words to build the feature matrix. Table 1 shows an example of the feature matrix. In this manner, we generated five feature matrices.

	word 1	word 2	word 3	...	word K
review 1	count	..	...	...	...
...	...	...	...	...	...
review n	...	...	...	...	...

Table 1: Example of the feature matrix

	1 Star	2 Star	...	...	5 Star
review 1	0.2	0.4	...	...	0.22
...	...	...	...	...	...
review n	...	...	...	...	...

Table 2: Example of the probability matrix

Multinomial Logistic Regression (MLR) was used as the learning method. We trained separately on each of those five feature matrices to obtain five MLR models so that each of them is biased to be good at predicting their corresponding rating. Eventually, we output the prediction probability matrix of each MLR model and combine them by simply taking the element-wise product over all five output prediction probability matrices. An example of the prediction probability matrix is given in Table 2.

Besides the review text, we also trained MLR upon other features (useful, funny, cool, nchar, nword and sentiment score). In the same manner as above, we extracted the prediction probability matrix from this MLR model and combined it with the probability matrix calculated out of the review text by element-wise product.

The eventual prediction is made based on whichever label (rating) in the combined probability matrix has the highest probability value. The general structure of this method is indicated in Figure 1.

For comparison purposes, we also used one-hot encoding based on word occurrence to vectorize the review text and trained it all at once with MLR.

3.2 Multinomial Naive Bayes

Most of the pre-processing steps of making feature matrices are similar to the steps in Ensemble Logistic regression. However, for Multinomial Naive Bayes, we do not separate into five different categories. Rather, we tried to find the appropriate number of word features and words to maximize the accuracy score. In this section, we will present first how we come up with Naive Bayes classifier, and second move to a Multinomial Naive Bayes classifier.

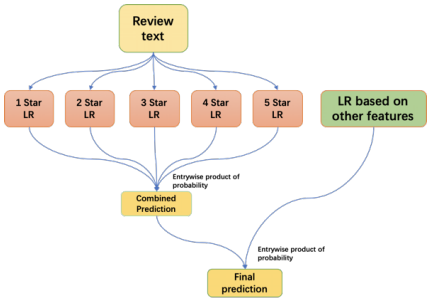


Figure 1: Block Diagram for Ensemble Logistic Regression

3.2.1 Naive Bayes classifier

Before we implement the Multinomial Naive Bayes, we first tested the accuracy of a simple Naive Bayes classifier with a one hot encoded text feature matrix to assess the performance of the Naive Bayes classifier on the given dataset. In general, Naive Bayes assumes strong independence between the features in the model, rather than assuming a particular distribution. In the case of text classification, we created a feature matrix for the Naive Bayes classifier, and each feature is assumed to be conditionally independent to one another given a certain label.

$$p(w_1, \dots w_n|c) = \prod_{i=1}^{n} p(w_i|c)$$

So to formally express this problem, if the purpose of the Naive Bayes classifier is to calculate the probability of observing features (words, notated as w_i), given some class c, the above equation holds. Based on the above equation, we can analyze the probability based on each of the posterior probabilities.

$$p(c|w_1, \dots w_n) \propto p(c)p(w_1|c) \dots p(w_n|c)$$

However, the simply implemented Naive Bayes classifier based on baseline model[2] was not very effective. We achieved about 25-28% scores throughout the baseline implementation.

3.2.2 Applying Multinomial Naive Bayes Classifier

After we implemented the baseline model with Naive Bayes classifier, we theorized the low accuracy score with the Naive Bayes classifier is due to the distribution of word count which does not follow a Gaussian distribution. When we created the feature matrix based on the text data, we simply transformed into vectors which expressed the count of each word in each review. Base on this quantification method, we cannot ensure each feature follows a Gaussian distribution as is the assumption in the normal Naive Bayes classifier. Therefore, we decided to use a Multinomial Naive Bayes classifier to fit to the distribution of our feature matrix.

3.2.3 Variations in Feature Matrix

Prior studies on text classification which used the Naive Bayes classifier utilized many different word embeddings. For example in their studies [1] on subjective sentiment analysis based on newspapers, researchers utilized tf-idf embeddings to extract important words that differentiate the news article from others. Motivated by such approaches, we also tried to test both word count embeddings and tf-idf embeddings.

word count embeddings : In this embedding method, we transform target words as features and count occurrences of each word. For example if the word "happy" occurs 3 times in a given sentence, it will be embedded as form of $\{happy : 3\}$.

binary word count embeddings : The Binary word count method counts the initial occurrence and ignores how many times that word actually occurs. So if word occurs at least one time, it is counted as 1 otherwise it is 0.

tf-idf embeddings : *tf-idf* is similar in its logic to the one-hot encoding method used in word count embeddings. Instead of only counting the occurrence, tf-idf focuses on the relation to the entire corpus and this allows us to find how important a feature is to a document.

