Exploring the Yelp Data Set: Extracting Useful Features with Text Mining and Exploring Regression Techniques for Count Data

Anonymous Author(s)
Affiliation
Address
email

Abstract

We attempted to predict the number of 'Useful' votes a fresh Yelp review will receive using training data provided by Yelp in a recent Kaggle competition. Features were extracted from the review text with text mining techniques, and other features were created from supplementary user, business, and review data sets. We explored the use of Random Forest Regression and LASSO Regression to predict how many 'Useful' votes a review received. We also investigated the use of Negative Binomial Regression and Zero Inflated Negative Binomial Regression using features selected with the aid of both the trained Random Forests and the LASSO models. The different prediction techniques were then compared and analyzed.

1 Introduction and Motivation

Generally speaking, people want to mitigate the risk of their decisions. One way to reduce risk is to consult the experiences of others before making a decision. For example, when considering the purchase of a new laptop, one can try to avoid a potentially bad purchase by consulting the ratings and reviews written by people who have already purchased the laptop. With the Internet, it has become easier to access and share reviews for almost every product and service imaginable.

However, some issues can arise due to the vast amounts of information that is available on the web. With respect to review data, we have the problem of data quality and data freshness. If left unchecked, users would have to sift through a multitude of data to find up-to-date reviews that are of good quality. Machine learning can alleviate these issues by learning which reviews are considered to be of good quality from the users and then predicting the quality of new reviews without depending on the input of the users.

After a brief literature search, we found there has been some success at approaching this problem as a classification task [1]. In this paper, we attempt to answer a regression problem rather than a classification problem - that is to say, for a given review, can we use machine learning to predict the number of users that will find the review useful?

1.1 Yelp

Yelp is a local business directory that includes social networking features. With the help of its users, Yelp provides a wealth of information about a business, such as its location, price range, and ambiance, among many other things. More interestingly, users are able to rate and write reviews about a business. The moderating of the reviews is largely left to the community, where users can judge the quality of a review by flagging it as being 'Useful', 'Funny', or 'Cool'. Other social networking features include the ability to 'check-in' to the business via the Yelp mobile app.

Yelp recently issued a Kaggle contest that challenges participants to predict the number of 'Useful' votes a fresh review will receive. In this paper, we explain the approach we took to address this question.

1.2 The Data

The data sets that Yelp provided consisted of 229,907 reviews of 11,537 businesses in the Phoenix, Arizona area by 43,873 users. A data set containing 8,282 sets of check-ins was also provided. The training data was collected from March 2005 up to January 2013.

Due to the daily submission limit on the Kaggle competition, it was inconvenient for us to assess our model performance on their test set, as we were training many models. Instead, we tried to emulate the conditions of the competition by splitting the training data into two parts. The oldest 80% of the reviews were used to train the models, and testing was done on the remaining 20% of the data. We will refer to the older 80% of the data as the training set and the newer 20% as the validation set unless otherwise specified. Once the models were tuned based on the results of the validation set, we trained the final models on the entirety of the labeled data and submitted our predictions for Yelp's private test set.

2 Methods

2.1 Data Processing and Feature Extraction

Before any analysis was done, features had to be extracted from the review text using text mining techniques. All work on the review text was done using the text mining package `tm` [2] in the `R` statistical programming language. We followed the basic work flow for preprocessing the data and extracting features as proposed by the `tm` package authors [2].

We chose to pursue a simple *bag-of-words* representation of the text data in a document term matrix (DTM), where a DTM has the documents as rows, the words as columns and a (weighted) frequency measure (explained below) of how many times the *j*th word appears in the *i*th document as the entry in row *i* and column *j*. These text features were later augmented with other features provided in the supplementary data sets.

2.1.1 Processing the Text Data

Some preprocessing of the data was necessary before features could be extracted from the text. The data cleaning process involved removing formatting, converting the data into plain text, stemming words, removing whitespace and uppercase characters, and removing stopwords.

