Auto-Categorization of Businesses on Yelp.com

Jason Fennell, Karen Shiells, Bharath Sitaraman
{jfennell, kshiells, bharath}@stanford.edu

June 5, 2009

Abstract

We built a system to infer semantic categories of businesses (Auto-Categorization) from Yelp.com based on business titles and reviews. Auto-Categorization is accomplished by training a Naïve Bayes classifier on labeled word frequencies (supervised learning). We extend this classifier by using Expectation-Maximization to bootstrap off of unlabeled word frequencies (semi-supervised learning). Given more time, we would have liked to look at adding named entities and chunking of words with high mutual information as additional features to our model.

1 Introduction

Yelp.com is a website that aggregates user reviews of local businesses ranging from restaurants to flower shops. One of the ways of organizing businesses is through categories: a hierarchy of semantic tags that correspond to the primary purposes of a business. For example, "Back A Yard" is a Caribbean BBQ restaurant in Menlo Park, CA. It is categorized as "Caribbean" which is also a subcategory of "Restaurant". These categories are useful for browsing data and improving relevancy of search results. Unfortunately, a large percentage of the businesses on Yelp remain uncategorized. This paper outlines our work in building a tool that Auto-Categorizes a corpus of Yelp businesses based on information on the business page of each business. Not only does such a tool have great potential for filling in missing categories in Yelp's data, but it could also be used to audit existing category labels.

We approached this problem by assuming that the text of user reviews would be a good predictor of the categories associated with a given business. However, we don't think that the order of the underlying words is nearly as important as their simple frequency counts, that is, we viewed the review and title text of each business as a bag of words. We model the relationship between words and categories using a simple Naïve Bayes model. Now, the fundamental assumption of this model: conditional independence of words given categories (labels) is obviously

violated because categories are arranged hierarchically. We will have more to say on this subject in later sections, but we nonetheless decided to use Naïve Bayes because it is simple and has been shown to perform well in practice despite badly violated assumptions.

While there are many businesses without categories in the Yelp database, there are also businesses that have review text but no category label. However, we would still like to make use of this data if possible. This motivated us to extend Naïve Bayes, which does supervised learning on labeled training examples, to use "semi-supervised learning" to make use of unlabeled data in its training, similar to the methods used in Nigam et al's paper. In addition, we explored several methods of removing low-information words from the reviews and selecting sets of categories consistent with the constraints of the dataset.

The rest of this paper is as follows: Section 2 describes the Yelp corpus on which we performed our training and testing. Section 3 gives more information on the Naïve Bayes model and our stop words and category selection methods, while the Expectation-Maximization extension of Naïve Bayes is described in Section 4. Section 5 contains our conclusions, as well as potential areas for further exploration. Finally, in Section 6 we have listed each of our contributions to the project.

2 Data

Our corpus is a subset of 10,000 businesses from Yelp's full, private corpus. For each business in this corpus we know

- the name of the business,
- the categories of the business, and
- all reviews of the business.

Every business in our corpus is guaranteed to have a title, at least one category, and at least one review (we simulate un-categorized businesses by ignoring the category data inside our classifier). We hold back 1000 businesses for use as a gold standard test set on which we never run any training. We use the remaining 9000 businesses for training, dividing them in various ways as we will discuss later.

mmmmmmmmmmmmmmmmmmmmmtexas de brazillllll arrrghhhh (like homer)
The panninis are outstanding - really recommend them! Coffee wasn't great, because it wasn't freshly brewed apparently. Considering it's a gourmet cafe, we expected it to be perfect. But overall very good expereine and we will definitely be back.
Micheal's attention to detail and artistic vision created many memorable photographs for our family. We were also very appreciative of his willingness to rush our order and get the photos matted and framed just in time for Christmas. Thanks Micheal, Kevin & Heidi

