Prediction of Useful Reviews on Yelp Dataset

Final Report

Yanrong Li, Yuhao Liu, Richard Chiou, Pradeep Kalipatnapu

Problem Statement and Background

Online reviews play a very important role in information dissemination and are influencing user decision. However, a user may only read a limited number of reviews before coming to a decision. An important aspect to the success of a rating and reviews site such as yelp, is to identify which reviews to promote as being useful.

To that extent, Yelp introduced voting on its reviews. Users vote “Useful”, “Funny” or “Cool” on yelp reviews, thus indicating which reviews should be promoted. However, for new reviews or businesses with low traffic this information does not exist. User votes are also not available on other consumer review sites. Thus, automatically predicting which reviews are useful and which are not is a problem of quite some interest.

Our data comes from the Yelp Dataset Challenge. As part of this challenge Yelp releases information about reviews, users and businesses from 4 US cities. The dataset (1.77 GB) is available for download on Yelp’s contest page and contains the following information:

- **1.6M** reviews and **500K** tips by **366K** users for **61K** businesses
- **481K** business attributes, e.g., hours, parking availability, ambience.
- Social network of **366K** users for a total of **2.9M** social edges.
- Aggregated check-ins over time for each of the **61K** businesses

As described in our preliminary reports the data is quite consistent, with very limited amounts of missing data. It does however, have other weaknesses. For example, since the “useful” voting feature on yelp was only introduced recently, many good reviews may not have been marked useful. Also, as a Web service, yelp’s data suffers from numerous grammatical errors.

Evaluation Techniques

In order to evaluate our methods, and models used, we need to agree on a set of success measures. For our project, we decided to classify a review as useful if it has at least one “useful” vote in the yelp dataset. The advantage of this metric is that, these are the reviews that Yelp is actually seeking to promote, so we’d like to identify similar reviews. The disadvantage however

is that, many good reviews may not have been read sufficient number of times to garner a “useful” vote. As such our data would have many false negatives to begin with.

With this usefulness metric, we evaluate our models on accuracy of the validation set. However, since the training data has far more not-useful samples than useful ones, we would also be interested in a breakdown of how our model is doing in each category.

There has been, unsurprisingly, quite some research in this area. There is one particular that we are interested in: Automatically Assessing Review Helpfulness by Soo-Min-Kim et al[1]. We will generate features similar to those mentioned in the paper, and attempt to create an SVM model with various kernels for satisfying this problem. [2] creates a text regression model, utilizing bag of words and reviewers’ RFM dimensions to predict usefulness of reviews on websites like Amazon, IMDB and TripAdvisor. [3] attempts LDA using features such as text length, funny votes, stars and dates on Yelp reviews.

Methods

Data Collection

As mentioned above, we obtained the dataset from Yelp. As part of the collection, we loaded the data into MongoDB. MongoDB provides an import tool that makes it easy to load json files. Using MongoDB made the rest of the data pipeline processes far faster.

Data Cleaning

There were two parts to our Data Cleaning approach. We first removed data we are not interested in to keep the dataset size manageable. Afterwards, we cleaned noisy data. As we were interested in user data but not their social information, we deleted information about check-ins, and social edges.

To remove noisy data, we did the following:

- We removed all non-letter symbols such as ‘&, /’ etc.
- We kept all the letter words and transform them to lower cases. We also kept all the numbers because we assume numbers such as prices of food would influence the usefulness of a review.
- We ignore all symbols that are not letters or numbers. Since we are using bag of words model, the sequence of words and sentence structures can be lost, we removed all the punctuations and split every review to a collection of words.
- Stopwords: we deleted all words that do not contain much meaning using stopwords provided by nltk package.

Data Transformation: Feature Extraction

We extracted numerous features relevant to our problem from the structured data, some of which were used in [1]. The features we extracted fall into the following broad categories:

Structural features

- Total number of tokens in a tokenized list of the review: A longer review is expected to be more useful and information to readers.
- Number of sentences per review: Similarly, a review with more sentences is expected to be more useful and information to readers.
- Average sentence length per review: Longer sentences yield more information in general, so a review with higher average sentence.
- Number of exclamation marks per review: A review with more exclamation marks suggests more enthusiasm from the reviewer. Of course, exclamation marks suggest a positive review as well.

