Inferring Future Business Attention

Bryan Hood, Victor Hwang and Jennifer King

Yelp collects many different forms of data about a particular business - user reviews, ratings, location and more. However, it would be valuable to businesses to get a glimpse of their future Yelp reviews - how many reviews will they get in the next month? In this paper, we describe methods for inferring future business attention using regression models and sentiment analysis of reviews.

■

1. PROBLEM OVERVIEW

Yelp kicked off in 2005 as a way for users to rate and review local businesses. Businesses organize their own listings while users rate the business from 1-5 stars and write their own text reviews. Within this system, there is a meta review system where users can vote on helpful or funny reviews.

After eight years of operation, Yelp has amassed an enormous amount of raw data on businesses. While businesses are able to see their current ratings and the raw text of their reviews, there is no information relating to forecasts about their business - are they going to become more popular? Will their business increase? With this knowledge, a business owner may determine that something is amiss with their business model, or that a new product they added is a really big hit.

In this work, we describe our efforts to process this raw information and determine the amount of attention a business receives. In order to avoid making assumptions about what constitutes a successful business, we will instead look at predicting the number of reviews a particular business has at different points in time. Specifically, we predict the number of reviews that a particular business should have at the current time, and how many reviews it should have in the next six months. The former tells us if the attention a business currently has matches what the Yelp market expects. The latter describes the trend that their business is following - will their attention increase dramatically in the upcoming days?

We use a Yelp-provided data set to try and learn features related to a high attention business. This data is limited to businesses, users, and review text in Phoenix, Arizona. The provided raw data is clean, but does not offer many features to work with.

The majority of our work involves generating sets of features to use in our model. We describe two methods: simple manipulation of the given data and sentiment analysis on user-provided reviews. We then run feature selection methods to try and pick out the best features. In our experimental results section, we describe which features were helpful and how our predictor performs compared to a simple regression model.

2. DATA DESCRIPTION

2.1 11537 Business records

```
'type': 'business',
'business_id': (encrypted business id),
'name': (business name),
'neighborhoods': [(hood names)],
'full_address': (localized address),
'city': (city),
'state': (state),
'latitude': latitude,
'longitude': longitude,
'stars': (star rating, rounded to half-stars),
'review_count': review count,
'categories': [(localized category names)]
'open': True / False
```

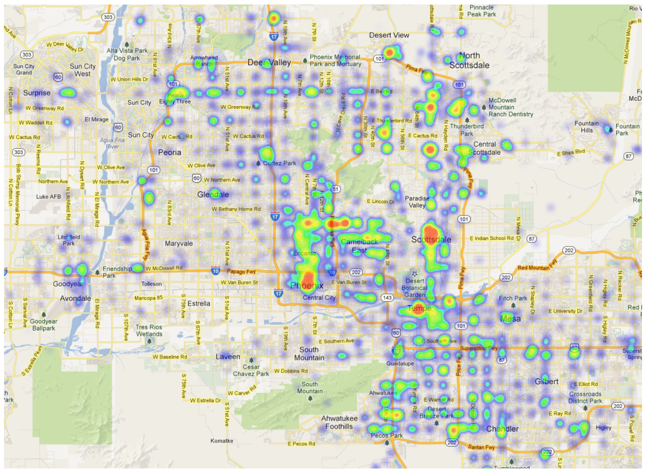

Fig. 1. Heatmap showing the number of reviews in Phoenix, Arizona. Red spots depict areas with a large number of overall reviews.

2.2 229908 Reviews

```
'type': 'review',
'business_id': (encrypted business id),
'user_id': (encrypted user id),
'stars': (star rating, rounded to half-stars),
'text': (review text),
'date': (date, formatted like '2012-03-14'),
'votes': {(vote type): (count)}
```

2.3 43873 User Records

```
'type': 'user',
'user_id': (encrypted user id),
'name': (first name),
'review_count': (review count),
'average_stars': (floating point average, like 4.31),
'votes': {(vote type): (count)}
```

A visualization of the data in Figure 1 shows clusters of businesses with large number of reviews. Presumably these are high traffic areas.

3. GENERATED FEATURES

In this section, we describe two methods used to generate additional features. We begin with simple manipulation of the data to create time-dependent features. Then, we describe how opinions are extracted from the raw review text and casted into a usable feature vector.

3.1 Time Dependent Features

We identify two categories of features. The first category contains features which describe metadata about the business. Examples of such features are location (latitude and lontigude), number of businesses within 1km and number of businesses sharing a category.

