**Categorization of Businesses:**

**Comparing Text and Metadata Classification**

**Charlotte Chen Dan Hui David Neiman David Tsui**

chc190@pitt.edu dth26@pitt.edu dsn9@pitt.edu dat83@pitt.edu

**Abstract**

Naive Bayes and logistic regression were used to predict the primary categories of businesses. This was facilitated using the business and review data from the Yelp Data Challenge Set.

The Naive Bayes model had a precision average of 25.24% and a recall average of 21%. The logistic regression model, in contrast, was much more effective, producing an 85.29% precision average and a 41.12% recall average.

**Introduction**

Text classification is a common problem in natural language processing. The ability to properly categorize text allows humans to group pieces of text together and provide additional context into the text. With the increasing availability of metadata and textual information, the ability to classify text documents has improved and become more visible within different real world applications.

For our text classification problem, we were interested in identifying the type of business of a retailer or store, given the reviews that were written regarding the business. This would allow us to analyze the differences in reviews by category, and allow us to make conclusions about how people review businesses in each category. For our dataset, we used the Yelp Data Challenge to parse the text reviews, producing all the reviews for a given business for each business within the dataset. As comparison against our text classification, we will use the metadata of business attributes to classify businesses as well. This will give us insight into seeing how the type of business can affect the meta-data found in Yelp businesses.

To classify the Yelp businesses by the type of categories using text reviews, we began by pre-processing the data, taking out unnecessary words and stemming the remaining words. This was accomplished using both a custom-made Python program, as well as the PorterStemmer package.

For our text classification model, we implemented a multinomial Naive Bayes model to predict the business categories given reviews. We used the term frequencies that we aggregated while iterating over the review files, and sorted by category. Using the normalized term frequencies, we ran the Naive Bayes model and implemented smoothing to make sure that every word was represented.

For our metadata comparative classification model, we implemented a logistic regression classifier using the attributes of businesses as features. We implemented Python scripts to produce ARFF files that will aggregate all the attributes of a business and produce vectors for each given business. We then inputted that data into Weka, which will produce predictions on the business categories, given the vector of attributes it has compared against the business features.

As evaluation, we used precision and recall on the number of categories correctly guessed and the categories that the business actually has. The five-fold precision of the Naive Bayes model was 25.24%, with 21% recall. The five-fold precision of the logistic regression model was 85.29%, with 41% recall. Our results demonstrated that the logistic regression model was considerably more effective in classifying the text reviews, but both methods were considerably better than random guessing. Thus, we demonstrated that analyzing word frequencies was an effective approach in classifying the primary categories of the Yelp reviews.

**Background**

Our work mainly draws upon the idea of text categorization, and its implementation in the context of less structured, informal data such as the Yelp reviews.

Recent approaches toward text classification use variations of first order probabilistic models that make the Naive Bayes assumption of independence relationship between words and binary word features. The Naive Bayes assumption assumes that all attributes are independent of each other within the context of the class. As Friedman, Domingo and Pazzani point out, the Naive Bayes assumption works well in real world applications, despite its simplifying assumptions. This is mainly because the classification estimation is estimating only on the function of sign, and not on the actual function approximation. This means that the accuracy of the classifier can be high, even though the classifier doesn’t model the actual relationship.

Two prominent Naive Bayes models, the multi-variate Bernoulli model and multinomial model, are often used when approaching the problem of text classification. The multi-variate Bernoulli model is based off of binary data where every token in the feature vector can be a 1 or a 0. The feature of the vector is each word within the vocabulary. Comparatively, the multinomial model uses term frequencies instead of a 1 or 0 for each word within the vocabulary.

McCallum and Nigam analyzed the Bernoulli model and multinomial model, and compared the two models against text documents of varying vocabulary sizes. McCallum and Nigam tested the models on Yahoo! Articles of different categories and discovered that the Bernoulli model works slightly better when the vocabulary size is smaller (10~100), while the multinomial model generally works slightly better when the vocabulary size increases.

