# Classifying Yelp Reviews

Colin Mullaney

-

DS 4400: Machine Learning and Data Mining

Fall 2017

# Classifying Yelp Reviews:

A dive into multi-class classification and ordinal regression for textual documents

Colin Mullaney
College of Computer and Information Science, Northeastern University
mullaney.c@husky.neu.edu

***Abstract* — this paper introduces the problem of ordinal sentiment classification in a common textual domain. We document the entire project lifecycle of building and evaluating multiple supervised machine learning models to address this problem. Documents from three separate subdomains of the Yelp review corpus are used for the training of these models. Evaluation is performed via cross-validation, as well as out-of-sample testing between these three subdomains. Metrics are compared to a naïve baseline to ensure significance.**

***Index Terms*—multi-class classification; ordinal regression; supervised machine learning; Yelp reviews.**

## I. Motivation

The motivation of this project is to examine the possibilities for predicting an ordinal value, representing sentiment, from nothing but the content of a document. Deriving insights from text is of great concern in recent years as the amount of textual data available continues to rise (e.g. social media posts, emails, etc.). The highly unstructured form of textual data allows for a large amount of freedom and creativity when it comes to interpretation, so it is up to the researcher to derive adequate features to suit the needs of a predictive model.

In this project, the idea is to use a common source of textual data to learn relationships between the text and an ordinal value representing sentiment. If a successful model were to be created, it could be applied to many other textual domains to predict relative sentiment.

## II. Data Domain

The textual data being used for this project comes from Yelp.com. Yelp is a highly popular website that publishes crowd-sourced reviews about local businesses. As of late September, 2017, Yelp.com contained over 142 million reviews, each one rating a business on a scale from 1 to 5. The average rating distribution of these reviews can be seen in Figure 1. As can be seen, there is a very large class imbalance among ratings. It seems that most users choose to write reviews for businesses when they are happy and have had a positive experience. This distribution lines up with the distribution of data we present in section III.

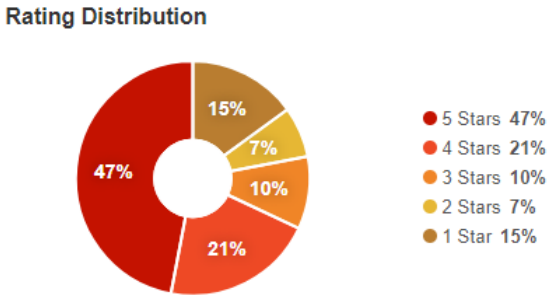


Figure 1. Distribution of ratings from Yelp.com

Yelp was an obvious choice when it came to collecting training data for this project, because each textual review was also given a numeric value from 1 to 5, signifying the reviewer's happiness with the reviewed business, with 1 being the worst and 5 being the best.

The businesses documented on Yelp range from restaurants, to contractors, to various service workers. In order to make a truly specialized model, we have decided to focus our study on three specific categories with the domain of 'restaurants'. These three categories include: 'breweries', 'burgers', and 'breakfast/brunch'.

While each of these categories is a subdomain of restaurants, they are all relatively unique. It can be implied that each set of reviews could have a unique vocabulary associated them, when compared to the other various categories available on Yelp. For this reason, we will be not only be performing cross-validation on each subdomain, but we will also perform out-of-sample testing by training on one domain and testing on another. Through this we hope to determine the importance of having a specialized vocabulary for text classification.

## III. Data Collection

As mentioned, the data for this project was gathered from Yelp.com. The process consisted of two steps:

First, the publically accessible Yelp Fusion API was queried to access lists of businesses that fell into the three aforementioned categories: breweries, burgers, and breakfast/brunch. From this data, we were only able to gather the business identification tags of the various businesses. We attempted to gather reviews from each of these businesses, however Yelp's API limits the user to a maximum of three written reviews per business. In addition,

each of these reviews only contained the first 160 characters of the review's text: clearly unusable for this project.