Stemming [3] refers to the process of erasing word suffixes to retrieve the root (or stem) of the words, which reduces the complexity of the data without significant loss of information in a bag-of-words representations of text data. We used the popular Porter stemming algorithm in our analysis to remove the common endings of words. For example, stemming reduces words such as *fishing, fished, fish, fisher,* and *fishes* to the stem word, *fish*. The stemming procedure reduces the number of words to consider and provides a better frequency representation in the DTM as a result.

Stopwords are defined as words in a language that are so common that their information value is practically null. Some common stopwords are words such as *a, and, but, the,* and *if*, among many others. It is common practice in text analysis to reduce the number of 'noisy' words in the data by removing stopwords. We relied on the SMART information retrieval system data base to remove stopwords in our reviews [4].

As a final step we reduced the sparsity of the DTM by removing uncommon words. There were many words that were very rare, and others that were grossly misspelled. These words appeared in only a handful of documents and offered little information value, so to reduce the number of noisy variables in our data further, we removed terms that appeared in less than 3% of the documents. The final DTM contained 288 words of interest.

2.1.2 The Review Text Features

It has been reported that a weighted term frequency DTM provides more information, as opposed to just a term frequency DTM. We used the Term Frequency-Inverse Document Frequency (TF-IDF)[5]

weighting to provide a metric for the importance of a word in our DTM. As its name suggests, TF-IDF is the product of the normalized Term Frequency (TF) and the Inverse Document Frequency (IFT) statistics, where

$$\text{TF}(t,d) = \frac{\text{frequency of term } t \text{ in document } d}{\max\{\text{frequency of term } w \text{ in document } d : w \in d\}}$$

$$\text{IDF}(t,D) = \log\left(\frac{\text{total number of documents in } D}{1 + \text{number of documents where term } t \text{ appears in } D}\right)$$

$$\text{TF-IDF}(t,d,D) = \text{TF}(t,d) \times \text{IDF}(t,D)$$

In our data, t is word in our DTM and d is a single review in the set of all reviews, D. A high TF-IDF value is obtained when a term appears frequently in a given review but appears rarely in the collection of all reviews. When a term appears in many reviews, the IDF function will approach 0. Therefore, we can view the TF-IDF weights as filtering out common words.

2.1.3 Other Features

In addition to the features created from the review texts, we had access to supplementary data connecting the reviews to the businesses, users, and check-ins data. We added the useful features from these data sets to the DTM, and created some new features as well. These features are summarized in Table 1. We also centered and scaled the continuous variables so that the features would all be in similar and comparable scales.

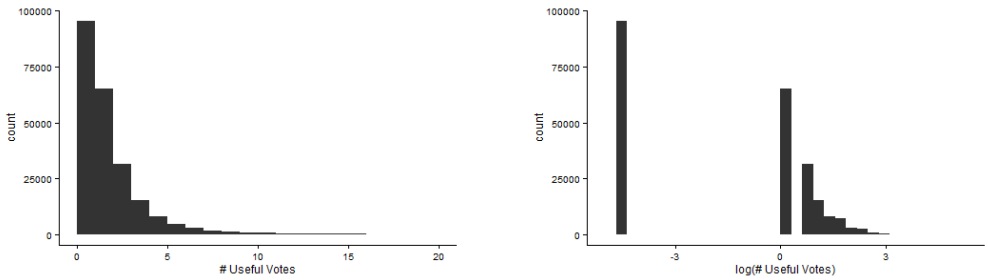
Table 1: The features extracted from the Yelp data sets.

DATA SET	FEATURES	DESCRIPTION
Review	Star Rating	Star rating associated with review (from 0 to 5)
	Date	Days until February 26, 2013
	No. words	Number of words in review
	No. lines	Number of lines in review
	Happy	Presence of a happy emoticon
	Sad	Presence of a sad emoticon
Business	Longitude	Longitude of business
	Latitude	Latitude of business
	Average Stars	Average star rating of business
	No. reviews	Number of reviews received
	Open	Is the business still operating
User	No. reviews	Total number of reviews written by user
	Average Stars	Average star rating assigned by user in all reviews
Check-ins	Weekday early morning	Check-ins between 12AM-7AM
	Weekday morning	Check-ins between 7AM-11AM
	Weekday midday	Check-ins between 11AM-2PM
	Weekday afternoon	Check-ins between 2PM-5PM
	Weekday evening	Check-ins between 5PM-9PM
	Weekday night	Check-ins between 9PM-12AM
	Weekend early morning	Check-ins between 12AM-7AM
	Weekend morning	Check-ins between 7AM-11AM
	Weekend midday	Check-ins between 11AM-2PM
	Weekend afternoon	Check-ins between 2PM-5PM
	Weekend evening	Check-ins between 5PM-9PM
	Weekend night	Check-ins between 9PM-12AM