Table 1: Sample Yelp Reviews

One of the major challenges of our task stems from all reviews on Yelp being user-generated content. This means that there is a lot of noise in the data. The reviews shown

in Table 1 provide examples of the range of content in the dataset. The first is barely comprehensible and contains few words that will be useful for analysis. The second is a poorly-written but not a terribly unusual review. Words are misspelled and capitalization is inconsistent. Many reviews contain poor grammar, emotions and ellipsis. Poor grammar in particular motivates our bag-of-words model: we just skip around the issue of grammar completely by only looking at word frequency. We further sidestep formatting noise like emoticons, ellipsis, and capitalization issues with some basic data normalization: we remove all punctuation and special characters, lowercase all strings, and normalize all whitespace to a single space. There are still other noise challenges that are present but we ignore these because they are harder to deal with, the primary one being spelling: movi instead of movie or japanees instead of Japanese. While there is certainly information to be gained from the underlying correct words, it is a whole new ML project in and of itself to do such spell corrections. Instead we hope their effect is minimized by the sparsity of any given misspelling.

An aspect of this problem that is particularly unusual is that our labels are not a simple set of independent categories. Category labels are instead organized into a DAG hierarchy. For example, Dim Sum, Chinese, and Restaurant are all possible categories, but Dim Sum is a subcategory of Chinese, and Chinese is a sub-category of the top-level category Restaurant. Another example is Radio Station, which is a sub-category of the top-level category Mass Media. We are guaranteed that if a business has a category label C, it also has the labels of the ancestors of C. However, we are not guaranteed that the most specific possible category is applied to a given business. For instance if we were to look at the dim sum restaurant "Delicious Dim Sum", it might be labeled with the category Dim Sum (and thus also Chinese and Restaurant), or just the category Chinese (and thus also Restaurant). We will further discuss how we deal with this hierarchy when discussing our classifier.

We dealt with the sparsity of words in our corpus by using Laplace smoothing. Also, we had to deal with sparsity issues for labels as well as words. The distribution over labels in the Yelp corpus is not even remotely uniform. Businesses that are part of the category Restaurants make up about 87% of the corpus, while other categories like Hindu Temple may not be present at all in our sample. One possible extension of this work would be to see if more advanced smoothing methods such as Good-Turing or absolute discounting could improve our results.

3 Naïve Bayes Model

The main classifier that we implemented was a standard Naïve Bayes model with a few modifications in order to work well with the Yelp category tree. In our model each business is stored as list of categories and a list of frequency counts of the words in the title and review text of the business. We do not differentiate the text in the title from the text in the review, but treating title text specially

(perhaps by giving it extra term frequency, perhaps by training separate model parameters for the title) is a logical extension of our work.

3.1 Training

We train this model by iterating over all the labeled businesses in our training set and, for each category label on the business, incrementing a global count of each word on the business and a global count of category occurrences. We currently only increment the total category count by one each time a business has that category, but one possible extension of this work would be to change the size of that increment dependent on the size (number of words) of the business in question. At the end of training we have a global count of how many times each category appeared, $P(C_i)$, and how many time each word appeared in each category $P(w_j|C_i)$.

3.2 Inference

Once the model is trained, the basic Naïve Bayes inference is simple: we choose the most likely category given the words in a business. So if business $b = w_1 \dots w_n$, the most likely category C^* for b is

$$C^* = argmax_i P(C_i) \prod_{j=1}^{n} P(w_j|C_i). \qquad (1)$$

However, this is not enough for our situation: each Yelp business can have multiple labels that form multiple disjoint subsets of the category tree. For instance, a bookstore that also sells coffee might have the labels Coffee and Bookstore, in addition to all of the ancestors of those categories. Initially, we simply selected the single label with highest probability. It quickly became clear, however, that selecting one label would not provide adequate coverage of the set of labels, and that simply setting a threshold could produce inconsistent label sets with, for example, missing ancestors.