The second category contains features which describe a subset of reviews. The following list describes such features:

(1) Number of reviews in the set
(2) Average number of stars across reviews in the set
(3) Maximum number of stars
(4) Mininum number of stars
(5) Number of reviews voted as 'cool'
(6) Number of reviews voted as 'funny'
(7) Number of reviews voted as 'useful'
(8) Number of unique users logging these reviews
(9) Number of days since the first review in the set (time since first attention)
(10) Number of days since the last review in the set (downtime since last attention)
(11) Number of days between first and last review in set (duration of attention)
(12) Max number of reviews in the set for a single day

The problem of generating feature vectors then reduces to generating subsets of reviews which could be be helpful in predicting attention for a particular business, then generating the above feature vector for each subset.

One such subset is the set of all reviews the business has received so far. This subset can be particularly helpful for answering the question: Given the amount of attention a business has received so far, can we predict future attention?

We also expect that reviews of businesses that share a category with our business can help predict future attention for the business. In particular, we believe this information will be helpful when predicting future attention for a young business (i.e. a recently opened business). To capture this idea we generate a subset that contains all reviews for these businesses.

Finally, we expect features describing the location of the business to be useful for calculating attention. To this end, we generate a subset that contains all reviews for any business located within 1km of the target business.

3.2 Generating Review Text Features

Raw review text can be processed and used in a variety of ways for a regression model - simple n-gram analysis, keyword associations and extractions, and sentiment analysis. In our application, we used the sentiment analysis described in [Hu and Liu 2004]. Our aim was to mine the most frequently occuring keywords among all restaurant reviews and use counts of the sentiments for each of these keywords as features. The assumption is that restaurants with many positive statements about a particularly popular keyword (for Phoenix, the most popular food-related keyword was pizza) would have high reviews.

With this goal, the following algorithm produces a feature vector containing the the counts of the number of positive and negative statements about the top 300 keywords. This is done in two steps - first, we compute the top keywords among all the reviews. Then, for each review, we count the number of statements critiquing those top keywords.

3.2.1 *Review Text Parsing.* Generating features from the review text begins with cleaning up the text (spell checking, removing excessive punctuation, etc). Then, a series of tokenizations occur - we tokenize each review into sentences, then tokenize the sentences into words. Using the Python Natural Language Toolkit (NLTK), we then part-of-speech tag each token in the sentence. The result below shows a sentence ("Our dishes of Pad See ewe and Dragon noodles were great, and huge portions.") being parsed using the Penn TreeBank corpus.

```
[(u'Our', 'PRP$'), (u'dishes', 'NNS'),
(u'of', 'IN'), (u'Pad', 'NNP'),
(u'See', 'NNP'), (u'ewe', 'NN'),
(u'and', 'CC'), (u'Dragon', 'NNP'),
(u'noodles', 'NNS'), (u'were', 'VBD'),
(u'great', 'JJ'), (u',', ','), (u'and', 'CC'),
(u'huge', 'JJ'), (u'portions', 'NNS'),
(u'.', '.')]
```

For our features, care only about adjectives (labeled 'JJ') and nouns (labels starting with 'NN'). In order to avoid casting a conditional independence assumption on the words, we must do text chunking to create relevant noun phrases.

Noun phrases (or noun groups) are groups of words (phrases) that are all associated with one particular noun. Typical examples of a properly chunked phrase might be "Five shiny red apples". In this application, we are not concerned with grouping adjectives with their associated nouns. Instead, we use a grammar that only chunk nouns that appear together. In the above case, we would end up grouping together "Pad See ewe" and "Dragon noodles" into two usable noun phrase. This allows us to avoid making the strong conditional independence assumption as Naive Bayes.

We then run the above computation over all sentences for all businesses in a related category (in our case, restaurants) while keeping counters of all noun phrases.

At the end, we take a subset of the keywords (we used 100) that appeared among all the reviews and use this to generate feature vectors given a single review. It should be noted that this method works the best over a single category of business - running it across many different categories would end up adding many unrelated keywords to the final set of keywords.

3.2.2 *Keyword-Opinion Extraction.* Now that we have a representative set of keywords used to describe restaurants, we count the number of positive and negative opinions about each keyword in order to generate a feature vector about a specific review.