Instead, we used the business ID tags that we gathered from the API to scrape Yelp's website. We used the unique ID tags to scrape reviews for each business. For a given business, we were able to scrape up to twenty of the most recent reviews available. We chose to gather the most recent reviews to ensure the most current vocabulary was being used for our training data, as well as to prevent any biasing that may occur from using Yelp's built-in sorting algorithms. Altogether, we scraped 16,344 reviews.

After obtaining all of this information, we could do some initial data exploration. The distribution of ratings, shown in Figure 2, was extremely consistent with the overall distribution taken from Yelp's yearly factsheet. This trend was consistent across each of the three subdomains that we sampled. The existing class imbalance of all Yelp reviews stopped us from performing any filtering or removal of data that we had collected. Usually it is important to have equal class sizes when performing supervised learning. However, since we have knowledge of the prior probabilities of each rating, we chose to use this to our advantage and not equalize the class sizes of the data that we collected. Doing so could have introduced a new bias to our model. Instead, we attempt to combat these class imbalances by adjusting our model training and evaluation methods.

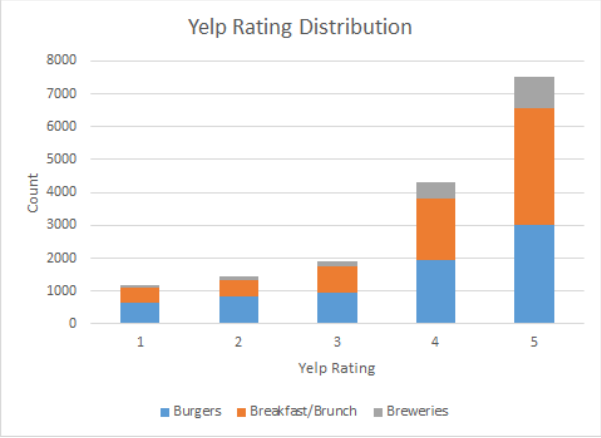


Figure 2: Distribution of ratings from 3 Yelp subdomains

Next, we briefly observe the lengths of the reviews that we have gathered by counting the number of space-delimited terms that are present. The general distribution of these term lengths, as well as the distribution of ratings, are detailed in Table 1. The average review contained about 90 terms, however there was a large variation in these term counts, due to some very long reviews (max 964 terms). Because of this, we ensured that any generated features would be normalized by review length, so that our model could be generalized to documents of varying lengths.

| Ratings | 1 | 2 | 3 | 4 | 5 | Mean | Std. |
|---|---|---|---|---|---|---|---|
| | **7%** | **9%** | **12%** | **26%** | **46%** | **3.93** | **1.29** |
| Terms* | 20% | 40% | 60% | 80% | 100% | Mean | Std. |
| | **31** | **52** | **80** | **133** | **964** | **89.6** | **82.3** |
| *Note: | Percentile values of term counts for collected reviews. | | | | | | |

Table 1: rating and term length distributions

## IV. FEATURE GENERATION

As previously mentioned, the goal of this study was to be able to predict relative sentiment using only the content of a given document. Therefore, the features for our model must be completely derived from the text of the reviews. By not relying on any additional metadata from Yelp (such as time of year posted, time of day posted, reliability of user, etc.), we ensure that our model can be tested on data gathered from other sources, besides Yelp.

The first derived feature takes into account the structure of the review, pre-cleaning. For this, we calculate the percentage of the characters in the document that are capitalized. This can often be used to catch extreme sentiment, when a user could be emphasizing words with anger or with excitement.

The next set of derived features took into account various elements of the text that would be removed post-cleaning. For this, we calculated normalized frequencies of the following punctuation marks: '.', ',', '!', '?', '#', '@'. These terms are common in user-generated documents, however are often omitted from classic text-based models that rather focus on word frequencies.

After generating those features, we systematically cleaned each review by removing all non-alphanumeric symbols and converting every character to lowercase. Next, we applied the Snowball stemmer from the Natural Language Toolkit (nltk) to each document, reducing each term to its "root" term (e.g. 'eating' -> 'eat'). Stemming each of our terms helps reduce the overall vocabulary of our documents by combining different forms of the same root term. In addition, this helps reduce the level of sparsity in our data, a common problem among text-based machine learning models.