Instead, our approach is to perform a hierarchical search for labels to apply. We first calculate the probability of each category independently, which is just the formula inside the argmax of Equation 1, for each category. We then do a modified breadth-first search of the tree starting with the top-level categories. Beginning with the dummy root node which we treat as the parent of all of the top-level nodes, each node examined is added to a queue. When it is popped from the queue, the probability for each of its children is compared to a threshold. If any of those have a probability greater than a threshold we set, then we add them to the label set for this business and them to the queue for later expansion of their children. Any node whose probability does not exceed the threshold is not added to the queue for expansion, and thus its branch of the tree is pruned.

We have also done some work to make the thresholds a function of the depth in the tree. Because the top-level categories occur at least as frequently, not only in the data as a whole but with individual words, than their descendents, the probabilities are skewed towards those. Having observed, after switching over to our hierarchical cate-

gory selection model, that we could either set the threshold high and receive as output, for example, only the tag"Restaurants" for a business that was originally labeled "Restaurants, Chinese" or set the threshold low and have the same business tagged "Restaurants, Nightlife, Food, Health and Medical,..", we looked for a way to use a higher threshold with the top-level classes and a lower threshold at deeper levels of the graph. In order to combat this problem, we tried multiplying the threshold by a scaling factor greater than one for each level the classifier progressed through the categories. Unfortunately, this produced only very limited benefits with the right parameters, and with the wrong ones produced labelings such as, for the example above, "Restaurants, Chinese, American(new), American(traditional), Vietnamese, Italian,...", favoring the deeper categories now over the top-level ones. While scaling did somewhat help to improve performance when the base threshold was higher than optimal, tuning that parameter produced greater gains than using the scaling factor.

3.3 Stop Words

We also discovered that, despite the limited information that such words provide, that our original model was assigning more importance to function words ("and", "the", "a") than content words ("food", "beef", "service"). In order to limit this, we decided to remove stop words from our model. To do this, we found a list of stop words from textfixer.com and took those words out of the business reviews and titles before the frequencies were tallied. Later, we explored other means of creating lists of stop words, but because none of those methods produced promising lists we did not end up running our model on the alternative sets. Some representative words are shown in Table 3.3 with their frequencies in the dataset, calculated information with categories (described below), and presence in or absence from the stop word list.

Word	Count	Information	Stop
the	396017	2.81	yes
and	261578	3.5	yes
i	243547	3.86	yes
food	33621	3.73	no
be	32208	5.49	yes
service	16682	5.77	no
also	15562	6.47	yes
fun	4601	7.93	no
these	4540	8.73	yes
rice	4531	5.13	no

Table 2: Sample Word Counts and Information

Our first method of collecting stop words was by raw frequency counts. That method, however, quickly proved to be unreliable when "food", an obviously relevant word in a dataset where identifying restaurants is an important measure of success, with a raw count of 33,621, came in above words like "be" and "also". Setting a threshold that would leave "food" in would only have removed about 20 words, all of which were al-

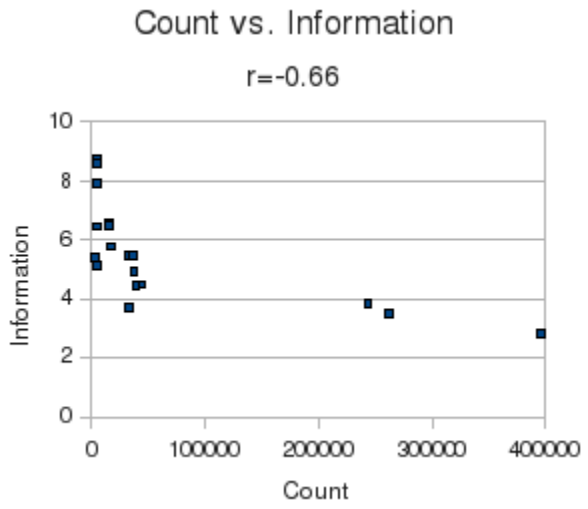

Figure 1: Word Frequency vs. Information Score

ready present in the internet stop list. The correlation of a sample of word counts with word presence in the stop list was only $r = 0.44$. In order to construct a more reliable list, we switched to using a mutual information metric, approximating information gain by summing the mutual information of the word with each of the categories and then, in theory, collecting the words with the lowest information scores as stop words. Unfortunately, this list also failed to produce useful results. While the lowest information scores in the dataset did indeed belong to stop words, "food" was still high in the list. In fact, counts and the information score had a correlation of $r = -0.66$, as shown in Figure 1. On the same sample of words, the information metric had even less correlation with the stop word list than the raw counts, with $r = -0.28$. Because these attempts at stop word extraction were unsuccessful, we continued to use the internet stop word list, which produce useful results.

