Yelp Restaurants' Open Hours

Samuel Bakouch[1], Adrien Boch[2], Benjamin Favreau[3]

Abstract
This paper aims at identifying when restaurants registered on Yelp are open or closed based on the reviews written by Yelp's users. Our initial approach was to use a binary classification model with a bag-of-words representation of the reviews. We then conducted multiclass classification where logistic regression reaches 83.22% accuracy but with a 11.22% false-positive error rate which is critical for Yelp's user experience. We then pointed out that there is a tradeoff between model accuracy and false-positive error rate.

[1] *sbakouch@stanford.edu, Department of Management Science and Engineering, Stanford University*
[2] *aboch@stanford.edu, Department of Earth Sciences, Stanford University*
[3] *bfavreau@stanford.edu, Department of Management Science and Engineering, Stanford University*

Introduction

Yelp is an online platform publishing crowd-sourced reviews about local businesses. It is now a platform broadly used and it attracts numerous types of businesses. It has even become a key success factor in several industries such as restaurants and bars. It is now essential for these types of businesses to be referenced on Yelp with positive reviews, high rating and meaningful information.

Objectives and Motivation To keep on growing and taking an increasing importance in our everyday life, Yelp aims at providing valuable and complete information to its users. Unfortunately, when entering information into the platform, restaurants often forget to include and update elements that are particularly relevant for Yelp's users - e.g. the open hours of a business. Yelp must provide its users a consistent experience across the platform, and a restaurant without registered or accurate open hours should not jeopardize the overall user experience. Indeed, what is worse than choosing a restaurant on Yelp, driving there thinking about the menu to eventually find the door closed? This need for consistency and accuracy led us to apply machine learning techniques to determine the open hours of a restaurant from Yelp users' reviews.

Dataset We used the dataset publicly available, from the Yelp Dataset Challenge website. The dataset provides two .json files that we used in our study:

- A file containing information and characteristics about more than 42,000 businesses from the city of Phoenix
- A second file consisting of more than 1,125,000 reviews over these businesses

Among the characteristics available for a business are some labeling/category settings - e.g. **Restaurants**, Bars, Health& Beauty, ... and **open hours**.

1. Data Transformation

In order to be able to apply learning algorithms on reviews to predict the hours of operation of a restaurant, we first needed to transform the .json files we received. In the review file, every line contained one review for a given business. In the business file, one line contained the hours of operation of a given business. In order to facilitate processing, we not only needed to merge both files, but also to aggregate all the reviews of a business and to filter businesses that are restaurants.

1.1 Map-Reduce on Stanford Corn cluster

The business and review files contained about 42,000 lines and 1,200,000 lines respectively. Looping over these two files would potentially entail over $42,000 \times 1,200,000 \approx 5 \times 10^{10}$ iterations. However, as we only consider restaurants, the number of iterations boils down to 9×10^9 iterations.

Nevertheless, it is still a massive number of iterations and as a result extremely time consuming. That is why we designed a Map-Reduce algorithm to do parallel computing on the Stanford Corn cluster. In the mapping part, we broke down the business file into 480 different new files or pieces. For every single one of these pieces, a script was submitted a job on the Corn cluster so that multiple tasks could be run in parallel, using up to 30 different nodes. 480 output files were therefore created in the process. A second script has reduced these 480 output files together in order to get the output into one single file. At this point, we built a .json file of 8,079 lines in which one line corresponds to one restaurant, containing its aggregated reviews and its open hours.

1.2 Data preprocessing

We performed stemming on the reviews in order to reduce the dimension of our feature vectors. Stemming is a common practice used in Natural Language Processing, which reduces words to their root - e.g. thinking is represented by think after stemming processing. We used the nltk python library, which performs Porter's stemming algorithm [1]. We also removed stop-words to eliminate tokens that do not convey any information. Applying stop-words algorithm enabled us to remove high-frequency words such as "the", "I", "and".

2. Binary Models

Our goal being to predict the hours of operation of a restaurant from users' reviews, we must predict for each day of the week, if a restaurant is open or closed for each one-hour block of the day. Due to the large computing requirements needed for such an operation, we chose to focus on Tuesday exclusively - one of the most representative day of the week for restaurants excluding the weekend. We ran all our models on python.

2.1 Model hypotheses

First, our feature selection will follow the bag-of-words model, i.e. the dictionary used by the learning algorithms is defined as the union of all the words that appear in the reviews included in our training set.

$$F = \overbrace{\begin{bmatrix} 1 & 1 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 0 & \cdots & 0 & 1 \end{bmatrix}}^{Size\ of\ dictionary} \Bigg\} \text{nb. rest.}$$

where $F_{i,j}$ indicates whether feature j appears in the reviews of restaurant i.