Lexical features

Lexical features are traditionally the most relevant features in a text based model. As such we focused on extracting numerous lexical features. This extraction was memory intensive, and was performed on the EC2 instance. We stored these features in sparse matrix representation. Lexical features were extracted **after removing stop words.**

- TF-IDF: For tf-idf features, we picked the 1000 most frequent words gathered from reviews and calculate their tf-idf values.
- Unigrams of the 1000 most frequent words.
- 1000 most frequent bigrams in the data. After training SVMs on bigrams alone, we settled on using just the first 100 bigrams in our final model in the interest of time and performance. Examples: ((u'go', u'back'), 913), ((u'first', u'time'), 664), 751), ((u'really', u'good'), 636), ((u'great', u'place'), 600), ((u'ice', u'cream'), 491), etc.

Syntactic features

Syntactic features measure the part-of-speech distribution per review, i.e. the percentage of words that are verbs, nouns, adjectives, and adverbs.

Metadata features

- Rating (number of stars) associated with each review. We believe that rating is related with the usefulness, because a customer giving higher rating is more likely to be satisfied with the business, and may tend to write the review more carefully. A similar argument can be made for the other extreme.
- The absolute value of the difference between the rating of the review and the average rating of the business given by all reviewers. If a customer writes a review casually, it is very probable that he/she will give a rating near average rating. But if the reviewer is

subjective enough to overlook the average rating, then the review should include some extra information that most people don't give, and will likely be more helpful.

Semantic features

The original paper mentioned two semantic features: product-feature and general-inquirer. For product-feature, the author extracted the attribute keywords of a product from Epinions.com. However, the paper was modeling on reviews from Amazon.com where the products are concrete entities, and each kind of entities have corresponding attributes on Epinions.com. But we are investigating Yelp reviews on business and services, and these services doesn't have attribute set because they are not as specific as certain products. Therefore, we will not include product-feature.

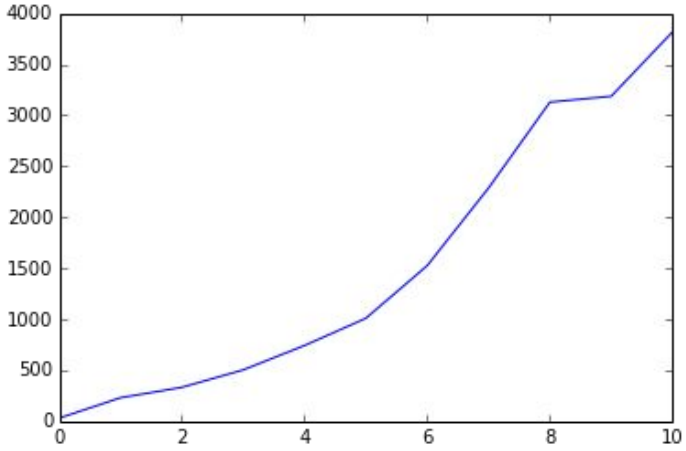
- For general-inquirer, the paper analyzed the appearance of sentiment words that describe the product features. Since we don't have product-feature, we simply analyzed the appearance of all the modifying words, because we believe each modifying word can convey some subjective emotion. The modifying word dictionary is adopted from General-Inquirer from Harvard University [4].
- We also think that people tend to vote for positive reviews more than negative reviews because they usually wish that the business has relatively high quality. So we want to make the positivity or negativity of a review as a feature. To quantify it, we simply counted the number of words that are strongly positive, moderately positive, weakly positive, strongly negative, moderately negative and weakly negative. Here we also used [4] as the sentimental words dictionary.

Foreign Key Features

The yelp dataset is not completely anonymized. While we do not have usernames, we still have access to user history. We also have access to business information, and its popularity. Unlike other papers in our relevant reading, we were able to mine this information.