2.2 Modeling the Response Variable

The variable of interest, 'Useful Votes', can be thought of as count data, taking on positive integer values. The mean number of votes in the data set was 1.387, and the data was heavily skewed to the right. Out of the 229,907 reviews, 41.5% of the reviews received 0 votes, 28.4% received 1 vote, 13.7% received 2 votes, 6.68% received 3 votes, and 9.75% received 4 or more votes. A histogram

of the response variable is displayed in Figure 1a. It should be noted that there are 39 reviews that receiving more than 30 votes, with one review receiving 120 votes. These entries were omitted in the figure in order to produce a readable figure.



(a) The discrete response variable.

(b) Log transformed response variable.

Figure 1: Histograms showing the distribution of the number of 'Useful' votes in the Yelp data set. There were 229,907 reviews in the data set. There were 39 reviews with more than 30 votes (not shown in a).

If we were to naively fit a regression model on the count data, we may end up with negative predictions, which do not make sense. A convenient transformation for discrete count data is to model the log of the response variable instead. Since the logarithm of 0 is negative infinity, we can add a small $\epsilon = 0.01$ to the count data to avoid this problem. The histogram of the log response is shown in Figure 1b.

We tried fitting the non-parametric Random Forest regression model and the L1 penalized linear regression model (LASSO) on both the response variable and the log of the response. We used the two models to conduct feature selection, and then tried fitting fully parametric regression models on the discrete response variable using Negative Binomial (NB) regression and Zero Inflated Negative Binomial (ZINB) regression models. These models are discussed below.

2.2.1 Random Forest Regression

Random Forest [6] is a popular machine learning algorithm that can perform both classification and regression, and it has been known to perform well on many different types of data sets. The basic idea of Random Forest is to train an ensemble of uncorrelated, weak learners with high variance on bootstrap samples of the data, and then output the average result.

We used the Python implementation of Random Forest in the package `scikit-learn`. We made a Random Fores of 100 trees, and tuned the maximum depth of the trees in the forest using 4-fold cross-validation to minimize prediction error (the exact error metric is discussed in Section 2.3) on the training set. The other parameters were kept at their default settings. A regression was fit on both the discrete response variable and the log of the response variable and their prediction performance was assessed on the validation set.

Random Forest also provides a variable importance measure. In the implementation by `scikit-learn`, variable importance is measured by the expected fraction of the splits that a feature contributes to. In other words, the number of times a feature is used to provide an optimal split in a node of a decision tree in the forest can provide an estimate of the relative importance of a feature. The relative importance of each feature is normalized to sum to 1. The variable importance measures from the final model were used to train the NB and ZINB models.

2.2.2 LASSO Regression

The LASSO [7] is a linear regression method that uses an L1 penalty to control the complexity of the model. Similar to Ridge regression, it is a shrinkage operator that causes the coefficients in the linear model to become smaller, but due to the nature of the L1 constraint, LASSO causes some coefficients to be shrunk to 0. Because of this result, we can perform feature selection by using the subset of the features with non-zero coefficients in the final LASSO model.

We use the LASSO implementation found in `scikit-learn` Python package. Like the Random Forest model, we tune the regularization parameter using 4 fold cross-validation on the training set to minimize prediction error. The model was fit on both the discrete response variable and the log of the response variable. The features with non-zero coefficients in the final LASSO model were used to guide the training of the NB and ZINB models.