After cleaning and stemming each document, we calculate term frequency-inverse document frequency (tfidf) values for every term. Tfidf is used to determine the relative importance of a term within a document, given the context of the entire corpus that you have available. To calculate tfidf for a single term, t, in a document, d, given a set of documents, D, you simply do the following: tfidf(t, d, D) = TF(t, d) x IDF(t, D), where TF(t, d) is the observed frequency of t in d (# occurrences of t in d/ # terms in d) and where IDF is the log of the inverse frequency of t with respect to D (log[# documents in D / # documents in D that contain t]). These values are much more robust than

simple word counts because they are normalized by the length of each document, an issue we addressed earlier. In addition, tfidf puts far less weight on common "stop words" that may appear in a very large proportion of documents in D (leading to a low IDF value, despite a potentially higher TF value), such as "a", "and", or "the". Tfidf is helpful for identifying keywords in documents, which should be weighted more when determining their impact on the model's prediction.

One issue that arose after calculating the tfidf values was the sheer dimensionality of our training data. When trained on all of our reviews, we had over 900 terms in our tfidf matrix, even after cleaning and stemming each document. To address this, we reduced the dimensionality of our tfidf features using a truncated singular value decomposition (svd) function. This type of svd is specialized for sparse data, which is often the case when calculating tfidf values for a large corpus.

## V. MODEL TRAINING

After generating features for our data, it was time to move to model training. For this project, we focused on four different machine learning algorithms. Three of these algorithms were multi-class classification algorithms and the fourth performed ordinal regression. The algorithms, and a brief description of their parameters, are as follows:

**Support Vector Classifier (SVC):** The two main parameters that we tuned for our SVC were C and γ. C is the regularization parameter that balances prediction accuracy with a simpler decision surface and simpler overall model. The γ parameter controls the 'radius' of influence for each of the support vectors in the model.

**Random Forest Classifier (RF):** The main parameters for a random forest model are the number of decision trees to grow, the maximum descriptive features to include for any split in the decision trees, and the minimum data points that are required per split before creating a leaf node.

**K Nearest Neighbors Classifier (KNN):** The two main parameters for a K nearest neighbors classifier are the number of neighbors to find when making a prediction, as well as the method for weighting each of these neighbors. Two weighting options are using uniform weights across all neighbors, or weighting 'closer' neighbors more heavily.

**Ordinal Ridge Regression (ORDR):** The only tunable parameter for the ordinal ridge regression was alpha, which controlled the level of l2 regularization.

For each of these models, we used a 75/25 train-test split of the various subdomains of reviews that we had collected. For each separate model, we performed a grid search over all combinations of parameters, and calculated metrics using 5-fold cross validation on the training subset. The best set of parameters was chosen from the cross validation, and the model was refit on the entire training set.

## VI. EVALUATION

For evaluation of our models during cross validation, we created two custom error functions. The first error function, which we called 'weighted entropy', made use of a model's ability to output probability predictions for each target class. These probabilities were multiplied by the absolute difference between the particular predicted class and the true rating for that data point. This allowed us to put larger weights on predictions that were further from the true value. This method is more flexible and granular than simply using predicted labels. We used this error function for training the random forest classifier, as well as the K nearest neighbors classifier.

The next error function, which we called 'class-averaged mean absolute deviation', only utilized the predicted label from a given model. We calculated the absolute difference between the predicted rating and true rating for each data point. These values were averaged across each class, giving each class the same weight, in the hopes of reducing the impact of the class imbalance in our training data. This value was used for training the support vector classifier, because it received much more promising training scores than when using the previous error function, and for training the ordinal ridge regression model, because the model could not output the class probabilities that were needed by the previous error function.

We used the 'class-averaged mean absolute deviation' metric to evaluate each model's performance on the left-out testing set, to allow for consistent comparison across all four models.

