Hidden Topics:
Predicting Business Categories with Review Text

UCSD CSE 190: Data Mining and Predictive Analytics
Joseph Anz A10640444, Luis Arriaga A12030985
December 1, 2015

Abstract:

Using the Yelp Dataset we explored this dataset comprehensively to figure out some predictive tasks that we can analyze and compute. The main predictive task we will elaborate on is how we figured out business category depending on the user's review. We will initially describe some trends and observations we discovered from messing with the dataset and then go on to explain the predictive task that actually worked. Furthermore, we will explain the steps we used to preprocess and condense the data, along with the procedure to obtain a baseline model for our predictive task using a Naive Bayes classifier. To beat the baseline we implemented a Support Vector Machine that performed better than the Naive Bayes solution. Our results section will confirm our findings and our literature section can be used for supplemental sources on predictive algorithms on the yelp dataset.

1. Introduction

The Dataset:

The Yelp Dataset Challenge consisted of 1.6 million reviews by 366,000 users for 61,000 businesses. Each business has up to 481,000 attributes such as hours, parking availability, ambience, location, etc. Luckily, Yelp split the data into logical json files. The business json file gives necessary information about the business such as address, category, number of reviews, business hours and etc. The next json file, the review json file was also essential to get a mapping between the review a user wrote and for what business. The other json files, corresponding to user, checkin, and tips, were used for initial testing to figure out what predictive task we wanted to implement, but not used for our finally predictive task that we implemented.

What stood out to us from the json files was that each business had categories assigned to it, 783 categories to be exact. Some examples of these categories are Restaurants, Bars, Health & Medical, Bakeries, Ice Cream & Frozen Yogurt, Pubs, Cafes, Dentists, etc. Each business is already classified into several of these 783 categories.

Since these businesses are already labeled it lead perfectly into a supervised learning approach - the basic foundation for a machine learning classification problem. This task interprets a function from labeled training data. When using supervised learning, each sample consists of two parts input values and the desired output which is already known, but in order to be able to predict this output we must design a supervised signal. The optimal goal for the algorithm was to accurately predict the classification of a business review to a type of business category.

Preprocessing:

After figuring out what predictive task we wanted to compute, we had to preprocess our data. Our first goal was to understand the category or categories each business is a part of. We created a set of directories corresponding to each of the 783 categories. For every user review that corresponded to a business, we would write the review text to a text file in the appropriate categories directory. Note, the process will write the same review to multiple categories if the business is tied to multiple categories such as Food, Pizza, and Restaurants.

We noticed there were categories with a very high number of occurrences and some with a very low number of occurrences, which could possibly alter our predictor for the worse. The objective is to determine an absolute training model excluding outliers. We came to a conclusion of any category appearing more than 2380 instances should not have it's own category file. The thought process and analytics behind this reasoning was a trend we observed between categories above that number and below. For example Restaurant and Food appeared well over 16,000 times but Pizza, Mexican, Burger, etc. were below this value, yet these categories are still part of and related to the more broad categories described, thus excluding outliers prevented overfitting. At the same time, any category occurring a minimum number of times would not have been helpful in developing our predictor as well, causing underfitting.

2. Predictive Tasks:

Obtaining Features from Data Processing:

What we are trying to predict is which category a review written by a user actually belongs to and if it accurately matches to a category which describes the business. Since the Yelp dataset is very dense, and we preprocessed 783 directories mapping to the 783 categories each with thousands of reviews, it was difficult to run our algorithm on the entire dataset. Instead we choose only 5 categories to see if we can read a review and figure out which category it belongs in. The 5 categories we chose were Auto Repair, Beer & Wine & Spirits, Buffets, Day Spas, and Hotels. The total amount of samples were 74,510, where each category had roughly 15k - allowing each to have an equal number of reviews, but not an equal review quality because we can't control what reviews a user writes.

Once we loaded the 75k reviews, we had to make sure that the actual text format did not mess with our vectors. We had issues where the user would include emojis and special characters into their review which would throw off our encoding. We standardized the encoding with utf-8 and made sure that if any errors were thrown, we would disregard that encoded symbol. We utilized the sklearn library to build our feature vector. We used a tfidfvectorizer to convert a collection of raw documents to a matrix of TF-IDF features. This is like a bag of words approach, but instead of word counts, it is a normalized way of getting text frequency inverse document frequency. Also, we removed all the english stop words from the text, but did not perform any stemming.

Extracting features from the training data using a sparse vectorizer we had n_samples as 20000 and n_features as 37360. Extracting features from the test data using the same vectorizer obtained n_samples of 54509 and n_features of 37360. So, we tested on more samples then we trained on. In the next section we describe the actual algorithms we used for training and testing.