All of this work was motivated by the across the board improvement we saw in our results by adding stop word removal, which can be seen in Table 3. Additionally, the removal of stop words improved the speed and memory usage of our code, so we used stop word removal for all other experiments we discuss in this paper.

Stop Words	Prec	Recall	F1
No	.569	.266	.363
Yes	.607	.286	.389

Table 3: Naïve Bayes on 5000 businesses with and without stop words

3.4 Results with Naïve Bayes

We evaluated our model by having it infer labels for our test set comparing the predicted categories to the categories we know are correct in terms of precision, recall, and F1 measure.

As shown in Table 4, the Naïve Bayes model produced vastly varying results on different categories. For categories like Restaurants and Shopping, it achieved remarkably high recall. This, unfortunately, was achieved at the expense of precision, especially in the case of Shopping, because having been trained on a corpus strongly skewed towards restaurants, it had a very high prior probability for those frequent categories and tends to

Category	Count	Guessed	Prec	Recall	F1
Restaurants	8715	12022	.66	.91	.77
Shopping	3700	7986	.4	.87	.55
Food	3590	3778	.5	.52	.51
Beauty and Spa	1595	1396	.65	.57	.6
Nightlife	1430	2215	.34	.53	.42
Local Flavor	690	94	.14	.02	.03
Fast Food	435	0	0	0	0
Desserts	240	9	.11	0	.01
Religious Orgs.	130	4	1	.04	.06
Pediatricians	5	0	0	0	0
Hindu Temples	0	0	0	0	0

Table 4: Naïve Bayes Results by Category

dominate other categories when the word information does not distinguish strongly. This could perhaps be compensated for, in further work, by adjusting the relative weighting of the prior and posterior probabilities in the category estimates, but the best way to compensate would be simply with more data. While an overwhelming portion of the dataset carried the Restaurants label, the majority of the categories behaved more like Fast Food, Desserts, Pediatricians, and Hindu Temples, which had relatively few examples in the dataset and were never or very rarely applied by the classifier. These categories all achieved zero or near-zero recall and, even if applied, mostly had zero precision as well. There were, however, a fair number of exceptions, such as Religious Organizations, which either by chance or by virtue of using distinctive vocabulary were applied correctly and received high precision, though still low recall. For the categories like Pediatricians that occured very few times, it is reasonable that they would not be guessed, particularly since it is very possible that they would all end up in the test set for a particular run. Fast Food, however, did have a decent number of examples and should have been learnable.

One major problem we noticed was the conflation of categories that are in similar. The most intuitive of these to understand is the conflation of the categories Restaurant and Food. Restaurants contains restaurants, for example a Chinese restaurant or Indian restaurant. Food contains places that serve food but aren't necessarily restaurants, breweries, grocery stores, coffee shops, etc. Since the focus with both categories is largely on food, they have very similar words and are thus easily conflated. We think that, given a larger data set, looking at word bigrams or other structures that are sparse but high in information would help address this.

We also performed experiments to determine the optimal parameters for category selection. We discovered that the system achieves the best F1 score with a base threshold of -3 applied to the log probabilities, corresponding to a probability threshold of about 0.05, as shown in Figure 2. After determining that, we then experimented with the scaling parameters using both the optimal threshold of -3 and an alternative one of -2, corresponding to about.14, to see how the scaling affected performance. Results for the -2 threshold, shown in Figure 3 showed some improvement with scaling, particularly at a scaling factor of 1.5. For the -3 threshold, performance was less affected, as shown

in Figure 4. This may reflect the scaling factor helping to compensate for the overly high base threshold in the -2 experiments, since changes in the base threshold seem to affect the deeper levels of the graph most strongly.