Based on the Multinomial Naive Bayes Classifier, we will extend the experiments to find the appropriate size of the word bag and a new combination of Multinomial Naive Bayes with different embeddings. Experiments and analysis on the results will be further explained in the Result section.

3.3 LSTM Models

3.3.1 General Approach

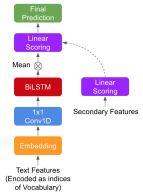


Figure 2: Block Diagram for LSTM Model

To properly apply LSTMs to this problem, we start by implementing a vocabulary of all words in the review text across the training dataset. Reverse sorted by frequency of occurrence in the training data, we create indices for each word which we can input to an embedding layer to generate word embeddings. As opposed to previous methods discussed which treat each word as a one-hot vector, we are interested in implicitly learning a continuous higher-dimensional representation of the words based on semantic information about each word. Similar to approaches such as Word2Vec, we would like words that serve similar functions in a sentence or have similar meaning to be close in a higher-dimensional space so that we may better make inferences about the meaning of a sentence [6]. Furthermore, the use of the LSTM is beneficial for utilizing the context surrounding a word at a particular time-step via the hidden states. An important note is that the network we use in this report is of variable sequence length. Rather than truncate the input to a fixed length, we would like to preserve information about the underlying features as much as possible. We do, however, recast the review text in terms of only the most frequent words which appear in the vocabulary. This ensures that we are learning a generalized representation of the data since words which appear infrequently and would not contribute to more than a few training instances are not included in the review text.

After passing through the embedding layer, we use a convolutional layer with stride=1 and kernel size=1 to extract relevant features from the word embeddings. This is not novel to the study of recurrent neural networks, but a literature review did not produce any examples where this had been applied to the problem of multi-class sentiment prediction [10].

Next, we pack the embedding features to obtain a format for the data that is more readily processed by the LSTM. We use a bidirectional BiLSTM as opposed to the work of Yu et. al. [10] to ensure that each time step is more informative since it carries the hidden states of not just the previous time steps, but also future time steps. This ensures that we obtain more contextual information for each word.

After the BiLSTM, we obtain logits over the class labels for each time step. Let each set of logits associated with a time step be denoted by a 1 by 5 vector h where a sequence of outputs is given as $(..., h_{t-2}, h_{t-1}, h_t)$. We average the logits from each time step to achieve a single vector over the class labels with which to make inferences about the star rating of a review. This is especially sound conceptually in the case with BiLSTMs since all time steps have some information about both the past and future.

$$y_{predicted} = argmax(\tfrac{1}{T}\sum_{i=1}^{T} h_i)$$

This vector is passed through a linear scoring layer and the argmax is taken to determine the final predicted class. Softmax cross entropy is used to train the model.

3.3.2 Utilizing Secondary Features

An extension of this model was implemented using other features of the training data in tandem with the review text. The "usefulness" score, sentiment score, number of words, and number of characters were used to tune the predictions. This was accomplished by injecting these values in to a linear scoring layer after taking the mean across all outputs of the BiLSTM. Another linear layer was appended after this scoring layer to learn weights between the review text and secondary features, and the final linear layer. Conceptually, sentiment score and usefulness score should be highly correlated to the star rating and could be used to make more accurate predictions in cases where the review text was not informative.

3.3.3 Regression with LSTM

Another BiLSTM model was implemented that recast the sentiment classification problem as a regression problem. Intuitively, there is a notion of "distance" between star ratings. If the true label of a review is 1 star, then misclassifying it as a 2 star review versus a 5 star review are not equally erroneous. In this sense, we can view the space of review ratings as continuous, which implies that regression could better approximate the true label of a review. To accomplish this, we can change the structure of our final linear scoring layer to output a single number and train the network with a mean squared error loss function. For the benefit of faster training, we can use a scaled hyperbolic tangent activation after the final scoring layer to bound our outputs to the range of a valid review rating. Here we are using $\alpha = 2.5$ and $\beta = 1$ to achieve a range of $[0, 5]$.