Our aim is to predict for each one-hour block of Tuesday if a restaurant is open or closed. There are several ways to represent this output and define the labels predicted by the learning algorithm. For instance, we could look at the 24-hour vectors, or we could view the output representation as 24 binary predictions where - for each one-hour block - the learning algorithm will predict 0 or 1. In this case, we apply a binary algorithm for every one-hour block, thus the number of labels is $|\{0,1\}| = 2$. In this section, we chose to take the latter approach considering the potential huge number of classes of the previous output representation - $2^{24} = 16,777,216$.

2.2 Methodology

2.2.1 Error metric

When choosing an error metric, a key characteristic should be the ability to benchmark our results against several types of algorithms. We chose the standard average test error looking at the accuracy of the prediction for each one-hour block

$$\varepsilon = 1 - \frac{\sum_{i=1}^{t}\sum_{ho=1}^{24} \mathbb{1}\left\{h(x^{(i)})^{ho} = y^{(ho)}\right\}}{24 \times t}$$

where t is the size of the test set, $y^{(ho)}$ is the true state of operation of the restaurant at hour ho, $h(x^{(i)})^{ho}$ being the predicted output for hour ho and training example i . We then used a 10-fold cross validation which is standard for text classification problems [2].

2.2.2 Training set size

In order to determine the training set size on which we will train our algorithms, we first decided to plot - cf. Figure 1, the training and test errors for different training set sizes ranging from 100 to the maximum 8,079. We used the Binomial Naive Bayes algorithm to perform this analysis. We observe two main behaviors: from 100 to 1,000, the algorithm overfits and from 1,000 to 8,079, the behavior is similar as the training and test errors slightly differ in functon of the training set size. Thus, as the complexity of the learning algorithms is at least $\mathcal{O}(t^2)$ where t is the size of the training set, we decided to choose 1,000 training examples for the rest of our study. Also, one can notice that 1,000 training examples correspond to the one standard error rule, which is a standard parameter selection criterion used in statistics [3].

Figure 1. Learing curve for Binomial Naive Bayes

2.2.3 Baseline algorithm

In order to accurately measure the performance of the learning algorithms we choose to apply, we need a baseline model for which the previously-defined error metric will be computed. We will consider two baseline algorithms:

- Predict close - i.e. 0 for every hour, error: $\overline{\varepsilon}^0$
- Predict open - i.e. 1 for every hour, error: $\overline{\varepsilon}^1$

To compute the error metric that should be the reference to benchmark against, we will compute the quantity

$$\overline{\varepsilon}^{ref} = \min(\overline{\varepsilon}^0, \overline{\varepsilon}^1)$$

Running these two naive algorithms on our dataset outputs, we obtained $\overline{\varepsilon}^{ref} = 42.08\%$. Thus, we will not consider learning algorithms with an error greater than 42.08% as the baseline algorithm performs better.

2.3 Learning algorithms

2.3.1 Naive Bayes

First, we decided to apply the binomial (bNB) and multinomial (mNB) Naive Bayes' algorithms in order to test our hypotheses. The bNB led to 35.31% of error and the mNB led to 37.96% of error. We note that these errors have the same order of magnitude and both are under the baseline error cap.

2.3.2 KNN, SVM, Logistic regression, Perceptron

Then, we decided to test additional algorithms and thus to refine our model. Indeed, applying Naive Bayes to our dataset,

we made the implicit Naive Bayes assumption that the features are conditionally independent given a label. As this is a strong assumption that is not verified in our problem, we applied SVM with a Gaussian kernel (SVM), K-nearest neighbors (KNN), logistic regression, and the perceptron algorithms and compared their respective performances.

Feature weighting Before applying learning algorithms, it is standard in text classification problems to apply feature weighting to our count matrix in order to reflect the relative importance of features[4]. To do so, we used the tf-idf measure, which increases proportionally to the number of times a word appears in the document, but is offset by the frequency of the word in the corpus, which helps to adjust for the fact that some words appear more frequently in general. The formula to compute tf-idf is the following:

$$w_{i,j} = tf_{i,j} \times \log \frac{N}{df_i}$$

where $tf_{i,j}$ is the number of occurrences of word i in training example j, df_i is the number of training examples containing the word i, and N is the total number of training examples. Applying tf-idf to weight our features and running the aformentionned learning algorithms, we note that KNN with $K = 240$ ranks first with 16.93% of error while logistic regression with regularization parameter $\lambda = 10^{-3}$ and SVM and a regularization parameter $C = 10^{-6}$ closely follow with respectively 16.97% and 17.39% of error - cf. Figure 2. Multiple parameters K, λ, C were tested and the chosen ones were determined as they minimized the cross validated error.