3. Our Model:

We created our models using the Naive Bayes estimator and the Linear SVC estimator. The reason we used these two was because we fell into a classification problem with labeled data and that we have less than 100k samples.

Strengths and Weaknesses:

	Naive Bayes	**Linear SVMs**
Pros	1. Easiest to implement a. that's why it is our baseline 2. Most efficient to train a. Referring to the other table, it is evident that training time is extremely fast.	1. Non-probabilistic: optimizes the classification error rather than the likelihood 2. Effective in high dimensional spaces.
Cons	1. It suffers from double counting	1. More expensive to train. a. With the same number of samples for training, the SVM took 31x longer to train.

Figure 1: Table showing the advantages and disadvantages of the estimator models chosen

Naive Bayes:

Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of independence between every pair of features.Bayes' theorem states the following relationship, given a class variable y and a dependent feature vector x1 through xn:

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots x_n \mid y)}{P(x_1, \ldots, x_n)}$$

with a naive assumption:

$$P(x_i|y, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = P(x_i|y),$$

for all i and simplified:

$$\hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i \mid y),$$

Our baseline was using the Bernoulli Naive Bayes Rule. This was chosen as our baseline because research shows that this estimator might perform better on some datasets, especially those with shorter documents. Since we can't control how much a yelper will review a business, they might have really short review or might have a very descriptive review; therefore, this seemed as an appropriate baseline because of the two extremes.

The decision rule for Bernoulli naive Bayes is based on:

$$P(x_i \mid y) = P(i \mid y)x_i + (1 - P(i \mid y))(1 - x_i),$$

having numerous features this estimator assumes that each feature is binary-valued.

To improve from the baseline, we used a different Naive Bayes function known as the multinomial Naive Bayes classifier. The multinomialNB is suitable for classification with discrete features such as word counts for text classification. The distribution we used was with fractional counts which was obtained from text frequency-inverse document frequency (tf-idf), with all english stopwords removed, which is what is usually done in practice. Looking at the below figure we can see that we improved our accuracy by 4% from our baseline.

	Bernoulli Naive Bayes	**Multinomial Naive Bayes**	**LinearSVC**
training time	0.082s	0.166s	2.555s
testing time	0.147s	0.069s	0.434s
accuracy	0.839	0.874	0.891

Figure 2: A table showing the results of running our 3 models on the same training and test sets.

Support Vector Machines:

The intuition behind Support Vector Machines (SVMs) is to train a classifier that focuses on the "difficult" examples by minimizing the misclassification error.

$$\arg\min_\theta \sum_i \delta(y_i(X_i \cdot \theta - \alpha) \leq 0)$$

Here it is important to note, that we were using python version 2.6 and the sklearn library heavily to develop our predictive algorithms. For LinearSVC, we had different parameter configurations until we got results that were consistent and accurate. The following italicized lines are our printed sklearn LinearSVC pipeline that shows all our feature selection and classification parameter configurations (some values printed are default values):

> *Pipeline(steps=[('feature_selection', LinearSVC(C=1.0, class_weight=None, dual=False, fit_intercept=True,intercept_scaling=1, loss='squared_hinge', max_iter=1000, multi_class='ovr', penalty='l2', random_state=None, tol=0.001,verbose=0)), ('classification', LinearSVC(C=1.0, class_weight=None, dual=True, fit_intercept=True, intercept_scaling=1, loss='squared_hinge', max_iter=1000, multi_class='ovr', penalty='l2', random_state=None, tol=0.0001, verbose=0))])*

Unsuccessful Attempts:

The 5 categories we chose were Auto Repair, Beer & Wine & Spirits, Buffets, Day Spas, and Hotels. These categories are pretty distinct with minimal overlap - some words do overlap which can be seen from the top 10 words from Figure 4. Before using these categories, we tested on these following categories: Baby Gear & Furniture, Bagels, Bakeries, Banks & Credit Unions, Furniture Stores. We had 50k samples and trained on 30k of them. This lead to overfitting because Baby Gear and Furniture had overlap with the review along with Bagels and Bakeries. Some of the businesses belonged to multiple of these categories which meant we trained on them twice, and would double count them, which would yield

a higher chance of being correct. With the LinearSVC we achieved an accuracy of 93% with the model, but the data was overfitted.