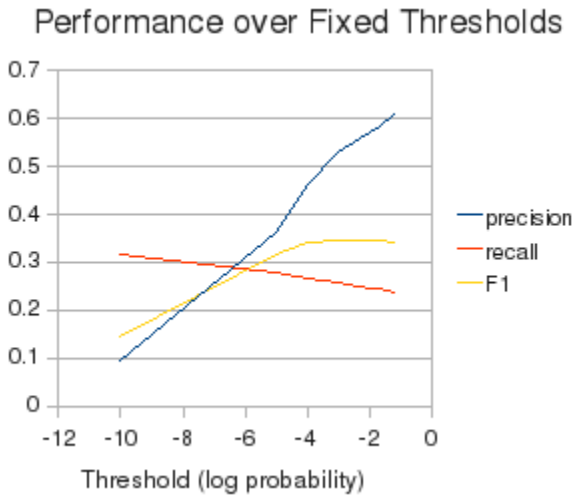

Figure 2: Fixed (log) Threshold

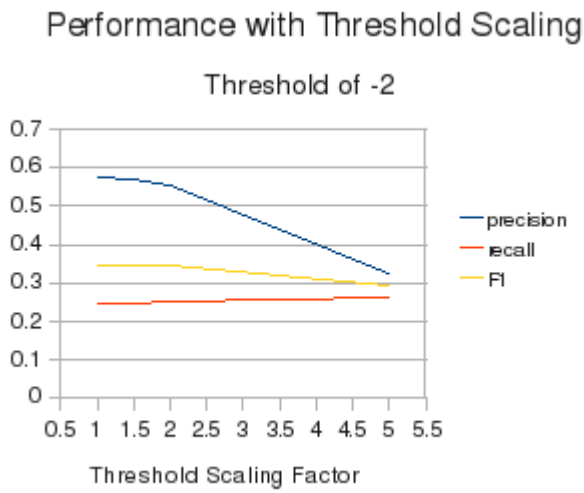

Figure 3: Scaled (log) Threshold, Base of -2

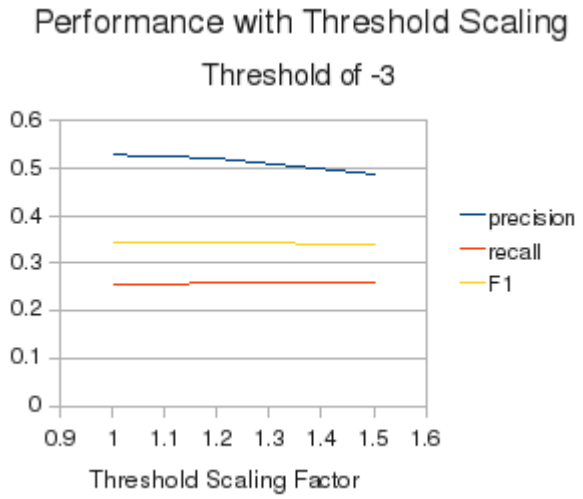

Figure 4: Scaled (log) Threshold, Base of -3

4 Semi-Supervised Learning

We wanted to see if we could leverage uncategorized but reviewed businesses to improve the accuracy of our categorizer, so we decided to extend our Naïve Bayes model by doing semi-supervised learning. In semi-supervised learning we first train an initial model on a small, labeled data set (A). That model is then used to label a large, unlabeled data set (B). Then the model is re-trained using both

the original data (A) and the model-labeled data (B) . This labeling/retraining is then iterated until it converges to a local maximum (which it is guaranteed to do because this is an instance of Expectation-Maximization).