Given McCallum and Nigam’s research, we decided to base our text classification algorithm on the multinomial Naive Bayes. There are many businesses within the dataset, each containing a number of reviews. As a result, we anticipated working with a very large vocabulary set.

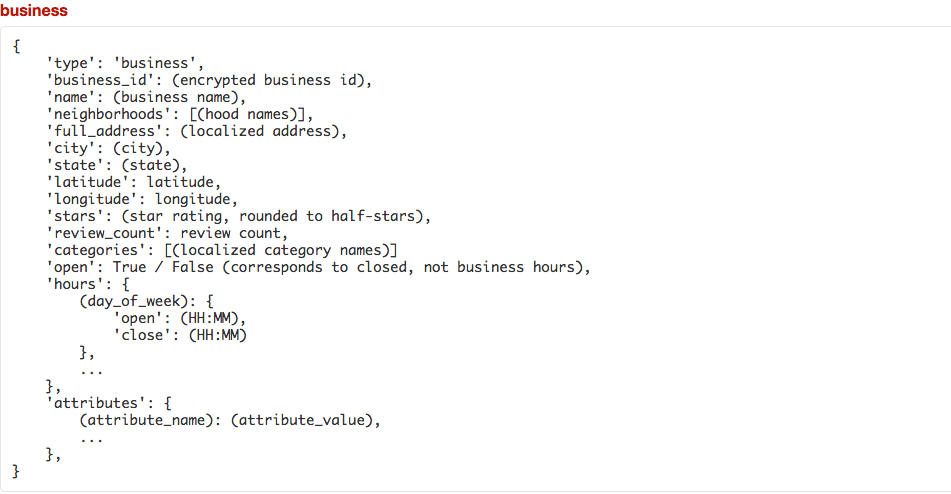
In processing the Yelp review set and developing our text classification models, one major decision we had to make was whether to restrict reviews to one classification, or attempt to predict one or more categories for a given review. While binary classifiers are well studied and consistent, multi-class classifiers are less consistent. Aly discusses how to decompose binary classifiers into multi-class classifiers for different machine learning algorithms. One of the simplest approach discussed was the use of One-versus-all (OVA), which treats each class within the multi-class classifiers as binary classifiers. The class that produces the top k output is then considered.

Using OVA, we will thus attempt to classify businesses using multinomial naive bayes and logistic regression.

**Data**

In our experiment we will be using the data from the Yelp Data Challenge Set. Specifically the yelp\_academic\_dataset\_business.json file and the yelp\_academic\_dataset\_review.json file.

The business file was formatted with a unique business id, information such as address and number of reviews, and then a list of attributes. Some attributes, such as "Good for Dancing", were given true or false values, while attributes such as "Alcohol" were given more descriptive values. Other attributes, such as ambience, contained even more nested information, such as information about being trendy or romantic. From the business file, we will be extracting the information of business\_id, categories, and attributes.



The review file contains the unique user id, review id, business id, information about the type of votes given, and the text review. From the review we will be extracting the business\_id of the review and the text, aggregating all the text reviews of a user to its respective business\_id.



**Pipeline Framework**

To approach the experiment, we used a standard machine learning pipeline of first preprocessing the data. Then, we split the data into training and testing. After the data split, we will apply logistic regression on the attributes of the business and Naïve Bayes on the text reviews of the business to classify the businesses by its categories. Both will output predictions on the set of test businesses and evaluated against its actual categories.

Because of the size of the data, we have to output the data into a text file at each step and feed it back in to the next step.

**Preprocessing**

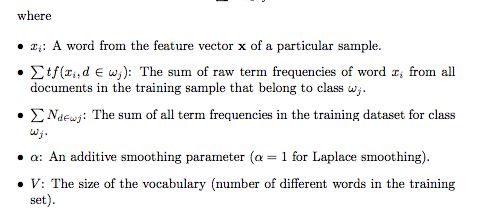
We began by preprocessing the text reviews, looking to remove unnecessary words and symbols in the text. This is done with a script that looks for specific stop words and removing the stop words from the sentence. The preprocessing step involved simply using porter stemming to stem and lemmatize all the text reviews of each business once the words have been tokenized.