2.4 Feature selection

Another way to account for the relevancy of features to a given classification problem is to select a subset of features. Indeed, the dataset contains many redundant and irrelevant features. Redundant features are those which provide no more information than the currently selected features, and irrelevant features provide no useful information in any context. The main advantages of applying feature selection are to improve the model interpretability, to enhance generalization by reducing overfitting, and to shorten training times.

To perform feature selection, we applied the chi-square method that is standard in the text classification literature [5]. We use it to test whether the occurrence of a specific term and the occurrence of a specific class are independent. Thus we estimate the following quantity for each feature j and we rank them by their score:

$$\chi^2(j) = \frac{1}{2} \sum_{i \in \{0,1\}} \sum_{e_i \in \{0,1\}} \sum_{e_j \in \{0,1\}} \frac{(N_{e_i e_j} - E_{e_i e_j})^2}{E_{e_i e_j}}$$

where e_j denotes the occurrence of feature j, e_i denotes the occurrence of the class i, and $N_{e_i e_j}$ denotes the counts of training examples when either e_i or e_j is equal to 1. Last, $E_{e_i e_j}$ is the expected frequency asserting that the occurrence of feature j and the occurrence of class i are independent.

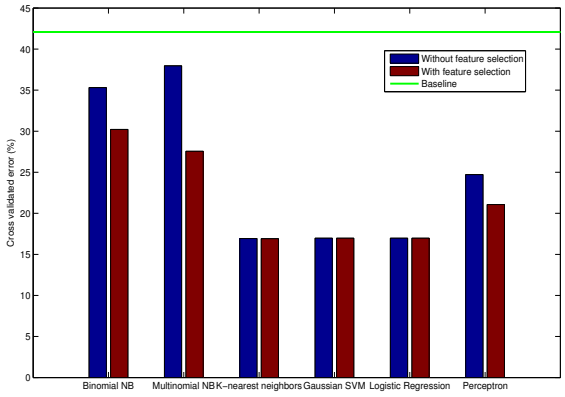

Figure 2. Binary Classification Results

We applied the bisection algorithm in order to find the optimal number of features to use for each learning algorithm, corresponding to the minimum of the cross validated error for the hour 11am - cf. Figure 3. We chose 11am as a proxy since it is the hour for which our baseline algorithm performs the worst: 11am represents 12% of the total error compared to 4.2% if the error was uniformly distributed.

2.5 Analysis

After applying feature selection, we conclude that KNN with $K = 240$, logistic regression with parameter $\lambda = 10^{-3}$ and SVM and parameter $C = 10^{-6}$ perform similarly as in the no feature selection case. Moreover, we note that the algorithms that were performing worse - i.e. bNB, mNB, and the perceptron algorithm, now perform significantly better with feature selection: 15% improvement for bNB, 28% improvement for mNB, and 15% improvement for perceptron. Though these algorithms improved with feature selection, they still do not reach the level of performance provided by KNN, SVM, and logistic regression.

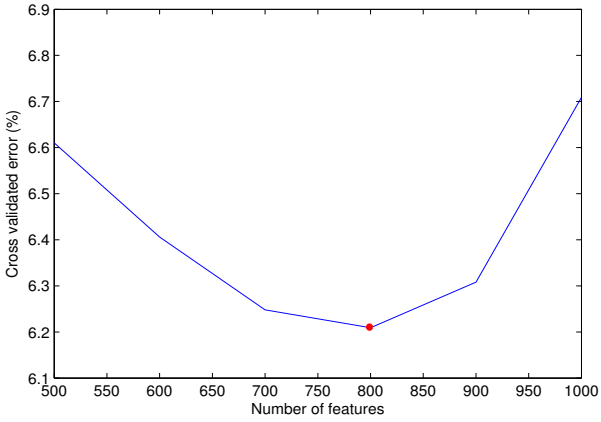

Figure 3. Feature Selection for Binomial Naive Bayes

3. Multi-class models

3.1 Motivation

In section 2, we applied several algorithms based on binary classification outputs. For a given business, the predicted operation hours are obtained by concatenating the prediction of one algorithm over 24 hours. Thus, algorithms are run independently every hour and the prediction from one hour is completely independent from the prediction of the adjacent hours. As mentioned earlier, in this setting, K-nearest neighbor, SVM and logistic regression have a similar prediction accuracy, but lead to unrealistic predicted answers as we can see in the example below. Such example would mean that the associated restaurant opens 5 times during Tuesday.

$$\hat{Y} = [00\underline{11}000\underline{111}00\underline{11}0\underline{1}0\underline{1111}000]^T$$

One solution to avoid this kind of patterns would be to group a given number of hours and make prediction over this group of hours. Such algorithms are called multi-class classification. However, how to make a prediction on a specific hour-slot *ho*? We would need either to make several predictions based on all the groups containing the slot *ho* or to arbitrarily choose one of these groups.

