**Text Categorization of Businesses**

**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. <Results here>

**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.

We were interested in classifying text reviews by primary category. 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. We used data from the Yelp Data Challenge Set to parse the text reviews, as well as the businesses and the appropriate categories for these objects.

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. Then we reduced the 698 Yelp categories into the 22 primary categories, so that we could place each review in a mutually exclusive category. We iterated over the words in each document, keeping track of word frequencies for each category.

We implemented a multinomial Naive Bayes model to predict categories. 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 Laplace smoothing to make sure that every word was represented.

We also ran a logistic regression model. We implemented a Python program to convert our review and business category data into ARFF format, and inputted that data into Weka. We used a ten-fold cross validation to evaluate the data using the logistic regression model.

**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.

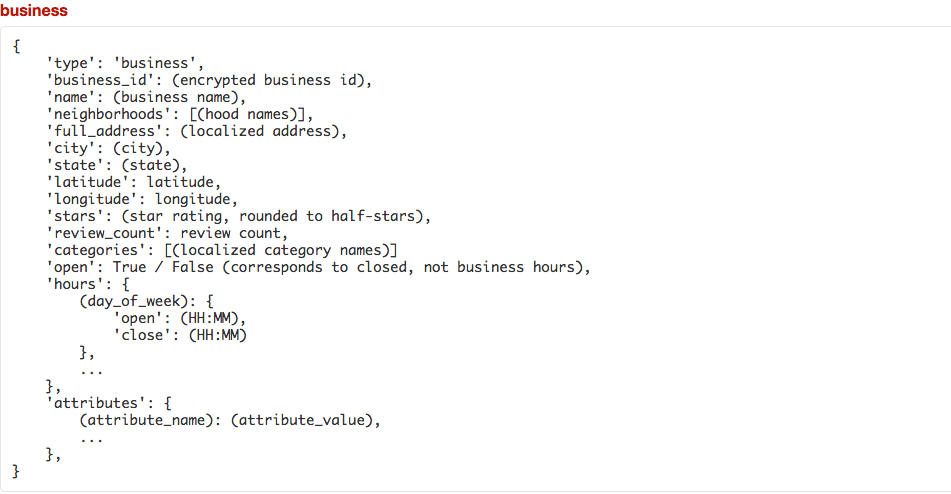
Lewis establishes three key measures of effectiveness for text categorization: recall, precision, and fallout. He notes that in the general case, recall can be problematic when used as a sole indicator of effectiveness; if we generate a model that mostly produces affirmative results, we will get a high recall despite obvious flaws in the experimental approach and low precision. Likewise, precision can be problematic when used as a sole indicator of effectiveness in the opposite case; if we generate a model that mostly produces negative results, we can achieve a high precision from correctly classifying most instances as not belonging to a particular category, but this would also produce a low recall from not identifying the correct category matches. For this reason, we chose to use precision and recall to evaluate our models.

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. Lewis discusses flaws in past classification studies from estimating over multiple categories. For example, researchers have made assumptions in the past about how many categories a particular instance belongs to. Assigning the top-k categories is another approach that has been tried in the past. These approaches don't fit well to analyzing a review data set with an arbitrary number of categories, split among three levels. We thus went with limiting our reviews to one category, to prevent possible errors from the varying number of categories and assumptions about the number of categories to expect. This also allowed us to keep the categories mutually exclusive, instead of containing overlapping categorizations over the three-level Yelp hierarchy.

**Data**

We used 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. One of the most important pieces of information was the list of categories provided. Business were grouped into many categories which were not mutually exclusive. We were specifically interested in the business id, categories, and set of attributes.



The review file contained the unique user id, review id, business id, information about the type of votes given, and the text review. We were specifically interested in the business id and the text review.



**Preprocessing**

We began by preprocessing the text reviews, looking to remove unnecessary words and symbols in the text. We began by removing apostrophes, and then fixing up those words by removing parts of the apostrophe phrase - for example, removing the "ll" in "she'll", or the "ve" in "would've", such that we were left the base word. We also removed extra punctuation, such as periods, question marks, and exclamation marks. After dealing with these, we moved on to tackling the actual words in the review. We removed "to be" verbs - "are", "is", etc. - as well as conjunctions, pronouns, and prepositions. We also removed numbers and additional spaces. Additionally, we used the PorterStemmer package on these edited words to further stem. This allowed us to intelligently group together phrases when necessary to preserve meaning, such as "night-life", while meaningless text phrases like "!!!!!!!" were appropriately filtered out.