As part of the preprocessing step, we also considered compressing the size of our classes. We discovered the size of the classes to be around 698 possible categories for a business to belong to. In addition, since the categories are not mutually exclusive, the variations of categories a business can belong to is even higher. To compress the size of our class, we utilized a given hierarchical relationship provided by Yelp which groups the categories into primary, secondary, and tertiary relationships. Primary categories will thus encompass all secondary categories, which in turn encompasses all tertiary categories. Thus we preprocess the data by keeping only the primary categories within all businesses.

**Multinomial Naïve Bayes Unigram Model**

To implement the multinomial Naïve Bayes training model, we find the term frequencies of each text of each business within the training set. For each business’s category, we aggregate the term frequencies of the the business under the category. This approach allows for us to obtain the term frequencies of each vocabulary given a category.

We then computed the maximum likelihood estimate for each of the categories in the Yelp data set as follows:

Macintosh HD:Users:dneiman11:Desktop:Screen Shot 2016-04-16 at 10.19.16 PM.png

Thus for each word in the test business, we find the probability of each word occurring given each category within the Yelp Dataset. This probability is calculated by finding the given word’s term frequency in each of the category divided by all the words found in the training dataset coupled with smoothing.

Macintosh HD:Users:dneiman11:Desktop:Screen Shot 2016-04-16 at 10.22.11 PM.pngWe can find the most likely class for each test business’s text by using chain rule to find the total probability of the whole text given the category and finding the most likely category from the produced probabilities.

Since the businesses can have multiple categories, we will have to manually find and apply a threshold over the expected probabilities produced from each classifier and find the probable categories an instance can belong to once we all the instance has been classified by each category.

**Logistic Regression**

For our logistic regression implementation, we wrote a script using the Weka package to help train and test business ids on the features of business attributes. Weka uses the Attribute-Relation File Format (.arff), to describe the structure of the data. The ARFF file models the logistic regression vector where each different business attribute is a feature within the vector and all the possible nominal values for the features are defined. Then, for each training and testing instance, the corresponding value is listed for the feature vector.

Below is an example of the header section, defining the possible business attributes and its nominal values. For each category, we create a separate ARFF file with the given attributes and the binary class values of 0 for an instance not belonging to the category and 1 for an instance belonging to the category.

@RELATION business

@ATTRIBUTE Drive-Thru {False,True}

@ATTRIBUTE Alcohol {beer\_and\_wine,none,full\_bar}

@ATTRIBUTE category {False,True}

…

@ATTRIBUTE class {0,1}

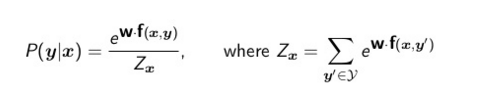
Below is an example of the data section, where each line is an instance of the training data with values corresponding to the attributes:

@DATA

False,none,False….,0

True, full\_bar,False,…,1

The conditional probability of a category given the text review, Pr(category | review text) is defined as follows:



The logistic regression model was designed to maximize the log probability of the review text belonging to a given category - ie. Pr(category | review text):

Macintosh HD:Users:dneiman11:Desktop:Screen Shot 2016-04-17 at 1.01.45 AM.png

After generating ARFF files from the business data set for each category, we use OVA to find the most likely probability for each category classifier, similar to how we try to find the most likely probability for each category classifier in Naïve Bayes. Like Naïve Bayes, we have to apply a threshold to find the multiple categories a given instance can belong to.

**Experimental Results and Discussion**

In our experiment with using multinomial Naive Bayes and logistic regression to classify two sets of features, we ran our development data through the parameter of filtered hierarchy vs non filtered hierarchy categories of businesses and the parameter of threshold value in our OVA machine learning classifier.

Our development data that is used by the two models feature 5,000 businesses along with all the preprocessed text reviews and attributes that belong to the businesses. We split the testing and training of the 5,000 businesses by business ID, producing 80% training and 20% testing.