To withdraw the uncertainty on how to make our predictions on each time slot, we decided to use the entire 24-hour vector as mentioned in part 2.1. We noticed that our dataset contains only 229 different vectors. Thus, running multi-class classification algorithms on 229 classes over more than 8,000 businesses seems realistic. In order to be able to benchmark these new algorithms, we used the same measure of accuracy as in section 2 using the same notations, $h(x^{(i)})^{(ho)}$ being the ho^{th} element of the predicted class $h(x^{(i)})$.

3.2 Results

We extended the models tested in section 2 to fit the new type of answer - except the SVM classifier only accepting binary answer vectors. In short, we tested bNB, mNB, KNN, logistic regression and perceptron algorithms. We used the same methodology as in section 2 for every models:

1. We used a bag-of-words model to transform reviews into usable feature vectors
2. We weighted the features based on tf-idf - except in the Naive Bayes algorithms
3. We fitted algorithms using all features available
4. Using χ^2 test, we reduced the number of features
5. We fitted our algorithms with the optimal number of features obtained at step 4

Based on this methodology, we obtained the results presented in Figure 4. We not only did not lose any prediction power, but we even improved it! Multi-class logistic regression hits 16.81% prediction accuracy improving our former performance by 0.7%.

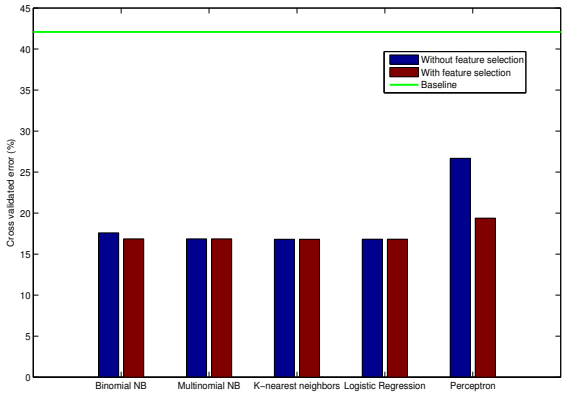

Figure 4. Multi-class Classification Results

3.3 Analysis

Even though our new methodology does not focus on maximizing the cross validated error and the $F1-score$ on an hourly basis, it classifies restaurants in observed classes. Binary classification reduces the *bias* on every hour-slot considered, but increases the overall *variance* of our prediction algorithm - as predictions are independent from one hour to another, variances are simply summed over the hours. Using multi-class classification, we have introduced an additional bias to our model, but we decreased the variance on a 24-hour metric - variance pooling, which explains why our new methodology has good performance.

We can also observe that our models are less sensitive to feature selection than earlier. This might be explained by the fact that the answer vectors contain more information than earlier. Moreover, in our subset of restaurants, we have 123 classes, i.e. on average one class has a size of less than 10 restaurants. While reducing the number of features, our algorithms tended to be less performant, and the performance was monotically increasing with the number of features included in the model, which tends to support our theory.

As we can see in Figure 5, the prediction error is not uniformly

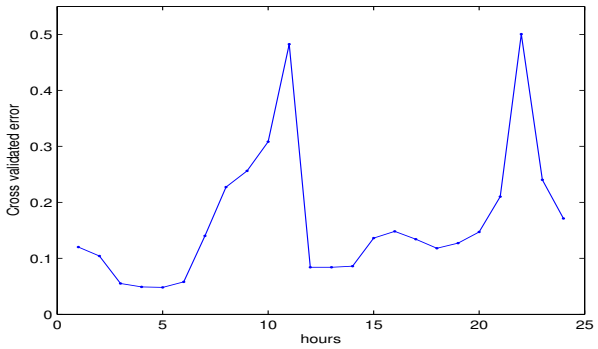

Figure 5. Multi-class logistic regression CV error per hour

distributed over the 24 hours of the day. In particular, we observe two error peaks centered at $hour = 11$ and 22. Even if the logistic regression model performs well in-between and outside of these peaks, there is still room for improvement.