4.1 Results

#Label	#Unlabel	Prec	Recall	F1
100	-	.381	.215	.275
100	1000	.451	.199	.276
100	2000	.459	.197	.276
500	-	.483	.241	.322
500	1000	.478	.218	.300
500	2000	.480	.217	.300

Table 5: Semi-Supervised Learning Results

We were limited in our experiments with semi-supervised learning by our data and the system we had built. Ideally we would have run semi-supervised learning experiments with with unlabeled sets of sizes that are fixed multiples of the size of the labeled data. Unfortunately, we ran into memory limitations for data sets of more than about 3000 businesses. Given that limitation, the data in Table (5) is all we were able to gather. In these cases our system is even more severely weighed down by sparsity issues. Many categories are not even seen! However, we draw what conclusions we can.

First, consider the runs in with 100 labeled businesses. Our baseline run is our best naïve bayes model with no expectation-maximization, which achieves precison of .381, recall of .215, and F1 of .275. Adding in a 1000 unlabeled businesses has a positive effect on model precision, boosting it from .381 to .451. However, there was a similar negative effect on recall, which meant that the overall F1 stayed just about the same. Doubling the number of unlabeled businesses to 2000 had essentially no further effect. We hypothesize that the addition of the first 1000 unlabeled businesses only helped because the original data set was so tiny that any and all information helped. However, unlabeled businesses are much less helpful than labeled ones, so even doubling the unlabeled data size to 2000 didn't provide much of a positive effect. This is also bourn out by the essentially non-existent change between the results of training of model on 500 labeled/1000 unlabeled and 500 labeled/2000 unlabeled.

Now, consider the runs with 500 labeled businesses. We see that the addition of unlabeled data actually *hurt* the performance of the model. This is because labeled and unlabeled businesses are weighted equally in the retraining of the model, so the unlabeled businesses just end up introducing noise. In order for unlabeled businesses to actually help classification there needs to more unlabeled than labeled businesses by far more than a factor of two (at which point the sheer fact that you are increasing your training set by so much helps you). We could downweight the unlabeled data by adding their word counts as $\lambda < 1$ instead of 1 to try to decrease the influence of the unlabeled data in the 500-1000 and 500-2000 cases.

Our conclusion from Tabel (5) is that semi-supervised learning is not a valuable addition

to our system. Perhaps if we had a much much larger corpus of unlabeled data it would be helpful, but in the real Yelp data set the number of reviewed by uncategorized businesses is not multiple orders of magnitude higher than the number of categorized businesses. We are much better served by putting effort into improving the way our model uses the labeled data that it has access to.

5 Conclusion

In this paper we build a system to autocategorize businesses drawn from Yelp.com com using a naïve bayes model based on their title and reviews. We also explored extending the data set of this classifier by using semi-supervised learning. We found that this model does hold some promise: we were able to achieve good F1 scores over certain very common categories such as Restaurants. However, the vast majority of categories in our small subset of Yelp's data are extremely infrequent and we simply do not have the information to accurately classify them. We believe that further improvements to our results would come from two major sources. First, simply having more labeled data, having more examples of less frequent categories, would be incredibly helpful. Second, coming up with better ways to suppress frequent categories like Restaurant so that we can surface things like Hindu Temples.

6 Contributions

- Jason got all the data from Yelp and additionally wrote all of the data-reading and data-cleaning code. He also architected the system and wrote the Expectation Maximization module. After the main code base was set up he spent a lot of his time trying to tune the code so that it was faster and less memory hungry. Given that he wrote the EM module, he also ran the experiments that used the EM module (which went along with the memory optimizations). Jason also did the LaTeX formatting of the final report after reorganizing and adding to it after Bharath's first draft of the report.

- Karen wrote the initial Naïve Bayes implementation and worked on category selection, including threshold experiments and threshold scaling, and both the collection of stop word lists and the code in the system to use them. She also helped with adding details to the evaluation code and wrote the corresponding sections of this report.

- Bharath set up the CVS repository, ran a few data experiments, and wrote the first draft of the paper.