$$Scaled\_Tanh(x) = \alpha(Tanh(x) + \beta)$$

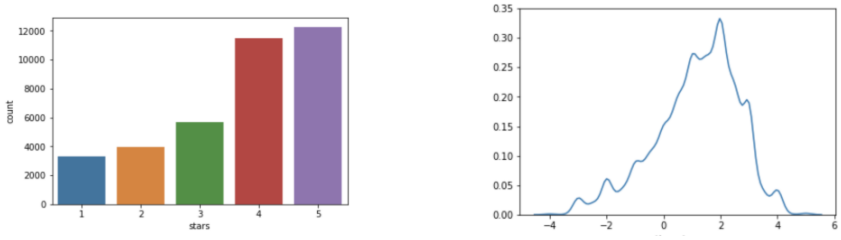
In the case of regression, our final prediction is obtained by rounding the output of our network to the closest integer. The benefits and drawbacks of these ideas are further discussed in the *Experiments and Results* section.

4 Dataset

4.1 Yelp Dataset

The Yelp training Dataset includes 36,692 Yelp Reviews. The given dataset consisted of train, validation, and test sets. However, the validation and test sets were unlabeled, so we were not able to utilize validation and test set for its intended purpose. Instead, we split the train set in to an 8:2 ratio, training to test. The dataset includes features including but not limited to "stars", "names", "sentiment_score" etc.

4.1.1 Characteristics of Yelp Dataset



(a) stars distribution (b) sentiment score distribution

Figure 3: As seen in above distribution plots, we are able to notice that given Yelp Dataset has positive bias which means star ratings for 4 and 5 appears more in dataset.

4.2 Pre-processing methods

In order to utilize text data effectively, it is necessary to pre-process it to obtain the most generalizable form of the data possible. These methods can be viewed as a form of "de-noising," which can help to

make clearer predictions from the data. We begin by performing a process called stemming in which we transform all words with the same root back to their root.

$$Stem(\text{"}delicacy\text{"},\text{"}delicious\text{"}) \rightarrow \text{"}delic\text{"}$$

This ensures that words with similar meaning are not counted separately in the case of binning for ELR and MNB, and ensures that separate word embeddings are not learned for these words in the case of the LSTM models. The next step is to remove uninformative words in every day speech called "stop words." Words like "the," "if," and "was," are not correlated with a specific sentiment, and their function in our models would be equivalent to noisy data. Similarly, punctuation is usually uninformative and can be removed.

For ELR, the remain words are encoded with top frequent words. One hot encoding is also used as a comparison. For MNB, it is useful to encode the remaining words as one-hot vectors. It regard each word as features. So to find appropriate set of words and appropriate size of word bags, we basically one hot encoded words and uses as feature matrix.

For LSTM models, we are trying to learn an implicit mapping between words based on similar meaning or usage. We are using an embedding layer in our network and can therefore recast words as integer indices of a vocabulary. The embedding layer creates a table which stores vectors associated with each word. These vectors are not one-hot, instead they contain continuous values allowing similar words to be closer in some higher-dimensional space. This distance metric in a higher-dimensional space is what is implicitly used as a definition of "similar" in terms of meaning and usage. We also tested methods involving recasting the review text in terms of only the most frequent words for this approach. We found optimal results keeping only the top 5000 most frequent words in the vocabulary.

5 Experiments and Results

Our goal is to minimize the empirical prediction error as stated in the introduction section. We use classification accuracy as the metric of performance for our models. The equation is given below:

$$\epsilon_{classification} = \frac{1}{N}\sum_i (y^i_{true} - predicted\_label^i) \tag{2}$$

5.1 Ensemble Logistic Regression

To evaluate the Ensemble Logistic Regression, we surveyed a range of parameters of most frequent words used in forming the feature matrix. Selected results are summarized in Table 3. Notice that accuracy of the final prediction goes up initially increasing the number of words in the vocabulary. At a point, the accuracy starts to go down indicating there is a maximum in test accuracy dependent on the choice of how many of the most frequent words we keep. The best accuracy we achieve with this model is 56.6%. It is worth it to point out that the best accuracy of a Logistic Regression model that is only based on the most frequent words of all of the training data can only reach 53%. As another comparison, we also evaluated the model based on the one-hot encoding technique. Selected results are summarised in Table 3. It is clear that the same trend was observed between accuracy and the the number of words considered in the feature engineering process. Notice that the best accuracy that can be achieved in this model is 55.96%, which is slightly lower than what we obtained from Ensemble Logistic Regression. From the discussion above, we can see that the same trend between

number of words	10000	15000	20000	25000
ELR(count)	0.54268	0.5594	0.5518	0.5418
number of top words	500	1000	1500	2000
ELR(top freq words)	0.5464	0.566	0.5631	0.5604