To test our first parameter, filtered hierarchy vs non-filtered hierarchy of categories of businesses, we take the same training and testing businesses and produce one set with its categories flattened out to only the 22 primary categories and another with all the categories of the businesses.

Figure 1. Side-by-side comparison of precision and recall of Naive Bayes model given hierarchical filtering

Our results show that the Naive Bayes model performed better with hierarchical filtering, with a roughly 20% increase in precision, and 200% increase in recall. Both models still produce relatively low precision and recall, but hierarchical filtering is clearly shown to be a useful procedure in classifying categories.

These results match up with what we expected prior to running the experiment with the multinomial Naive Bayes model. Without hierarchical filtering, we would be estimating among categories within the same primary group, such as between Chinese and Taiwanese restaurants. This is a significantly more difficult problem than differentiating among an automotive shop and a hotel, for example, and we were surprised to get 13.4% precision without hierarchical filtering. This suggests that the conditional probabilities for particular words in a review actually tend to be somewhat unique among different types of businesses, such that we could make accurate predictions over hundreds of categories with over 10% precision.

We then compare the parameter of hierarchical filtering on logistic regression for the same data.

Figure 2. Side-by-side comparison of precision and recall for logistic regression model given hierarchical filtering

These results are more pronounced than for the multinomial Naive Bayes model. The logistic regression model produced a 85.29% precision and 41.72% recall when hierarchical filtering was used. Without it, the model only had a precision of 35.53% and a recall of 8.19%. Even more so than with the Naive Bayes model. Like the Naïve Bayes, this result suggests that the attributes are better at distinguishing business types that are more different than business types that are more similar.

To test our second parameter, threshold range, we ran the same 5,000 businesses under both learning models with filtered classifiers for both models. Any classifier that produces a normalized distribution score more than the threshold is automatically counted as positive for the given instance.

Figure 3. Precision and recall over the threshold range for the logistic regression model. Note that the threshold ranges from 0 to 1, inclusive, and is used to determine whether a text review belongs to a given category.

As we expected, our precision increased as the threshold increased; a higher threshold means a greater probability of belonging to the given category, so it is reasonable that a higher threshold results in a higher proportion of our estimates being correct. Our recall goes down, however, because the greater threshold leads us to reject some of the possible candidate categories. This became more significant as we kept increasing the threshold, reaching 28.5% at a threshold of 0.9, and 1.3% at a threshold of 1%.

Another issue for our experiments is that we used a small development file, which included only 515 categories. Thus, we collected less information about the words and term frequencies within the dataset, and also collected no data for 183 of the 698 categories contained in the complete Yelp dataset. When using this file to determine parameters such as the threshold to qualify for a category, we will miss some information and not classify correctly.

Figure 4. Precision and recall over the threshold range for the logistic regression model. Note that the threshold ranges from 0 to 1, inclusive, and is used to determine whether a text review belongs to a given category.

At 0 threshold, the recall is 100% and the precision is 5.79%. After 0.1 threshold, however, our precision and recall plateaus and remains the same at precision of 17.59% and recall of 14.59% despite increasing threshold. The data at 0 threshold makes sense because when there is a threshold of 0, all categories are predicted to be the category of a business instance, resulting in 100% recall and low precision. However, the results afterwards differed greatly from what we came to expect. Like the logistic regression model parameter, we expected the precision and recall to follow an inverse curved relationship. This data seemingly suggests that the threshold does not affect the precision and recall of the Naive Bayes algorithm.

Upon further analysis, we found the reason for such occurrence to be because our script for Naive Bayes calls for the argmax of category when the threshold filters out all the predictions. As it turns out, each classifier produces a probability of a positive category of less than 10%. As a result, all precision and recall values remain the same after the 0.1 threshold.

This result means that each classifier category under the Naive Bayes model is very similar. However, the slight differences between each classifier allows for the precision and recall to be higher than expected, despite the similarities. We believe that this result is due to the fact that a lot of the reviews contain very similar unigram words. As a result, many of the classifiers are closely associated. However, there are still enough key words within each review to distinguish between the businesses. One possibility for improving the precision and recall might be to apply IDF weighting to our Naive Bayes model. IDF would differentiate each classifier more, and allow for higher thresholds to be applied to our data.

