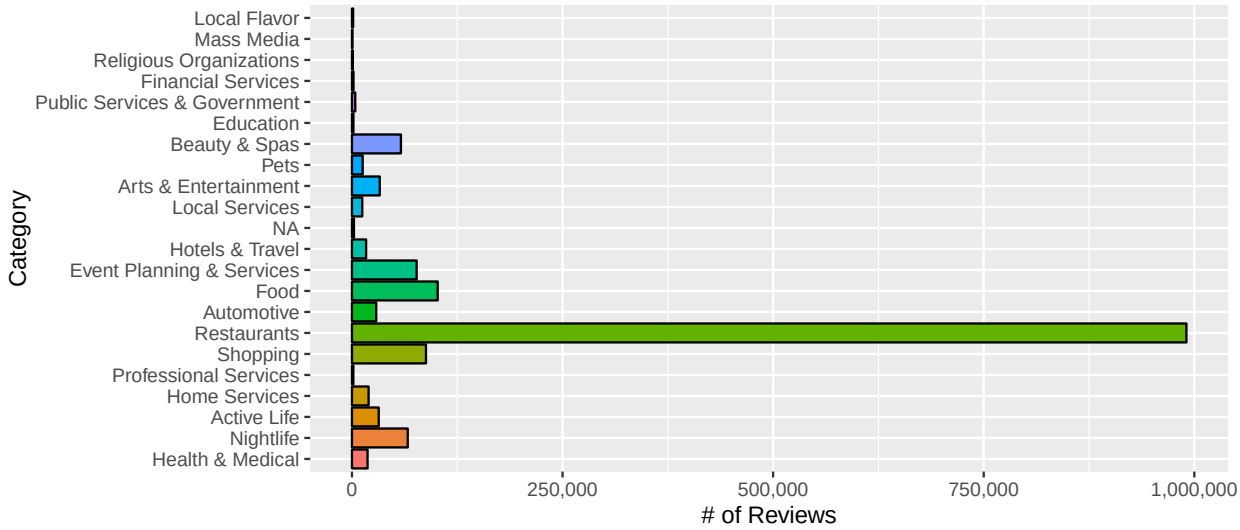
# Sentiment Analysis of Restaurant Reviews

## Introduction

Sites such as Yelp have a star rating system that lets users easily see what the general opinion about a particular establishment is without having to read all the reviews for that particular business. However, there is an abundance of user generated information online from social media platforms like twitter, facebook or blog posts where users express their sentiment about a particular product or business. Since the reviews data on the Yelp dataset provides both the star rating and the text for each review, I wanted to know if it would be possible to train a machine learning model with this data in order to identify if a particular business review was positive or negative based only on its text.

## Methods and Data

The Yelp Dataset Challenge reviews dataset contains 1,569,264 business reviews. The most prominent category for reviews is Restaurants with 990,627 restaurant reviews, so I focused on this category for my project.




A Naive Bayes algorithm was used to build a binary classification model that would predict if the review's sentiment was positive or negative. A Naive Bayes classifier assumes that the value of a particular feature is independent of the value of any other feature, given the class variable. It uses the training data to calculate a probability of each outcome based on the features. One important characteristic of the Naive Bayes algorithm is that it makes "naive" assumptions about the data. It assumes that all the features in the dataset are independent and equally important.

### Preparing the data

In order to get the data in the required format for building the model, I proceeded to load the reviews contained in the *yelp_academic_dataset_review.json* file. For brevity, only the attributes contained in the dataset that are relevant to my project will be listed. It includes the following information:

Table 1: Reviews dataset attributes

| Field | Description |
|---|---|
| business_id | The identifier of the reviewed business |
| stars | The review stars rating (1-5) |
| text | Review text |

Since I only wanted to train my model on restaurant reviews data, I loaded the *yelp_academic_dataset_business.json* file which provides information about the businesses so I could filter the reviews by business category. This file provides information about the business. Again, for brevity, I will only include the relevant attributes.

Table 2: Business dataset attributes

| Field | Description |
|---|---|
| business_id | The unique identifier for the business |
| name | The full business name |
| categories | Category this business belongs to |

Since the business attribute is a collection, meaning that a business can have more than one category, and I only needed the main category, I first made a list of all the different categories on the business dataset, and added a column with the count of occurrences for the category in the business dataset, then for each business I looked up which of the categories it had was the one with the highest count and used that one as my main category. For example, if a business had the following categories *Restaurant, Chinese, Nightlife* and *Food*, if out of these four categories *Restaurant* was the one with more businesses on this dataset, I selected that one as the main category.

After doing this, I ended up with a business dataset that had the business_id and its main category. Then this dataset was joined with the data needed from the reviews. The result was the following dataset and selected only the reviews for the *Restaurant* category.

Table 3: Final dataset used for building the models

| Field | Description |
|---|---|
| business_id | The a unique identifier for the business |
| Stars | The star rating of the review |
| Text | The text for the review |
| Category | Main business category |
| Sentiment | Positive or negative (based on the star rating) |

I wanted to classify the sentiment of the review as either positive or negative, so I decided that any review with a star rating of 1 or 2 would represent a negative review. Those with a rating of 3 or higher would represent a positive review.