- For each review, we extracted the history of the total number of votes the author received for past reviews.
- We also analyzed whether the author is an Elite user, and if so, for how many years. At Yelp, users who have a history of high quality reviews, are given Elite status. The following figure demonstrates the relationship between how long a user has maintained Elite status, and how many total votes their reviews have received. Considering that

most reviews do not get more than 5 votes, Elite users definitely pull their weight!



- We also extracted information about the popularity of businesses. This was determined by the total number of comments it received. We expect that more users read the useful reviews here, and that the quality of reviews would be affected as a result.

Data Analysis: Modelling

We used two primary models for our analysis: SVMs and Random Forests. SVMs were proposed by our reference paper [1], while ensemble learning methods have been documented to yield very high accuracies. So we implemented and compared the performance of both SVMs and Random Forests.

SVM

We used scikit learn's SVM implementation with Linear, Polynomial and RBF Kernels. We assessed performance (using default values of hyperparameters, k=1) and tuned hyperparameters on the best model.

Random Forest

We also used scikit learn's random forest implementation. We tuned the number of trees, and their depth using cross validation. We also experimented with which subset of features to include in our .

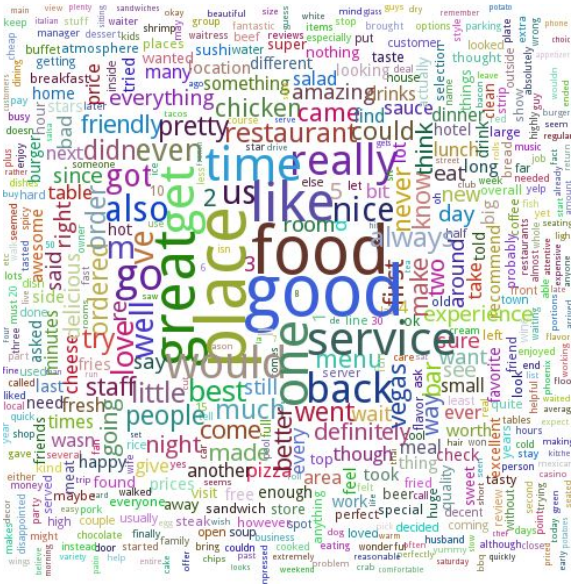
Data Visualization

Tag cloud of the most common words

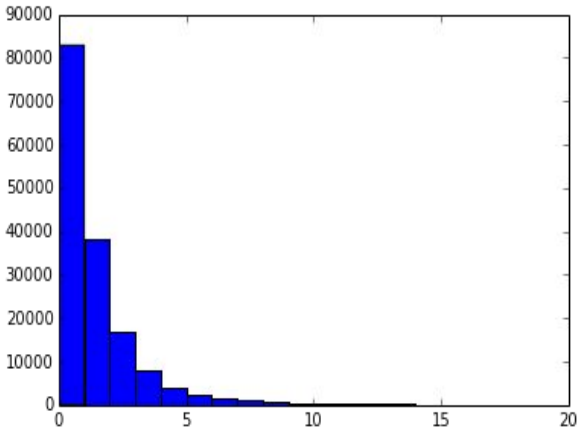
We created a tag cloud to visualize the words, and to confirm if our stopwords data cleaning was sufficient.

Reviews and Votes

The graph below is a histogram that shows the distribution of votes per review. The y-axis represents the number of reviews,



while the x-axis represents the number of votes the review had. **About 50% of reviews have 0 votes**. However, there are a significant number of reviews with 1 vote.



Failures

Even though we extracted numerous features, not all of them proved beneficial to our models. SVMs in particular are sensitive to noisy data. We used ablation to determine which sets of features led to the best results.

In particular, Metadata features, and Syntactic features did not improve the SVM model, but worked well with the Random Forest. We found that Lexical features that we extracted were not beneficial to either model.