## VII. RESULTS

Before evaluating any of our models, it was first important to set a baseline performance for this task. We simulated a simplistic 'naïve' model that randomly guessed each target class in accordance with its prior probability, shown in Figure 1. The confusion matrix for this model's predictions is shown in Figure 3. The model had an overall error score of 1.751. This is the value that we will use to measure the relative performance of our models.

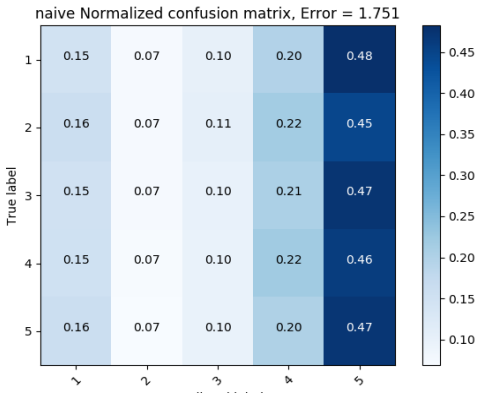


Figure 3: predictions for our naive model

For each of the three subdomains of Yelp reviews, as well as an aggregation of all three subdomains, we trained the four different models, and evaluated them on left-out test sets from the same subdomain. The error scores are shown in Table 2.

| | SVC | RF | KNN | ORDR |
|---|---|---|---|---|
| breweries | **0.699** | 1.439 | 1.128 | 0.895 |
| burgers | **0.696** | 1.229 | 1.089 | 0.764 |
| brunch | **0.763** | 1.586 | 1.094 | 0.854 |
| all | **0.563** | 1.006 | 0.869 | 0.779 |

Table 2: error metrics for each Yelp subdomain and model

The metrics from every model and subdomain combination outperform, though sometimes only slightly, the naïve baseline metric calculated at the beginning of this section. This shows that there is something promising to learn by analyzing the content of each review, rather than randomly guessing a rating.

Interestingly enough, the relative performance of each model was consistent across all subdomains used for testing. SVC consistently performed the best, followed by Ordinal Ridge Regression. This would imply that, in our case, model choice was not dependent on the specific subdomain of training data. One might have hypothesized that an ordinal regression model, specifically suited for this type of prediction task, would outperform a more traditional SVC. However, it is likely that the relatively high dimensionality of our input data, as is common among text classification problems, was not well suited for the Ordinal Ridge model. SVC models, especially those with a radial basis kernel, scale very well to larger dimensions of data. This, in concurrence with our customized error functions, which simulated the relative weights between the target classes, was likely enough to cause an SVC to outperform ORDR. Confusion matrix comparisons between SVC and ORDR are shown in Figures 4-7. The ORDR confusion matrices show stronger (darker), more confident predictions along the diagonal, however it has trouble classifying ratings of 1 and 2. Meanwhile, the SVC has much more consistent predictions along the diagonal, leading to a lower overall error.

It is also worth noting that the error metrics for SVC, RF, and KNN were all at a minimum when trained on the combination of all three data subdomains. This shows the importance of training data volume, especially in the context of deriving insights from textual data.

After comparing scores for each model and Yelp category based on its performance on the left-out test set from the same category, we performed out-of-sample testing using the SVC and ORDR models that we fit. For each model that was trained on a given subdomain of Yelp data, we tested it on the remaining subdomains, and evaluated its class-averaged mean absolute deviation scores. The values for the