4. Improvement strategies

At this point, we chose to focus on the logistic regression model based on the results observed in section 3. As mentioned earlier, multi-class logistic regression is not sensitive to feature selection, it performs even worse when we reduce the number of features. Given Yelp's business strategy, predicting that a restaurant is *open* while it is currently *closed* is particularly harmful for its brand image. Thus, we will try to implement new strategies not only to improve the overall prediction power of our model, but also to reduce the 11.2% false-positive error rate obtained so far.

$$\begin{array}{cc} & \overbrace{\textit{Predicted}}^{} \\ & \overbrace{\textit{Close} \quad \textit{Open}}^{} \\ \textit{Actual} \begin{cases} \textit{Close} \\ \textit{Open} \end{cases} & \left[\begin{array}{cc} 88.8\% & 11.2\% \\ 23.3\% & 76.7\% \end{array}\right] \end{array}$$

4.1 Bigram, trigram

Based on the observation that cross-validated error tends to its optimum while increasing the number of features, we decided to introduce new features in the model. Our model intrinsically depends on the order of words in a review. A simple observation supports this argument: "11am" is one of the top features in our current prediction algorithm. However, while "tuesday 11am" is relevant to our problem, "friday 11am" is particularly misleading! These observations motivated us to introduce a new type of feature: bigrams i.e. on top of the usual bag-of-word model, we added all the sequences of two consecutive words in the reviews. Therefore, from 60,000 unigram dictionary used so far, we introduced $> 1,300,000$ new bigram features. We then ran our multi-class algorithm on the new input matrix and obtained an error rate of 16.80%, which did not lead to a significant improvement in terms of prediction power. We then ran our algorithm including trigrams - consecutive sequence of three words, and obtained 16.82% error rate. Thus, we can conclude that neither bigram nor trigram enabled significant accuracy jump.

4.2 Classification error weighting

One strategy to align our algorithm with our objective to decrease the false-positive error rate is to weight the misclassification of the class "closed" more heavily than the misclassification of the class "open". In order to be able to implement this strategy, we came back to binary algorithms. We can notice in Figure 6 that our optimal cross-validated test error 16.78% is obtained when penalization false-positive misclassification three times more - false-positive error rate of 10.18% . We note that there is a tradeoff between decreasing the cross-validated error and reducing the false-positive error rate. To further reduce the false-positive error rate, we thus need to tolerate a lower prediction accuracy.

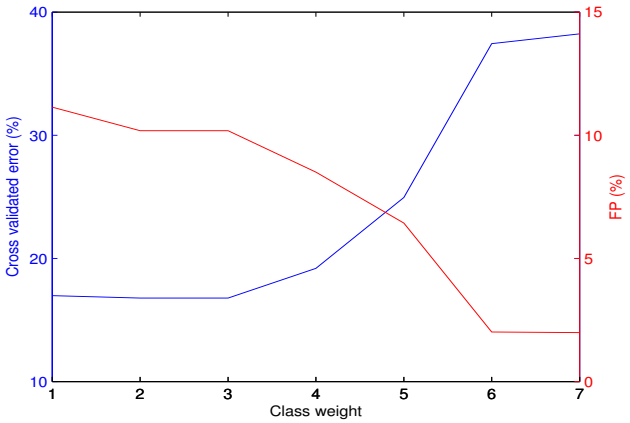

Figure 6. CV error vs. false-positive error rate

5. Discussion

During this project, we have implemented several supervised text-classifier algorithms with different output representations. While binary models gave good prediction power, they produced unrealistic predictions. We then transformed our outputs and performed multi-class classification. Using a very large dataset of Yelp reviews and a logistic classifier, we have been able to correctly predict hours of operation 83.22% of the time - error rate of 16.78%. Moreover, multi-class classification algorithms performed particularly well, since they significantly reduced the variance problem observed for binary models. We then particularly focused on refining our algorithm in order not only to increase its prediction power but also to fit to Yelp's overall strategy. By penalizing a certain type of misclassification, we have been able to significantly cut the false positive error rate, which is critical to Yelp.

In the future, it would be interesting to consider several new strategies to improve the performance of our algorithm:

- **Training set selection:** monitor more closely the businesses present in our dataset - restaurants with fewer reviews introduce bias due to the lack of information
- **Feature filtering:** use backward or forward selections to reduce the noise due to the high number of features
- **Feature representation:** test new feature representation instead of the usual count, which is slightly biased toward businesses that contain more reviews

References

[1] E. Loper. (2009), *Natural Language Toolkit*,
[2] M. Stone. (1977), *Asymptotics for and against cross-validation*
[3] R. Tibshirani. (2013), *Model assessment and cross-validation*
[4] P. Meesad. (2011), *A chi-square-test for word importance differentiation in text classification*
[5] S. M. Weiss & al. (2004), *Text mining: predictive methods for analyzing unstructured information*