Table 3: Prediction accuracy for Ensemble Logistic Regression(ELR)

number of words used in the feature engineering process and the eventual prediction accuracy was observed in both cases. We believe this has to do with the nature of the review text. There are always meaningless parts and more meaningful parts in a sentence in terms of identifying the sentiment of that sentence. Initially when we increased the number of words used in the model, we actually increased the number of meaningful words considered in the model. After the maximum, we started to introduce less meaningful words in to the model. This eventually decreases the prediction accuracy

of our models. More importantly, Ensemble Logistic Regression exhibits better prediction accuracy over a wider range of the number of words used in the model. This supports our hypothesis that rather than directly using logistic regression, we can achieve a better prediction accuracy by implementing models with structure or hierarchy.

5.2 Multinomial Naive Bayes

number of words	10000	15000	20000	25000
MNB(tf-idf)	0.5368	0.263	0.5183	0.5068
MNB(count)	0.5704	0.5725	0.5783	0.5789
MNB(bin)	0.5739	0.5795	0.5819	0.5762

Table 4: Prediction accuracy for Multinomial Bayes classifier

To evaluate Multinomial Naive Bayes with the combination of different embeddings, we prepared different sizes of word bag to find the most appropriate combination. Since the results based on word embeddings are highly influenced by the starting set of corpus, we first start with fewer words to find the tendency and appropriate number of word features that we have to make.
As noted in table 4, we see that there is some range in number of words (appropriate size) that gives an increase in accuracy. Regardless of the embedding methods that we have applied, if too many word features are included in the corpus, the probability of specific features are not influential enough for the classification process in Naive Bayes classification, therefore the accuracy score goes down. As

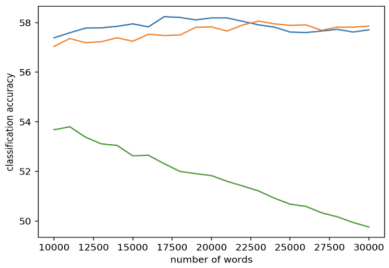


Figure 4: Green line shows tf-idf embeddings, Blue is for binary word count and Orange is for normal word count, tf-idf does not show similar results, and binary word count shows the best result when number of word features are about 18000.

seen in Figure 4, we see that tf-idf is not showing a similar result compared to word count and binary word count embeddings. In general, we estimate that this is because there are not many specific words in our pre-processed sentences to differentiate certain labels from others. To utilize tf-idf, there should be some "signal words" that can differentiate one object from the other, but the irregularity of the review sentences is too large to take relations from those reviews. However, by using binary word count, we were able to lessen the influence of high occurrence - non influential words effectively, and able to get a best accuracy score of 58.19%.

5.3 LSTM Models

	Yu et. Al.	BiLSTM	Ensemble	Regress. w/o Tanh	Regress. with Tanh
Accuracy	51%	59.2%	56.1%	42.4%	47.39%

Table 5: Best Validation Results of LSTM Models. Results taken over 20 epochs.

Of all surveyed methods the basic LSTM model performed best and allowed the most flexibility in terms of tuneable parameters. Through experimentation, we determined that the model had a tendency to overfit quickly depending on the number of parameters in the network. This was evident when training loss far surpassed validation loss in the early epochs. To help the network learn a more general representation of the data, we determined that utilizing dropout in between each of the layers with a probability of 0.4 and minimizing the batch size, hidden dimension size, and word embedding

size was effective in preventing overfitting to an extent. Through trial and error, we settled on batch size=32, hidden dimension size=64, and embedding size=100.

The ensemble method performed slightly worse than the BiLSTM using only review text. We theorize this is due to conflicting predictions between different features and lack of a more complex structure for incorporating the secondary features. Future iterations of this work would involve a more principled MLP for making predictions using secondary features.

The regression models performed significantly worse than the models trained using cross entropy. This is theorized to be because finding the correct label on a continuous space is a much harder problem. In the case of classification, we assign probabilities to each class whereas in regression we aim to find the exact correct value (or close to) that denotes the true label. With a network whose output is also continuous this is especially difficult to do in a relatively small range of numbers as is the case with star ratings.

Of interest is the fact that applying a scaled hyperbolic tangent activation to the output of the regression model dramatically improved accuracy. This is due to the aforementioned benefits of the hyperbolic tangent function. With bounded outputs, the network can learn to predict values within the appropriate range much easier than if the predictions are over the entire space of real numbers.