We begin by first generating sets of positive and negative adjectives. As described in [Miller et al. 1990], we use Wordnet to quickly generate these sets. Wordnet is a lexical database for the English language and groups words together into sets of synonyms and antonyms. Starting with a small list of positive seed adjectives, we accumulate all synonymous adjectives to the seed and count those as positive words. By doing the same thing with negative adjectives, we generate a list of adjectives used to consider an opinion about a noun phrase.

Now with a list of positive and negative adjectives and a list of keywords, we analyze each sentence in each review by counting the number of positive and negative adjectives associated with the keywords. For the sentence "The Pad See ew was great", "great" would be associated with "Pad See ew" (assuming this was a keyword), and this sentence would have one positive count for the keyword ”Pad See ew”. For sentences with multiple noun phrases and adjectives in them, we associate the adjective with the closest noun phrase.

By computing this over all reviews for a specific business, we hope to extract the good and bad features of a business. Presumably, if the restaurant has the best Pad See ew in the city, we would see many positive comments about that feature in the reviews.

Below shows a slice of positive and negative opinion counts on a few features.

```
# Positive statements:
(u'bread', 1), (u'breakfast', 0), (u'buffet', 0),
(u'burger', 1), (u'burrito', 0), (u'business', 0),
(u'butter', 12)

# Negative statements:
(u'bread', 0), (u'breakfast', 0), (u'buffet', 0),
(u'burger', 0), (u'burrito', 0), (u'business', 0),
(u'butter', 3)
```

3.3 User Clustering

For predicting the number of reviews a business has and its rating, knowing something about the user base is important. The number of users is quite large, so a tractable way to deal with users is to cluster them. Using the list of user features below, we clustered the users using k-means.

(1) Number of reviews
(2) Average number of stars
(3) Number of “Funny” votes
(4) Number of “Useful” votes
(5) Number of “Cool” votes
(6) Date of first review
(7) Date of last review
(8) Days between first and last review

Clusters were intepreted by their center means into labels that humans can understand. These labels were basic, happy, long timers, angry, early adopters and power users. A couple of the clusters received the same human readable label. Basic users had on average close to 30 reviews each, and had not been active for very long. Happy and angry users had only a review or two each, rated a business very well or very low and then never wrote another review. Long time users wrote about 50 reviews on average, and had been active for close to 3 years. Early adopters had similarly written about 50 reviews, wrote their first review before shortly after yelp launched, and were likely to have stopped reviewing. Finally, power users had written over 1000 reviews, and had amassed several thousand “useful”, “funny“, and “cool“ votes. Power users are also more likely to have quit writing reviews than basic users.

4. FEATURE SELECTION AND ANALYSIS

4.1 PCA

Principal component analysis (PCA) is a tool that reduces dimensionality of a feature space. It is particularly useful when the feature space is too large for a regression or classification to be computationally feasible or when data contains large amounts of noise. In order to reduce the dimensionality, PCA selects orthogonal vectors in the feature space that maximize variance. This way, the features are compressed and a few principal component vectors can describe a large portion of the feature space. The downside to using PCA is that there is loss of information in the compression and there is no guarantee that the principal components will be separable.

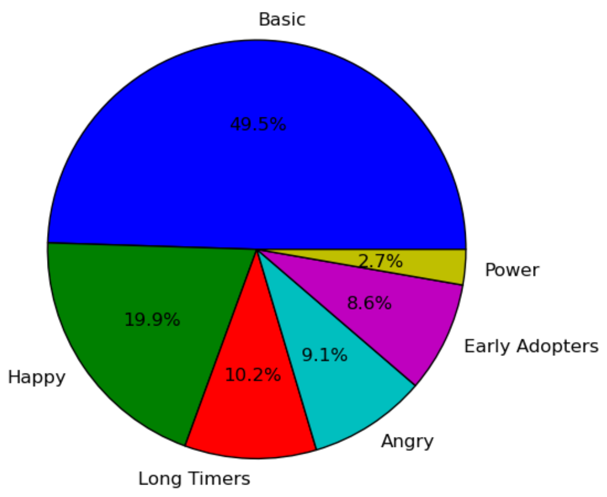

Fig. 2. Piechart showing the percent of users in each cluster found by k-means. Clusters were intepreted with a human understandable label.