4. Literature

Our dataset is a Yelp Dataset used for annual challenges intended for academic project as part of ongoing study for students. This dataset is obtained from reviews of multiple businesses throughout the entire world. Over the span of the last few years the dataset has been changed or concatenated with more data from Yelp reviews. For the past 5 years students in the academia have been attempting Yelp challenges with similar types of datasets and challenges. The first year the challenge was held, Professor Julian McAuley attempted and won the same challenge as we attempted this year. The dataset he used was much larger than ours consisting of 42 million reviews, written by 10 million users. He also used a different predictive task compared to ours, which was a recommender system using Latent-factor models.

Currently, recommender systems are the best methods to determine the category of the business a specific review was made for. Although, Professor McAuley went even more in depth designing a Hidden Factors as Topics Model, which discovers topics relevant with hidden factors of products and users, with only a few reviews, making it the best model of its type. The HFT model does not require output variables, it is an unsupervised model, thus making it distinct from our supervised model. The results from his experiment are similar to ours, in the sense that utilizing the review text you are able to predict a product or in our case the category of the business.

5. Results

From Figure 3, it is evident to see that the Support Vector Machine Pipeline is the most accurate of the 3 at the cost of more time to train and test (in seconds). With nearly 31 times longer to train, the algorithm is about 7% more accurate. With larger datasets this tradeoff might not be worth it because it is computationally expensive.

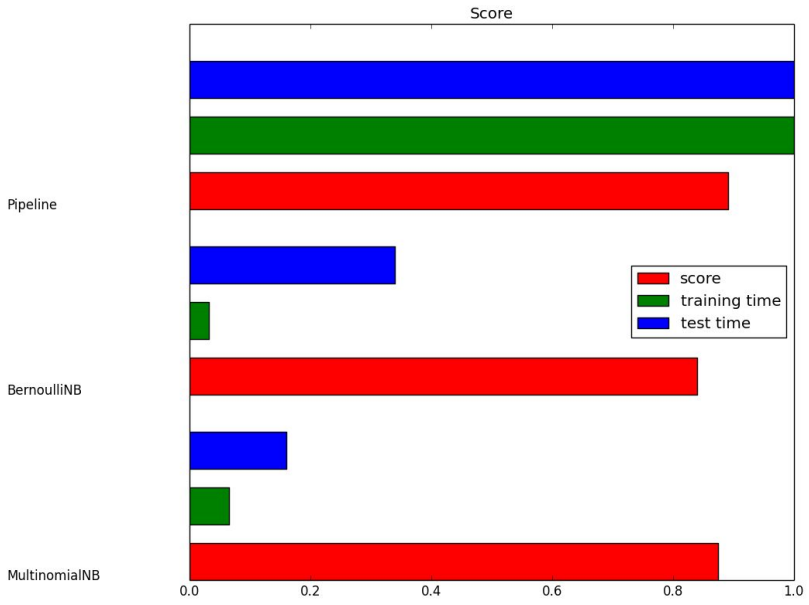

Figure 3: A visual representation of the table from figure 2.

At the same time, it takes longer to implement a support vector machine than a naive bayes estimator. The reason behind this is that SVM's have number parameter combinations and depending on what data is used, the format it is in, and what is trying to be predicted, these parameters need to be configured and tested extensively.

From Figure 4, it shows that the feature vector using a tfidvectorizer for words, without stopwords yields consistent results because the top words in each cluster agrees with our mental models for each category. Yes, there might be better words that more accurately describe each category, but if the average yelper doesn't use those words, then it won't show up in the list. The best example of this is that Auto Repairs yelpers use the word "guys" more frequently than "mechanic", which shows that some "slang" words appear more than the actual technical words in the yelpers vernacular.

	Auto Repair	**Beer, Wine & Spirits**	**Buffets**	**Day Spas**	**Hotels**
Top 10 words for each category	1. car 2. oil 3. service 4. work 5. shop 6. honest 7. change 8. auto 9. repair 10. guys	1. place 2. great 3. good 4. food 5. service 6. love 7. like 8. time 9. wine 10. just	1. buffet 2. food 3. good 4. buffets 5. crab 6. lunch 7. like 8. eat 9. price 10. sushi	1. massage 2. spa 3. facial 4. room 5. great 6. relaxing 7. therapist 8. massages 9. time 10. day	1. room 2. hotel 3. nice 4. stay 5. rooms 6. pool 7. strip 8. great 9. stayed 10. Vegas

Figure 4: Print ten most discriminative terms per class.

Overall, the support vector machine model is appropriate for this dataset. It has the ability to retrain the model as new samples appear, and can be penalized more or less for certain categories depending on how strict the classification needs to be for certain categories. In the future, this algorithm has the potential to predict which categories a business review belongs in with a high enough accuracy if we are able to train on the majority of the categories from the dataset if we had the computational power to do so.

Word Count: 2421