Results

Results on SVMs

Feature Combinations	SVM - Linear Kernel (Accuracy)	SVM - Polynomial Kernel (Accuracy)	SVM - Radial Kernel (Accuracy)
All	0.5984	0.5112	0.6427
All - {Structural}	0.6031	0.522	0.6709
All - {Structural, MetaData}	0.6275	0.5431	0.6712
All - {Structural, MetaData, Syntactic}	0.61	0.47	0.6598

Breakdown of the Best SVM Model

Class	Precision	Recall	F1-Score
Not Useful	0.69	0.73	0.71
Useful	0.64	0.59	0.62

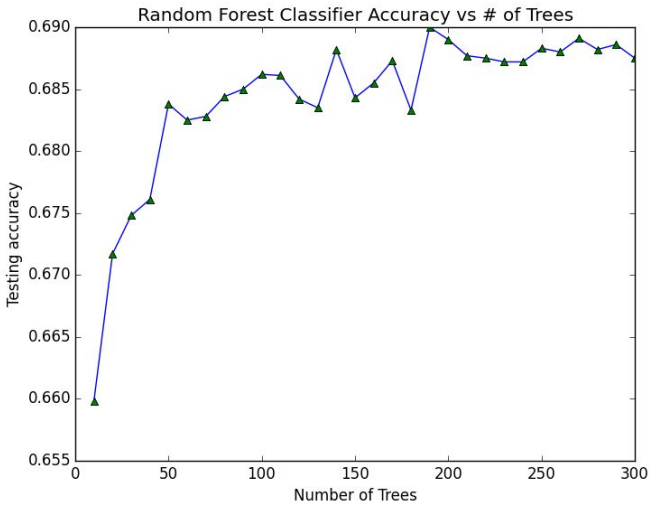
Result on Lexical Features

We extracted numerous lexical features, but did not find results using them to be promising. Here we mention accuracy scores using just Lexical features to underscore the result of the rest of our work.

Lexical Feature	Linear SVM	Radial SVM	Logistic Regression
Top 1000 frequent unigrams	0.5213	0.5558	0.5221
Top 100 frequent bigrams	0.548	0.5558	0.5416
Top 1000 frequent words + 100 frequent bigrams	0.5214	0.5558	0.5224

Results on Random Forests

The best random forest model incorporated **190** trees on six categories of features: the number of stars given by the reviewer, syntactic features, user history, metadata, structural features, and business popularity statistics. The random forest classifier returned **0.689** accuracy, a 0.02 increase over the best SVM model.



Class	Precision	Recall	F1-Score
Not Useful	0.69	0.78	0.71
Useful	0.68	0.57	0.62

Tools

NLTK: We used NLTK package for data cleaning and lexical feature extraction. The built in functions for removing stopwords and retrieving unigrams, bigrams were helpful.

The NLTK package worked well out of the box, but it was quite slow for POS tagging. We researched this topic for a fair amount of time, and came across the hunpos tagger. This combined with a model specifically meant for web data sped up our tagging process.

MongoDB: Fast joins between tables, helped with metadata and user history features.

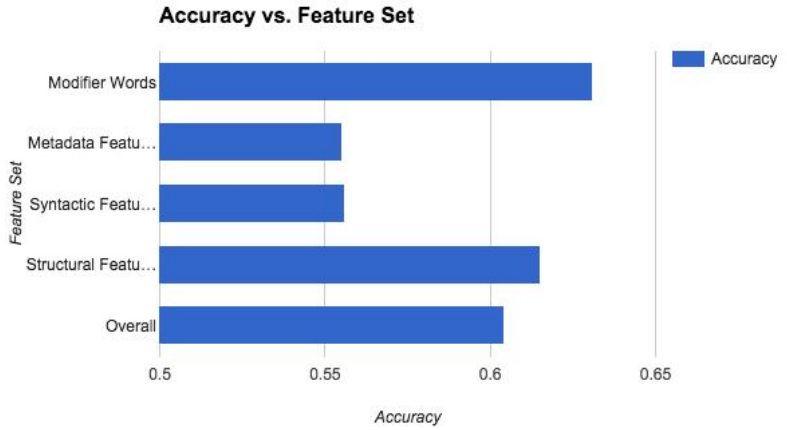
We go into further detail under the Lessons learnt section how MongoDB was very useful. The highlight was how simple it was to use, and how it worked glitch free.