4.2 Feature Selection

Often, engineered and extracted features will sometimes contain redundant or irrelevant information. Adding too many of these poor features can inhibit the performance of a learning algorithm because there is a lot of extra noise in the data. The best features are those that make data easily separable or predictable. A naive approach to feature selection uses an exhaustive search over all possible subsets of features to pick the subset with the smallest test error. For large datasets with many features, this method is not feasible. We will look at three other types of feature selection, using K-Fold cross-validation to minimize test error.

4.2.1 *Univariate Feature Analysis.* A simple and efficient way of selecting features is to look at the prediction quality of each feature individually. The best n of these features are then used for the prediction model. The major advantages of this method are that it is fast and simple. However, a major drawback is that combinations of bad individual features can sometimes interact to create a good prediction model.

4.2.2 *Greedy Feature Removal.* In greedy feature removal, the worst performing feature is removed at each iteration until a certain performance is achieved or until there are a specified number of remaining features left. This process can be computational expensive, but it usually gives better results than analyzing features independently. When a linear model is used, there are methods for evaluating each feature once and using the residual model error to remove each additional feature.

Because greedy feature removal is expensive, we propose to use a K-fold removal technique. In this, we will segment the data into K random divisions. We will then remove each of the K divisions and evaluate the success of the model fit. The worst will be removed. This process will be repeated until we have a desired success rate or a fixed number of features threshold has been met.

4.3 Scaling

For numerical stability reasons and proper comparison between features, we scale our data. We chose to do mean removal and variance scaling.

Given k feature vectors of n features in the training set, the mean and standard deviation of each feature is computed.

$$\mu^i = \frac{1}{k}\sum_{j=0}^{k} x_j^i$$

$$\sigma^i = \sqrt{\frac{1}{k}\sum_{j=0}^{k}(\mu^i - x_j^i)^2}$$

Both the training and test data sets are then scaled using the scaling factors generated for the training set.

$$x^i_{scaled} = \frac{x^i - \mu^i}{\sigma x^i}$$

5. PREDICTION

We chose to use Support Vector Regression (SVR) as our regression model in an effort to keep scalability in mind. One can imagine that an implementation of our algorithm shows up as a statistic for every business. Training (an expensive task) can be done perhaps once a day. Every time a business owner logs into their listing, they get an up-to-date forecast about their business. As a result, predictions will occur more than training. Because SVRs provide very fast prediction performance, this will likely hold up on running numerous predictions once the data sets become bigger. However, because our test set only covers one city, we were never able to test the limits of prediction performance.

SVRs are able to run fast predictions because of their similarity to Support Vector Machines (SVM). It aims to find a function $f(x)$ that does not deviate more than some ϵ away from the training data. By having the ϵ margin, the training process can determine a subset of training points as having the most influence on the model parameters. The points with the most influence end up defining the margin and are called support vectors. Thus, the final model that is produced depends only on a subset of the training data. Predictions then run only with these smaller number of support vector. More information on SVR can be found in [Smola and Schölkopf 2004]

6. EXPERIMENTAL SETUP

We execute a series of experiments to validate the regression models. In each experiment, we use K-fold cross validation with 10 folds. We use two metrics to evaluate performance. For the first we calculate error percentage across predictions.

$$error = \frac{actual - predicted}{actual}$$

Our second metric evaluates the performance of our model against a naive model. In the case of temporal prediction, this would be a model which predicts the average number of reviews the businesses in the training set received during the 6 month time window.

$$errorratio = \frac{predicted - actual}{average - actual}$$

A ratio larger than 1 indicates better performance by the naive model. A ratio smaller than 1 indicates better performance by our model.

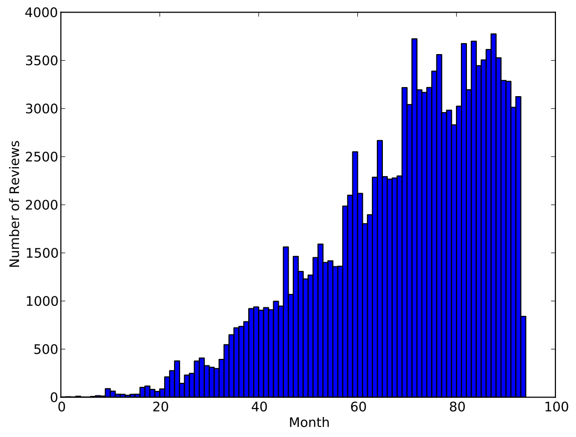

Fig. 3. Plot showing the number of reviews logged for all restaurants during each month.