6 Conclusion and Future Work

6.1 Ensemble Logistic Regression

In the Ensemble method of Logistic regression, we have tried to combine several biased classifiers together to form an ensemble network. We surveyed this ensemble structure model and compared it with some simpler models that are implemented without any structure or hierarchy. The final results support our hypothesis that better prediction accuracy can be obtained with implementation of models of higher complexity.
As for future work, one implementation to investigate is to replace the simple method of element-wise product used in probability matrix combining process with a learning algorithm that learns the weights between the probability matrix and different features. We could then combine these predictions with the element-wise product. Also, the identification of a maximum in performance during the training process indicates that it will also be worthwhile to do some work on feature engineering and filter out unimportant words.

6.2 Multinomial Naive Bayes

In Multinomial Naive Bayes, we tried to extend from normal Naive Bayes and tried to combine with different types of embeddings. Although there were other approaches which implemented tf-idf, we propose that binary word count embeddings are much more applicable in Yelp Review Dataset.

In the future, one extended implementation can be to utilize already structured word embeddings - Glove word embeddings, for example. Rather than manually creating a word corpus, Glove word embeddings can be helpful for structuring an effective feature matrix.

6.3 LSTM Models

In this work we showed that LSTM models are among the best for analyzing time-series data such as text for sentiment prediction. The results suggest there is room for improvement. The addition of a convolutional layer before the LSTM did help, but feature engineering could be further explored especially in the case of secondary features to improve rating predictions. Even text could be further processed to achieve better results as previously mentioned. An idea explored in literature is to break down reviews in to single sentences and assign the labels accordingly [5] with better results. Though this was used for binary classification, we are interested in investigating if this would achieve similar results for multi-class prediction.
In the case of the regression models, we showed that scaling the outputs using the hyperbolic tangent function provided promising results, but the underlying issues with viewing the task as regression remain. In future work, we would like to explore the use of different loss functions along with the aforementioned feature engineering to improve results.

References

[1] Sadik Bessou and Rania Aberkane. "Subjective Sentiment Analysis for Arabic Newswire Comments". In: *arXiv e-prints*, arXiv:1911.03776 (Nov. 2019), arXiv:1911.03776. arXiv: 1911.03776 [cs.CL].

[2] Lopamudra Dey et al. "Sentiment Analysis of Review Datasets Using Naïve Bayes' and K-NN Classifier". In: *International Journal of Information Engineering and Electronic Business* 8.4 (July 2016), pp. 54–62. ISSN: 2074-9031. DOI: 10.5815/ijieeb.2016.04.07. URL: http://dx.doi.org/10.5815/ijieeb.2016.04.07.

[3] Xing Fang and Justin Zhan. "Sentiment analysis using product review data". In: *Journal of Big Data* 2.1 (2015), p. 5.

[4] Gayatree Ganu, Noemie Elhadad, and Amélie Marian. "Beyond the stars: improving rating predictions using review text content." In: *WebDB*. Vol. 9. Citeseer. 2009, pp. 1–6.

[5] Yoon Kim. "Convolutional Neural Networks for Sentence Classification". In: *CoRR* abs/1408.5882 (2014). arXiv: 1408.5882. URL: http://arxiv.org/abs/1408.5882.

[6] Tomas Mikolov et al. *Efficient Estimation of Word Representations in Vector Space*. 2013. arXiv: 1301.3781 [cs.CL]. URL: https://arxiv.org/abs/1301.3781.

[7] Lizhen Qu, Georgiana Ifrim, and Gerhard Weikum. "The bag-of-opinions method for review rating prediction from sparse text patterns". In: *Proceedings of the 23rd international conference on computational linguistics*. Association for Computational Linguistics. 2010, pp. 913–921.

[8] Yun Xu, Xinhui Wu, and Qinxia Wang. *Sentiment Analysis of Yelp's Ratings Based on Text Reviews*. 2014.

[9] April Yu and Daryl Chang. "Multiclass Sentiment Prediction using Yelp Business". In: ().

[10] April Yu and Daryl Chang. *Multiclass Sentiment Prediction using Yelp Business Reviews*. June 2015. URL: https://cs224d.stanford.edu/reports/YuApril.pdf.

[11] Mengqi Yu, Meng Xue, and Wenjia Ouyang. *Restaurants Review Star Prediction for Yelp Dataset*. Tech. rep. Technical Report 17. UCSD, 2015.