2.2.3 Negative Binomial Regression

The aforementioned Random Forest regression and LASSO regression models assume that the response variable is unbounded. However, the number of 'Useful' votes is a positive discrete variable, and so we made an effort to correct for this by modeling the logarithm of the votes as the response.

We now consider assuming a distribution on the response variable and fitting a linear model on the mean response. This is referred to as a generalized linear model (GLM) [8], where we assume that the expected value of the response variable follows a linear model after being transformed by a link function. The reader may be aware of logistic regression, one of the more commonly used GLMs. Logistic regression is used when the response variable is binary and is assumed to follow a Bournoulli distribution. The mean parameter is then assumed to follow a linear model after being transformed by the logit link function. More formally, in the logistic regression, we assume that $y_i|\mathbf{X} \sim \text{Bern}(p_i)$ and

$$\text{logit}(p_i) = \log\left(\frac{p_i}{1-p_i}\right) = \mathbf{X}\beta$$
$$p_i = \frac{1}{1+e^{-\mathbf{X}\beta}}$$

where $p_i = \Pr(y_i = 1|\mathbf{X}) = \mathbf{E}(y_i|\mathbf{X})$, and the logit link function is what constrains the linear model to make predictions between 0 and 1.

Now in negative binomial regression [9], we assume that the response variable is discrete and positive (i.e. is count data), and we assume that it follows a negative binomial distribution. We can then use the log link function to model the mean response variable with a linear model. More formally, we have $y_i|\mathbf{X} \sim NegBin(\mu_i, \kappa)$, where μ_i is the mean and κ is the dispersion parameter that controls the shape (i.e. the variance) of the distribution. We can model the expected value with

$$\log(\mathbf{E}(y_i|\mathbf{X})) = \log(\mu_i) = \mathbf{X}\beta$$
$$\mu_i = \exp(\mathbf{X}\beta)$$

The coefficient parameters β can then be estimated via maximum likelihood methods (via iteratively reweighted least squares algorithm, for example). For prediction on count data, NB regression has been shown to be comparable to neural networks in some situations [10]. We fit this fully parameterized NB regression using the `glm` package in `R`. We also tried fitting the NB regression with the variables selected by Random Forest and LASSO, and then tried another feature selection step by using features that were deemed to be statistically significant by the Z-test.

2.2.4 Zero Inflated Negative Binomial Regression

In some instances, we may observe more zero counts than can be explained by the negative binomial model. As can be seen in Figure 1a, the number of zero 'Useful' votes in the Yelp review data is very high, accounting for over 40% of the responses. This problem can be amended by fitting a Zero Inflated Negative Binomial model [9], where it is assumed that the response y_i is generated from a mixture of two distributions: (1) responses that are zero with probability one, and (2) responses that follow the negative binomial model. In regards to our Yelp review data, we may have some reviews with that have zero votes with probability one because the reviews are truly useless, and then we may have other reviews that follow the negative binomial model.

Now two distinct sets of features can be used to model the ZINB regression. Features can be used in the Zero Inflating part of the model, which can be expressed as a logit model for predicting the latent variable of whether a review is useful or not. Another set of features can enter in the NB model, which determines how many votes a useful review will receive.

We used the `pscl` package in `R` to fit the ZINB model.The features that were selected by Random Forest and LASSO were used to fit both the ZI and the NB parts of the model. Afterwards, we

tried to fit a reduced model using the statistically significant features according to the Z-test, and the results were compared our other fitted regression models.

2.3 Evaluation of Performance

We use root mean squared log error (RMSLE) on the validation set to evaluate the performance of our models:

$$\text{RMSLE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(p_i+1) - \log(a_i+1))^2}$$

where n is the number of reviews in the validation set, p_i is the predicted number of useful votes for review i, a_i is the actual number of useful votes for review i, and $\log(x)$ is the natural log of x.

This error function places more importance on minimizing the error for reviews that have a smaller number of actual 'Useful' votes. For example, an absolute error of 1 on a review with 3 'Useful' votes will be worse than an absolute error of 1 on a review with 100 votes.