It should be noted that this metric is only useful in determining useful features, but is not a comprehensive measure on the performance of our predictor. This will be described more in detail in a later section.

The SK-Learn library ([Pedregosa et al. 2011]) was used in all experiments.

7. RESULTS

7.1 Temporal Prediction

In this section we describe results for a classifier which can predict the expected future attention of a business given the attention received so far. For these results, we limit ourselves to only consider businesses which list 'Restaurants' as a category. This gives us a set of 3901 businesses. Figure 3 shows the total number of reviews received per month for all restaurants.

For each of the experiments reported on in this question we attempt to build a model which can be used for the following: **Given metadata about a business and all reviews in the Yelp database logged before a target date, predict the number of reviews that will be received for that business during the 6 month time period starting from the target date.**

We present results for four distinct set of experiments:

(1) *Model performance using simple features*
(2) *Model performance using review text features*
(3) *Model performance against simple regression*
(4) *Performance of models as target date varies through time*

7.1.1 *Model performance using simple features.* For our analysis in this section, we select a single date and use this as our target date. In other words, we select a single date and use all reviews up until that date to attempt to build our regression model. The date we will select is 2011-02-01 (month 70). This date is in the middle of our dataset and provides a large number of reviews logged before the date, while also having a significant number of reviews logged in the 6 months following the date.

Using the method described in section 3.1 we first generate a vector of 40 features highlighting metadata about each business.

Table I. Comparison of error rates

Features	Error %
Naive Model	86.8
All Features	64.3
PCA	67.6
Univariate Feature Analysis	54.9
Greedy Feature Removal	56.7

Table II. Comparison to Naive Model

Features	Error Ratio
All Features	0.740
PCA	0.779
Univariate Feature Analysis	0.632
Greedy Feature Removal	0.654

We build feature vectors for each of the 3158 restaurants which have received at least one review before our selected date.

For our initial test, we build a model using all 40 features. The results of this produce a model that exhibits an error percentage of 64.3%. When compared to the naive model we see an error ratio of 0.740. This indicates our model outperforms the naive model.

In an attempt to improve our results, we perform feature analysis to reduce the dimensionality and understand the important features. We use three different analysis methods:

(1) *PCA:* First we examine the use of PCA (see Section 4.1) to perform dimensionality reduction in hopes of filtering out some noise in the data. Here we choose to keep a number of components equal to 25% of our feature vector length (10 components). The following shows the variances of only the first five components:

$$\begin{pmatrix} 9.69 & 6.61 & 4.44 & 2.53 & 1.92 \end{pmatrix}$$

When we examine the results of the model built by projecting to PCA's new feature space we see an error percentage of 67.6%. When compared to the naive model we see an error ratio of 0.779, indicating our algorithm outperforms the naive model.

(2) *Univariate Feature Analysis:* Next we examine the use of Univariate Feature Analysis (see Section 4.2.1). Here we attempt to select the best n features and use them to build our regression model. We choose to select the best 25% of features. The following list shows the features selected by the algorithm:
   (a) Number of reviews logged for the business before the target date
   (b) Max number of stars for any review logged for the business before the target date
   (c) Number of reviews logged for the business before the target date that are marked as 'cool'
   (d) Number of reviews logged for the business before the target date that are marked as 'funny'
   (e) Number of reviews logged for the business before the target date that are marked as 'useful'
   (f) Number of unique users logging reviews for the business before the target date
   (g) Number of days since the last review for the business logged before the target date
   (h) Number of days between the first review and the last review for the business logged before the target date
   (i) Max number of reviews for the business logged in a single day before the target date
   (j) Number of unique users logging reviews before the target date for any business within a 1km radius of the business

   This list indicates that features describing a business's received reviews are the strongest indictor of future attention. In addition, features describing reviews of businesses that share a category with the target business are unimportant. This is not particularly suprising. Only businesses with additional categories besides 'Restaurant' would have unique reviews in this set.

   Examining the results of the model built using this subset of features, we see an error percentage of 54.9%. When compared to the naive model we see an error ratio of 0.632, indicating our model performs better than the naive model. In addition, the use of this feature selection method shows improvement over the use of PCA.