Scikit-Learn: Standard implementations of ML models SVM, Logistic Regression and Random Forest.

Scikit learn performed reasonably well. Our SVM model on linear kernels took about an hour to converge on the complete dataset. But all other implementations were reasonably quick.

Lessons Learned

- Through testing with a fair amount of feature sets we realized the score of accuracy does not necessarily increase with the number of features. For example, while training SVMs, we originally expected lexical features of review text will have a great influence on the usefulness, but the end result shows that on the contrary they drag the accuracy down. As part of our initial analysis we came across an interesting accuracy graph. A few features were doing all the heavy lifting



- While extracting features, we quickly realized how long it takes to read all the reviews. Some features, especially ones that involved lookup two json files, like user history, were taking very long (18 mins). To solve this problem, we used MongoDb. We loaded all of our data on to mongodb and indexed on the review_id, user_id and business_id. After indexing, looking user history for each review took just 116 seconds.

CS294-16 Students: Baseline Model

We drew heavily from a paper on “Automatically Assessing Review Helpfulness”[1], so we decided to choose this paper as our baseline model. Although these models are not directly comparable, due to differences in what they assume to be ground truth and a different dataset they tested on, we feel that this baseline is valuable to judge the success of our project.

In [1], the authors have the benefit of training on two categories of reviews, MP3 players and Digital cameras. Their highest accuracy figure is achieved using RBF Kernel SVMs on Length (syntactic), unigrams and stars (metadata) features. The accuracy **0.656**$\pm$**0.33**. In the interest of a fair comparison, we replicated the work of the paper using the Yelp Dataset. The highest accuracy was once again using RBF Kernels. These same features scored an accuracy of **0.63**.

With this as baseline, we embarked to improve on it using Random Forests. We achieved an accuracy score of **0.689** as discussed in the results section.

We attribute this gain to being able to mine data that was unavailable to the authors of [1]. Specifically, we had access to User History information and Business History. Random forest was able to integrate these features into its decision making extremely well.
Also, as noticed in the baseline paper, SVM performance tails off with a large number of features, this creates need for more complex kernels. Since we were using many many features, we feel random forest was able to more robustly integrate the higher dimensional data.

Team Contributions

All team members contributed equally to the project (25% each). Key accomplishments are listed here:

Yanrong Li

- Examined using various models, and taggers for POS tagging reviews. With the large amount of review text, efficient POS tagging saved us time.
- Extracted Semantic Features and MetaData features.

Yuhao Liu

- Data Cleaning to manage dataset size, Removed general stop words. Identified yelp specific stop words from tf-idf analysis for removal as well.
- Extracted tf-idf, unigram and bigram features

- Setup General-inquirer to identify modifier and sentiment specific words. These were our best performing features.
- Analyzed usefulness of the traditional lexical feature by training models on these features alone

Richard Chiou

- Extracted Structural features.
- Suggested using Random Forests, and modelled them on extracted features.
- Tuned hyperparameters for our final model using cross validation after determining best feature set for Random Forests.

Pradeep Kalipatnapu

- Suggested and setup MongoDB, this made extracting features manifolds faster.
- Extracted foreign key features, such as user history and business popularity.
- Modelled SVM on the extracted data. Suggested ablation.

Bibliography

1. Kim, S.M., Pantel, P., and Chklovski, T., Pennacchiotti, M. 2006. Automatically Assessing Review Helpfulness. In Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing. Sydney, July, 423-430.
2. Thomas L. Ngo-Ye and Atish P. Sinha. 2013. The influence of reviewer engagement characteristics on online review helpfulness: A text regression model. In Decision Support Systems. Volume 61, 47-58
3. Shuyan Wang. 2015. Predicting Yelp Review Upvotes by Mining Underlying Topics.
4. Harvard University. 2002. General Inquirer. Retrieved Dec 10, 2015, from William James Hall: http://www.wjh.harvard.edu/~inquirer/