3 Results and Discussion

3.1 Tuning Random Forest and LASSO

We assessed the optimum tree depth of the Random Forest Regression using 4 fold cross-validation on the training set. We considered using trees with maximum depths of $\{1, 10, 20, 30, 50, \infty\}$. It was found that the optimum depth of the trees in a Random Forest was 20 - any more resulted in overfitting on the training set, and any less resulted in poor predictive performance. Similarly, the regularization parameter for LASSO was chosen using 4 fold cross-validation on the training set.

3.2 Benefits of Term Weighting and Fitting on the Log Response

We begin by comparing the best results we obtained from fitting the Random Forest and LASSO models (as described in Section 2 and 3.1) using the Term Frequency text features and the TF-IDF weighted text features. We also report the observed effect of fitting the models on the log of the response versus the discrete response. The results are summarized in Table 2.

Table 2: The RMSLE values from predicting on the validation set using models trained on the training set. Note that the features mentioned Section 2.1.3 were included in all data sets. λ refers to the regularization parameter for LASSO.

MODEL	TEXT FEATURES	RESPONSE TYPE	RMSLE	NOTES
Random Forest	Term Frequency	Votes	0.5834	100 trees,
	Term Frequency	log(Votes)	0.5522	max depth 20
	TF-IDF	Votes	0.5819	
	TF-IDF	log(Votes)	**0.5457**	
LASSO	Term Frequency	Votes	0.5447	λ=0.0045
	Term Frequency	log(Votes)	0.6384	λ=0.6384
	TF-IDF	Votes	**0.5313***	λ=0.0670
	TF-IDF	log(Votes)	0.6020	λ=0.5573

It seems that using the TF-IDF weighted text features consistently provides better predictions with regards to the RMSLE error metric in both Random Forest and LASSO, but not by much. We note that the TF-IDF weighting may have been more useful if we had kept more words in the DTM.

On the other hand, taking the log of the response before training the Random Forest yielded noticeably better results, while LASSO seemed to perform better on the untransformed response. We are not exactly sure why this is the case, but we suspect that taking the log transform of the response variable may make the relationship between the response and the covariates highly non-linear, resulting in lower predictive performance for LASSO. The best results were achieved using LASSO on the TF-IDF weighted text features and the discrete response variable.

3.3 NB and ZINB Regression Results

We used the features selected by Random Forest and LASSO to fit the NB and ZINB models (see Table 3). Interestingly, Random Forests found more text features important, while LASSO found more supplementary features important. An important feature for both models was the length of the review (num_words, num_lines).

For the ZINB model, the selected features were used for both the Zero Inflating part and the NB part of the model. In addition, if a feature was found to be statistically insignificant, we tried dropping the feature from the model. However, we found that dropping feature did not improve the RMSLE measure on the validation set.

Table 3: Selected features for negative binomial regression

Features Selected by Random Forest			**Features Selected by LASSO**		
Text		Other Features	Text	Other Features	
reason	return	user_review_count	locat	user_review_count	num_words
attent	hit	num_words	beer	date	num_lines
chicken	servic	date	area	WD_morn	WD_aftnoon
special	leav	num_lines	food	biz_review_count	WD_night
taco	steak	user_avg_stars	place	biz_avg_stars	user_avg_stars
fresh	option	longitude	pretti	WE_aftnoon	open
includ	hour	review_stars	pizza	WE_morn	latitude
call	good	latitude_biz	great	WD_eve	review_stars
owner	arriv	biz_review_count	nice	longitude	WE_mid
half	mix		good	WE_eve	

We found that the ZINB regression model performed slightly better on the validation set when compared to the NB regression model (see Table 4). However, both the fitted ZINB and NB models performed worse than the LASSO and Random Forest Regression models. It seems that the distribution assumption on the count data did not yield any advantages over the non-parametric Random Forest and the linear LASSO model with the features that we selected.

Table 4: The RMSLE values from predicting on the validation set using models trained on the training set.