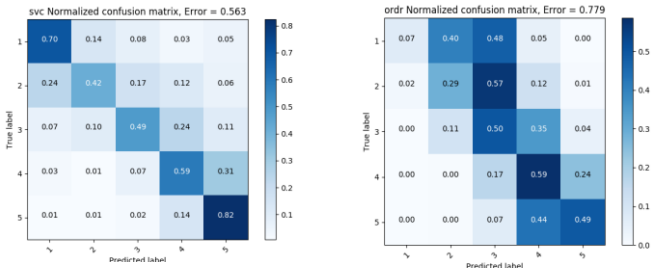


Figure 4: SVC vs ORDR: all categories

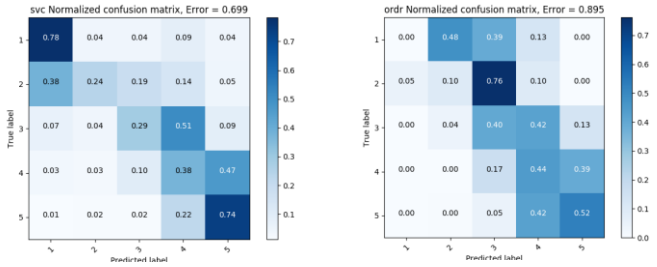


Figure 5: SVC vs ORDR: breweries


Figure 6: SVC vs ORDR: burgers

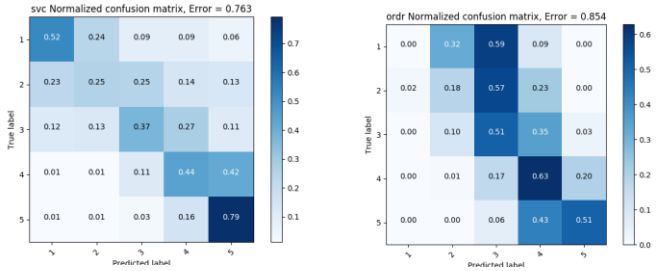


Figure 7: SVC vs ORDR: brunch

SVC models can be seen in Table 3, and the values for ORDR can be seen in Table 4.

The results from out-of-sample testing do not show large differences between the performances of models when evaluated on an 'unseen' domain. In some cases, these metrics even surpassed the metrics from the original evaluation on a left-out test set. Two possible conclusions are that: a) these three domains, all being subsets of restaurants within Yelp, are similar enough in vocabulary that each model can perform well on a category that it was not trained on, or b) the training domain and vocabulary is not the most important factor when it comes to the performance of a text classification model.

| SVC: | | Test | | |
|---|---|---|---|---|
| | | breweries | burgers | brunch |
| Train | breweries | **0.699*** | 0.849 | 0.871 |
| | burgers | 0.682 | 0.696* | **0.644** |
| | brunch | 0.778 | **0.698** | 0.763* |

Table 3: out of sample metrics, * represents left-out set metric from Table 2.

| ORDR: | | Test | | |
|---|---|---|---|---|
| | | breweries | burgers | brunch |
| Train | breweries | 0.895* | 0.896 | **0.893** |
| | burgers | 0.821 | **0.764*** | 0.893 |
| | brunch | 0.872 | 0.892 | **0.854*** |

Table 4: out of sample metrics, * represents left-out set metric from Table 2.

## VIII. FUTURE STEPS

Future steps for this project include greater development of derived features. Since textual data is incredibly unstructured, the level of insights that you can obtain directly correlates to the quality of your features. One possibility is taking into account ordering of terms in each review by creating n-gram terms for tfidf. In addition, incorporating an outside vocabulary of terms with distinct sentimental values, such as those from the Linguistic Inquiry and Word Count (LIWC). Finally, directly accounting for negating words (not, never, won't), and their impact on the interpretation of the other terms in the document.

Other than feature generation, further testing on other subdomains of Yelp reviews, even those outside the sphere of restaurants, could be of interest. It would be beneficial to reevaluate the out-of-sample results found in section VII to determine the cause for their similarities.

Finally, the problem of relative sentiment classification through an ordinal target variable could easily be simplified to a ternary sentiment classification (positive, negative, neutral). This could account for the variability in ratings given by users who are expressing the same general sentiment.

## REFERENCES

[1] "Evaluation measures for ordinal regression", Baccianella, Esuli and Sebastiani, 2009 Ninth International Conference on Intelligent Systems Design and Applications

[2] "Feature extraction and supervised learning on fMRI: from practice to theory", Pedregosa-Izquierdo, PhD. Thesis

[3] Yelp Factsheet, https://www.yelp.com/factsheet