**Category Grouping**

The original Yelp dataset contains hierarchical grouping of categories, with 22 primary categories, 498 secondary categories, and 178 tertiary categories, for a total of 698 categories. Using different classifiers for each of these categories would have been unfeasible for both our language model and logistic regression models, so we aggregated the groupings into the primary categories. In the original setup, categories were not mutually exclusive; businesses could be grouped in multiple different categories that could be related. For example, there are businesses in the data set that are placed in three categories: as a restaurant (primary category), as Chinese restaurant (secondary category), and as a Taiwanese restaurant (secondary category). Our grouping procedure categorizes this business simply as a restaurant. In this way, we now assign businesses to one distinct category. Additionally, categories are now mutually exclusive.

**Parsing**

We parsed the business data to keep a frequency of word counts, as well as the business category, the number of words, and the number of reviews aggregated for the business. These were stored in a business object model, which allowed us to maintain our set of businesses in a dictionary.

This word frequency data by business was then used for updating word frequencies by the 22 primary categories that the businesses were split among. These were also stored in a dictionary set and outputted to a JSON file. The separate steps of finding term frequencies over the businesses, and then grouping these frequencies into categories, are redundant because each of our businesses maps to one of the primary categories; in the early stages of this project, we planned to leave businesses assigned to an arbitrary number of categories, and thus this step would have been necessary to attribute term frequencies from a given business to each of the categories it mapped to.

Laplace smoothing was applied to the set of vocabulary found in the parsed review files.

**Language Model**

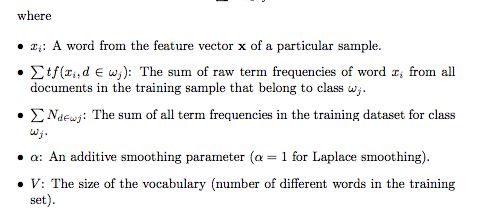
We used a multinomial Naive Bayes model to predict categories. We used the term frequencies acquired from the parsing step, which are written as tf(t, d), where t is the given word or token, and d is the smoothed number of times it appears in the reviews.

We then normalized the term frequencies by dividing each frequency by the number of words, denoted by nd:

Macintosh HD:Users:dneiman11:Desktop:Screen Shot 2016-04-16 at 8.36.52 PM.png

We then computed the maximum likelihood estimate for each of the 22 primary categories in the Yelp data set as follows:

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



Additionally, as part of the Naive Bayes model, the probability of encountering a word or token given a particular category of business can be determined as the product of the probabilities of finding each specific word or token given that category of business:

Macintosh HD:Users:dneiman11:Desktop:Screen Shot 2016-04-16 at 10.22.11 PM.png

This provided us with the predicted category based on the normalized term frequencies. We then implemented wrote a program to determine the precision and recall of these predictions.

**Logistic Regression**

We used Weka to perform logistic regression. Logistic regression is the main reason why we needed to have one category for each business in the data set. If we had left the businesses with an arbitrary number of categories, we could not have generated an arbitrary grouping of categories as a prediction.

Weka uses the Attribute-Relation File Format (.arff), to describe the structure of the data. This is split into two parts: the header part defines the relation, along with the attributes and the types of data they can contain. The second part is the data section, which consists of rows of data following the format specified in the header section.

There was no straightforward method to convert our JSON data into ARFF format. Therefore, we parsed the JSON data and output the file in the ARFF format.

Below is an example of the header section, defining the relations and attributes:

@RELATION business

@ATTRIBUTE businessID NUMERIC

@ATTRIBUTE stars NUMERIC

@ATTRIBUTE category {Nightlife, Restaurant, ...}

Below is an example of the data section, using the above format:

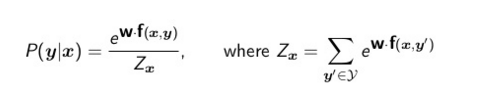
@DATA

1, 4.5, Restaurant

2, 3.4, Nightlife

3, 5.0, Restaurant

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



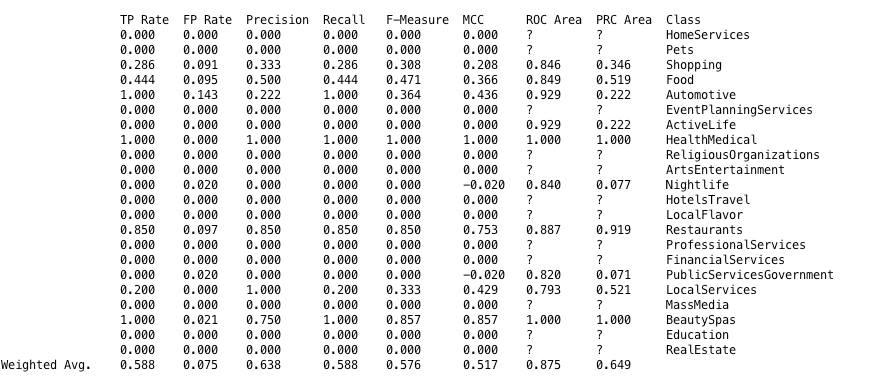
The logistic regression model was designed to maximize the log probability of the review text belonging to a given category - ie. Pr(category | review text):