(3) *Greedy Feature Removal:* Finally, we examine the use of Greedy Feature Removal for selecting features (see Section 4.2.2). Here we remove the worst performing feature at each iteration until only 25% of the features are remaining. The following list shows the features selected by the algorithm:
   (a) Latitude of the business
   (b) Number of reviews logged for the business before the target date
   (c) Number of reviews logged for the business before the target date that are marked as 'useful'
   (d) Number of days since the first review for the business
   (e) Max number of reviews for the business logged in a single day before the target date
   (f) Minimum number of stars logged for any review of a business in the same category
   (g) Number of days since the first review for any business in the same category
   (h) Number of reviews logged before the target date for any business within a 1km radius of the business
   (i) Number of reviews voted as 'cool' for all businesses within a 1km radius of the business
   (j) Number of days since the last review was logged for any business within a 1km radius of the business

   Examining the results of the model built using this subset of features, we see an error percentage of 56.7%. When compared to the naive model we see an error ratio of 0.654. Suprisingly, we see no improvement over the use of the Univariate Feature Analysis method.

Tables I and II show a summary of the results presented in this section. Figure 4 shows compares the predicted future attention using each method for 3 individual businesses.

7.1.2 *Model performance using review text features.* For the next set of results, we expand the feature vector to include the review text features described in Section 3.2. This increases our vectors to include 640 features. First we attempt to build a model using all 640 features. This gives fairly poor performance, showing an error percentage of 87.5% and an error ratio of 0.982 when compared to the naive model. As can be seen, the additional features seem to add some noise to the system, causing a drop in performance.

In attempt to improve results, and eliminate some of the introduced noise, we use the same methods listed above to examine reducing dimensionality or eliminating unimportant features.

(1) *PCA:* Again, we first examine the use of PCA to perform dimensionality reduction. Here we choose to keep a number of components equal to 10% of our feature vector length. Again

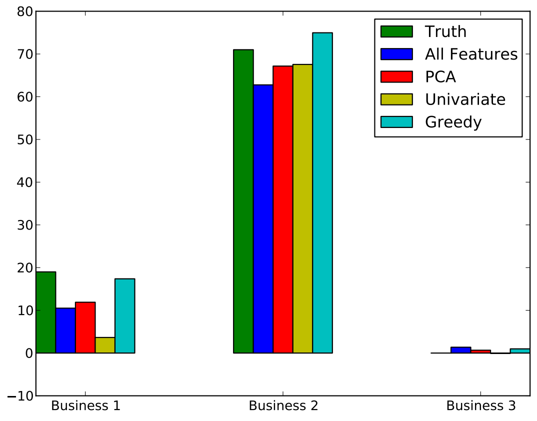

Fig. 4. Comparison of predicted future attention for 3 individual businesses.

we show only the variances for the top five components:

$$\begin{pmatrix} 64.75 & 8.96 & 8.64 & 7.78 & 6.91 \end{pmatrix}$$

Examining the results of the model build using this feature space, we see an error percentage of 84.2% and an error ration of 0.945 when compared with the naive model. This shows some improvement over the model built using all features, but overall performance remains weak.

(2) *Univariate Feature Analysis:* For the Univariate Feature Analysis, we choose to select only the best 10% of features. This will reduce the number of features used from 640 to 64. Examining the selected features we see only 6 of the original 40 features selected. They are:

(a) Number of reviews received for the business
(b) Number of 'cool' votes for reviews received for the business
(c) Number of 'funny' votes for reviews received for the business
(d) Number of 'useful' votes for reviews received for the business
(e) Number of unique users logging reviews for the business
(f) Max reviews in a day received for the business

The remaining 57 selected features describe positive and negative reference to various keywords. The model generated using this set of features exhibits an error of 83.4% and an error ratio of 0.936 when compared with the naive model. This model shows a slight performance over the naive model. In addition, it performs better than models built using all features or PCA.

Table III. Comparison of error rates when using review text features

Features	Error %
Naive Model	89.1
All Features	87.5
PCA	84.2
Univariate Feature Analysis	83.4

Tables III and IV provide an overview of the results presented in this section.