Taking the best of both parameters, we applied the parameters of filtered hierarchy and optimal threshold onto a larger test dataset of businesses. We used our test data of 20,000 businesses and the corresponding reviews as training and testing data. Because there is a lot more data, we decided to apply a 5-fold cross evaluation on the data to ensure that the results are consistent, splitting the businesses 80% training 20% testing.

Figure 5. Side-by-side comparison of the precision and recall measures for the Naive Bayes and logistic regression models.

For logistic regression, we applied a threshold of 0.3 and filtered hierarchy to the learning model to produce a very good result of 85.29% precision and 41.12% recall. For the Naive Bayes, 25.24% precision and 21% recall is produced from the same data. In both the Naive Bayes and logistic regression models, this procedure led to an improvement in results. This is most likely due to an increase in the number of data to train and test with.

Like the previous experiments with parameters however, the logistic regression has consistently outperformed the Naive Bayes. Such occurrences are likely due to the fact that the features of the logistic regression are better determinants than the term frequencies of the Naive Bayes. This result is expected because the features selected are very specific to the type each business has. For example, having the attribute of “Alcohol” narrows a lot of the categories a business can possible become. Given all 72 features and a large dataset, the logistic regression model becomes very powerful in predicting the category a business belongs to.

**Conclusion and Future Work**

Our experiments showed that the logistic regression model was effective in estimating the primary categories of Yelp reviews. With a precision of 85.29% and a recall of 41.12%, the logistic regression model considerably outperformed the multinomial Naive Bayes model, which had a precision of 25.24% and a recall of 21%. Again, this result suggests that the term frequencies alone are not as good of a determinant for categories as the metadata attributes the businesses have.

Additionally, we were able to conclude that hierarchical filtering was important in reducing the number of categories that can be classified, improving the precision and recall.

To improve our experiments, we would greatly benefit from using a more extensive set of data. We were using reduced datasets in the interest of time and performance, but using more data would allow us to make more definitive conclusions. We would be able to identify more words that belong in the reviews, and would also be able to come up with more stable estimates of term frequencies when considering millions of reviews covering a diverse list of categories.

Additionally, as mentioned before, we can use the inverse document frequency to further differentiate among categories, thus allowing us to use higher thresholds.

While we focused on individual words and term frequencies in our approach, we would like to consider other information contained within the reviews, such as the number of stars given in the review, and location information such as the city and state. These would allow us to classify reviews with greater context, to hopefully improve our precision and recall.

Including more review data would also give us the opportunity to explore other questions about the trends in the data and allow us to further extend our use of text classification. For example, analyzing the number of stars would allow us to consider questions such as variations in ratings among businesses of different categories. Alternatively, incorporating the location information into our analysis could allow us to break down the problem by regions. This would open up many possibilities of dividing the data to see if the performance of our text classification models would improve.

**References**

Peter L. Bartlett and Ambuj Tewari. On the Consistency of Multiclass Classification Methods. <http://www.jmlr.org/papers/volume8/tewari07a/tewari07a.pdf>

Pedro Domingos and Michael Pazzani. On the Optimality of the Simple Bayesian Classifier under Zero-One Loss. <http://www.ics.uci.edu/~pazzani/Publications/mlj97-pedro.pdf>

Nir Friedman, Dan Geiger, and Moises Goldszmidt. Bayesian Network Classifiers. <http://www.cs.technion.ac.il/~dang/journal_papers/friedman1997Bayesian.pdf>

David D. Lewis. Evaluating Text Categorization. <http://www.aclweb.org/anthology/H91-1061>

Andrew McKallum and Kamal Nigam. A Comparison of Event Models for Naive Bayes Text Classification. <http://www.cs.cmu.edu/~knigam/papers/multinomial-aaaiws98.pdf>

Sebastian Raschka. Naive Bayes and Text Classification. <http://sebastianraschka.com/Articles/2014_naive_bayes_1.html>