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

After generating a large ARFF file from the business data set, Weka performed the logistic regression analysis. A ten-fold cross validation was used, and the prediction score, probability distribution, precision, and recall were outputted.

**Experimental Results and Discussion**

The logistic regression model was run on 550 businesses, producing the results in the figure below:



The logistic regression model yielded 63.8% precision rate, 58.8% recall, and

One key finding is that while the logistic regression model overall produced decent results, it was only effective with certain categories. For many categories, such as Hotels & Travel, or Event Planning Services, the model failed to detect the categories and produced 0% precision and recall.

One possible source of error in the logistic regression model would be the relatively small sample size of 550 businesses that we used. This was due to time and performance constraints; we originally tried to use 19,000 businesses, but found that it took a long time to run and thus

**Conclusion and Future Work**

**References**

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

Used sample of Yelp dataset (business and review). probably something like 5% of the businesses and the reviews

Problem Solution: Categorize businesses into categories

-Supervised learning: Language Model and Logistic Regression and Naive Bayes

-Evaluate: ten-fold cross validation. 90/10 split

-Precision and recall based on the number of categories

guessed correctly (precision) and the recall (of those predicted, how many out of the correct ones were selected)

Preprocessing: Using two files,

* business: 5% business
* All in json. general layout = ID (unique), category (how we evaluate), attributes (list of characteristics the business has - ie. parking yes no, mostly true/false, with a few with several choices like dresswear)
* review file: 5% review
* Straight review with customer id, business id, etc. What we need are the business id and the text
* So we get the business
* bid
* cat
* attributes
* 729 categories and that's too many classifiers to both run through for logistic regression and language model - too low accuracy, etc. So we create a hierarchy of concepts
* So we realize a lot of the categories appear flat, they are not actually flat. Such as Taiwanese, Chinese, and restaurant. But these are grouped under restaurant. We originally planned to manually assign it but after brief search found that Yelp provided hierarchical category tree in format of Excel files. So we had to go through excel and find the root category and get rid of the subcategories. We now had 24 categories. **Mutually exclusive!** One business cannot be classified into 2 different categories.
* Also eliminated stopwords. And used lemmatisation.
* So we decreased the vocab
* use those to train, evaluate
* then go back and use 10-fold cross-validation on the big file

Language Model

* Need frequency count of words said given categories
* Use dictionaries in Python with frequency count
* Use a business object
* bid uniquely identifies a business. Then use that for the category's frequency counts
* Originally we had multiple categories. Created category object to handle multiple categories and etc., this became a bit redundant when we simplified to one category per business
* Refer to language model equation

[http://sebastianraschka.com/Articles/2014\_naive\_bayes\_1.html](http://l.facebook.com/l.php?u=http%3A%2F%2Fsebastianraschka.com%2FArticles%2F2014_naive_bayes_1.html&h=aAQENduXC" \t "_blank)

* read that in pdf form
* multinomial equation #37.
* laplace smoothing
* Custom-made precision and recall

Logistic Regression

* Weka uses special file format called .arff file
* We had to convert JSON to arff
* No easy conversion
* features are @relation \_\_\_\_
* such as parking {T, F}
* @attribute
* @attribute
* @class {categories}
* @data [T, T, T, t] [cat]
* each data entry is a business. Array of features. And then what category it is
* generate one large arff file given the business file.
* Used Java to go through the folder of all the arff files
* Gives us the 10-fold cross, the prediction, the probability distribution, precision, recall
* slides for logistic regression
* http://www.slideshare.net/marinasantini1/lecture03-ml4-ltmarinasantini2013
* logistic regression results are only from 550 businesses; 19k businesses took too long
* [**David Tsui**](https://www.facebook.com/david.tsui.56)
* Ok you know what you can write with that data
* so the top thing is with 1 category classifier
* essentially becasue of the really poor precision and recall
* I tried to just not count any businesses with more than 1 class
* and see what happens