MODEL	**RESPONSE TYPE**	**BEST RMSLE**
LASSO	Votes	0.5313
Random Forest	log(Votes)	0.5457
ZINB Regression	Votes	0.5787
NB Regression	Votes	0.5856

3.4 Kaggle Contest Results

We trained the LASSO, Random Forest, NB and ZINB models on the entirety of the training and validation sets using the best parameters we found in our experiments and submitted our predictions of the test set to the Kaggle competition. Our LASSO performed the best out of our submitted models, placing us in 20^{th} place on the leaderboard out of 89 participants as of April 16th. Surprisingly, Random Forests performed quite poorly on the test set, indicating that our fit may not have been very robust. NB regression performed nearly as well as LASSO, and much better than Random Forest. Unfortunately, due to the submission limit, we were not able to submit our predictions from the ZINB model before the project deadline. We would predict that its performance falls between the LASSO and NB Regression results. The test set errors are summarized in Table 5.

4 Future Work and Conclusions

Having better features will almost always guarantee better results. We took a very simplistic approach and extracted unigram features from the text data, but there are many more methods of extracting information from texts that we did not explore. We believe that we could improve our results if we use a sparser DTM with more terms, look at n-gram text features (e.g. consider pairs or

Table 5: Current Kaggle standings, as of April 16, 2013. The RMSLE is calculated by Kaggle on an unlabeled test set consisting of 22,956 reviews.

MODEL	**RMSLE**
Current Leader	0.44808
LASSO	0.53765
NB Regression	0.55022
Random Forest	0.68775
Global Mean Benchmark	0.72327
All Zeros Benchmark	0.72745
ZINB Regression	N/A

triples of words), perform synonym matching, and do part-of-speech tagging. It would be interesting to apply our models with better features using natural language processing.

Also, we only considered the vanilla implementations of Random Forest and LASSO provided by `scikit-learn`. We may have observed better results had we customized the algorithms for our specific task. For example, we wonder if we would have seen an improvement in the Random Forest results if we had changed the criterion for spliting nodes to one suitable for count data, or if we had fit Poisson Regression or Negative Binomial regression models at the nodes of decision trees.

In addition, we relied heavily on Random Forests and LASSO to select features for fitting the Negative Binomial regression models. It may have been worthwhile to explore other feature selection methods, such as forward selection and backward selection using statistical tests such as the likelihood ratio test to compare models. Maybe Bayesian Optimization could be applied in the feature selection process, or tuning of our models. Also, it would have been worthwhile to investigate the Zero Inflating process more carefully in the ZINB models.

In conclusion, we have found some success at predicting the number of 'Useful' votes a Yelp review will receive using LASSO and Random Forest Regression, two well known tools in the machine learning toolkit. In addition, we introduced the Negative Binomial and Zero Inflated Negative Binomial regression models as potential alternatives for modeling relationships in count data, and have shown that they are somewhat comparable to LASSO and Random Forests. In the long run, it may be worthwhile to investigate parametric models such as the NB and ZINB models, as they may provide a framework that can help us understand how the data was generated.

References

[1] O'Mahony, M. P., Cunningham, P., & Smyth, B. (2010). An assessment of machine learning techniques for review recommendation. *In Artificial Intelligence and Cognitive Science* (pp. 241-250). Springer Berlin Heidelberg.

[2] Feinerer, I., Hornik, K., & Meyer, D. (2008). Text mining infrastructure in R. *Journal of Statistical Software*, 25(5), 1-54.

[3] Porter, M. F. (1980). An algorithm for suffix stripping. *Program: electronic library and information systems*, 14(3), 130-137.

[4] Salton, G. (1971) *The SMART Retrieval System–Experiments in Automatic Documnet Processing.* Upper Saddle River: Prentice-Hall.

[5] Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5), 513-523.

[6] Breiman, L. (2001). Random forests. *Machine learning*, 45(1), 5-32.

[7] Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society*. Series B (Methodological), 267-288.

[8] Nelder, J. A., & Wedderburn, R. W. (1972). Generalized linear models. *Journal of the Royal Statistical Society*. Series A (General), 370-384.

[9] Hilbe, J. M. (2011). Negative binomial regression. Cambridge University Press.

[10] Chang, L. Y. (2005). Analysis of freeway accident frequencies: negative binomial regression versus artificial neural network. *Safety science*, 43(8), 541-557.