Table IV. Comparison to Naive Model when using review text features

Features	Error Ratio
All Features	0.982
PCA	0.945
Univariate Feature Analysis	0.936

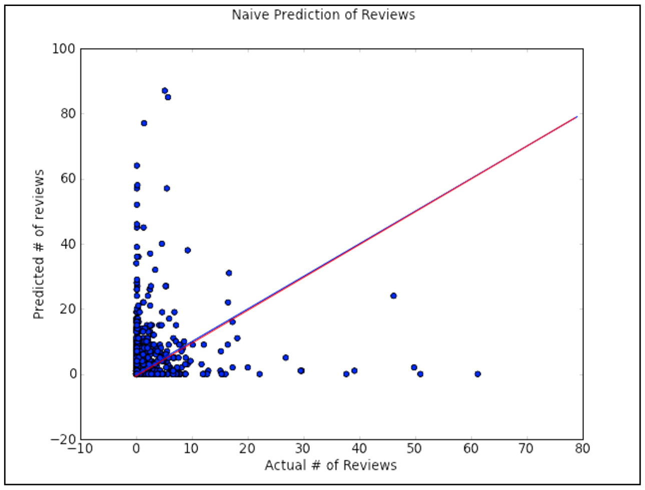

Fig. 5. A plot of the predicted vs. actual reviews for a naive model. This model performs regression on the rate of reviews received by a business and uses the result to project forward a prediction for the next 6 months.

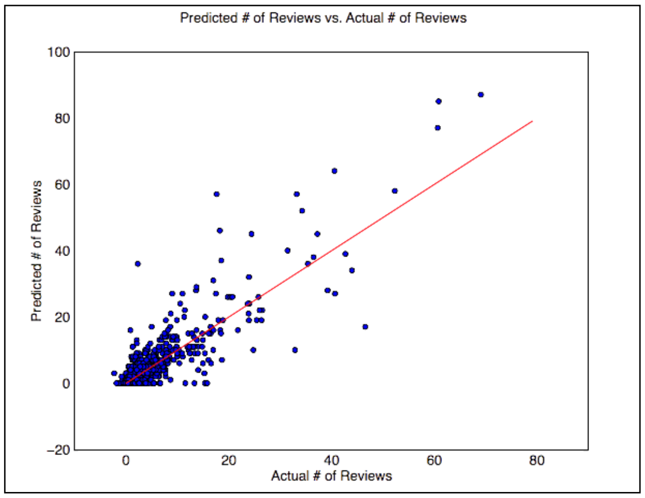

Fig. 6. A plot of the predicted vs. actual reviews with our feature set (using univariate feature selection) for the upcoming next 6 months.

7.2 Performance Comparison Against Naive Predictor

In the previous section, we optimize our feature set using a simple error metric. While effective for selecting features, the actual error values are very high. However, this is not a good metric for the performance of the predictor. One example in particular is that if a business only receives 1 additional review, but we predict 2, that yields a 100% error.

In this section, we show a comparison against a naive predictor. Here we use a model which predicts business attention over the next six months by performing regression on the reviews received for a given business in the past. In other words, this model attempts to fit a line to the number of reviews received by the business over time windows (we bin the number of reviews into 6 month windows), and then use that to project forward in time the number of reviews that will be received in the next 6 months. This seems reasonable from a glance, but due to the sporadic nature of reviews over time, fitting a polynomial does not work at all.

Instead, we again use SVR (with a linear kernel) with only the reviews received over time as our naive regression model. Figure 5 shows a plot of the predicted vs. actual values for this naive model over 500 businesses. Figure 6 shows a plot of the predicted vs. actual using our predictor with the same set of businesses. The red line in the figures show a perfect prediction, or where the actual number of reviews equals the predicted number of reviews.

This figure better depicts the performance of our predictor - visually, we see that our model follows the perfect prediction line much closer than the naive regression method. Also, it can easily be seen that most businesses receive under 20 additional reviews in the next six months, which is the major cause of our high error numbers in the previous section.

8. CONCLUSION

In this paper, we described a series of techniques used to infer future attention of businesses using numerous kinds of generated features. These features were generated from business statistics and raw review text.

In our experiments, we analyze all of these features to see their effects, and then attempt to infer the number of additional reviews businesses will receive in the next six months. We find that of all features, the temporal features provided the best prediction results. These features also outperform a simple regression models as well. The features we generate are interesting in and of themselves and may still be useful for answering other inference questions. In future work, we will try and predict other statistics with similar features, such as predicting the rating a user gives to a business based on the review text. We also aim to generate more features to improve accuracy.

REFERENCES

M Hu and B Liu. 2004. Mining opinion features in customer reviews. In *Proceedings of the National Conference on Artificial . . . .*

G A Miller, R Beckwith, and C Fellbaum. 1990. Introduction to wordnet: An on-line lexical database*. *International journal . . .* (1990).

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

A J Smola and B Schölkopf. 2004. A tutorial on support vector regression. *Statistics and computing* (2004).