**Identifying the features**

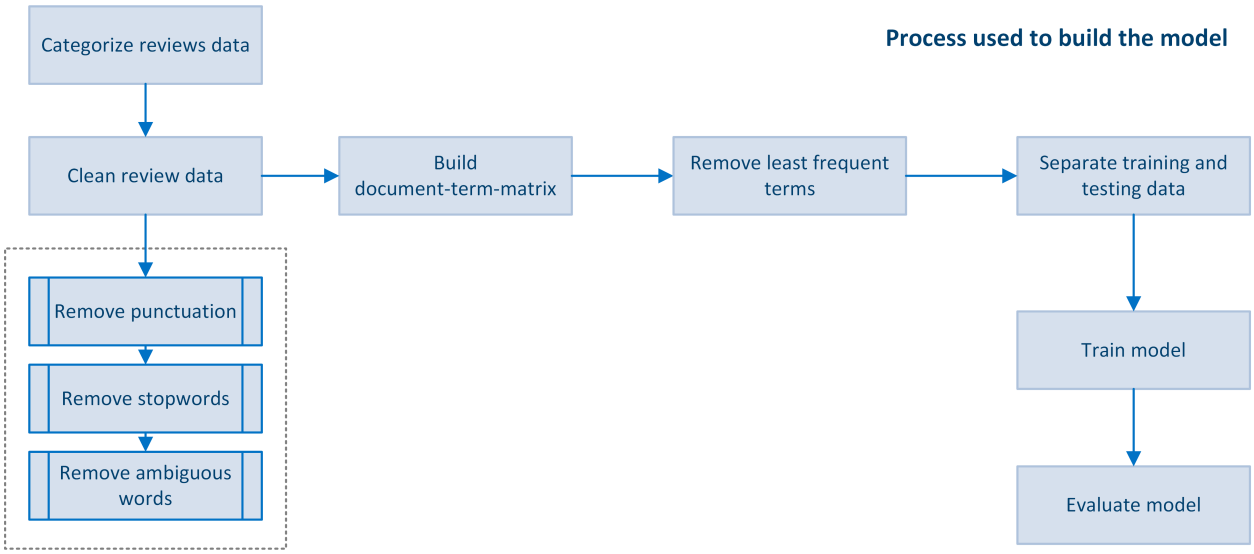
In order to build my classifier, it was necessary to identify the features that the model would use. Since I will be training the model only on review text, my features will be the words or sequence of words of these reviews. I decided to explore different models with different combinations. I would first try a model in which each particular word would be a feature. I also used n-gram combinations. An n-gram is a contiguous sequence

of n items from a given sequence of text or speech. Different models with unigrams (a n-gram of size 1), bigrams and trigrams, which are n-grams of size 2 and 3 respectively, were used.

See the following illustration for an example of the different n-grams for the phrase **"Food is good and staff is friendly"**.


I proceeded to create a corpus which is a collection of documents. In my case, a document is a restaurant review.

I cleaned the data by removing numbers, punctuation characters, stopwords (extremely common words that have little value in helping with classification) and separators from the corpus. I also removed some terms that I identified would confuse the model, words like *beer, chicken, burger, cookie, drink, café, dessert* would not be relevant for classification so I eliminated them from the corpus.




From this corpus, I built a document term matrix. A document term matrix is a matrix that contains a row for each restaurant review on my dataset and a column for each word in the corpus. Each cell in this matrix stores the number that represents the number of times the word appears in the document represented by that row. This particular type of matrix is called a sparse matrix since the majority of the cells are filled with zeroes since most words appear only in a few documents.

The following is a sample document term matrix from this dataset.

| | us | will | also | can | nice | try | well | got | love |
|---|---|---|---|---|---|---|---|---|---|
| text1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| text2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| text3 | 2 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| text4 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| text5 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 |

Because it was a big dataset, I only kept the words that appeared on at least 100 documents at least 100 times in order to reduce the number of features. Since the Naive Bayes algorithm works on categorical data, one last transformation to the dataset consisted on generalizing the value of each cell in the matrix, if it had a value of zero, which meant that the word wasn't present in that document. I then replaced the value with "No", if it had a number greater than zero, I replaced the value with "Yes".

Using this reduced document term matrix as my features, I proceeded to split the dataset into training and testing sets. I used 80% of the data as my training set, and 20% as my testing set. After training my model, I proceeded to evaluate the model with both my training and test sets in order to get the in sample and out of sample accuracy.

## Results

I ran the process described in the previous section twice. First with a dataset of 4,954 reviews and then a second time with 49,532 reviews[1]. Below are some metrics for the models trained with both of these datasets.

Table 5: Out of Sample Model Performance - 4,954 Rows

| NGrams | Accuracy | Sensitivity | Specificity |
|---|---|---|---|
| 1 | 0.8678103 | 0.9225968 | 0.6368421 |
| 2 | 0.8506559 | 0.9200999 | 0.5578947 |
| 3 | 0.8183653 | 0.9013733 | 0.4684211 |

Table 6: Out of Sample Model Performance - 49,532 Rows

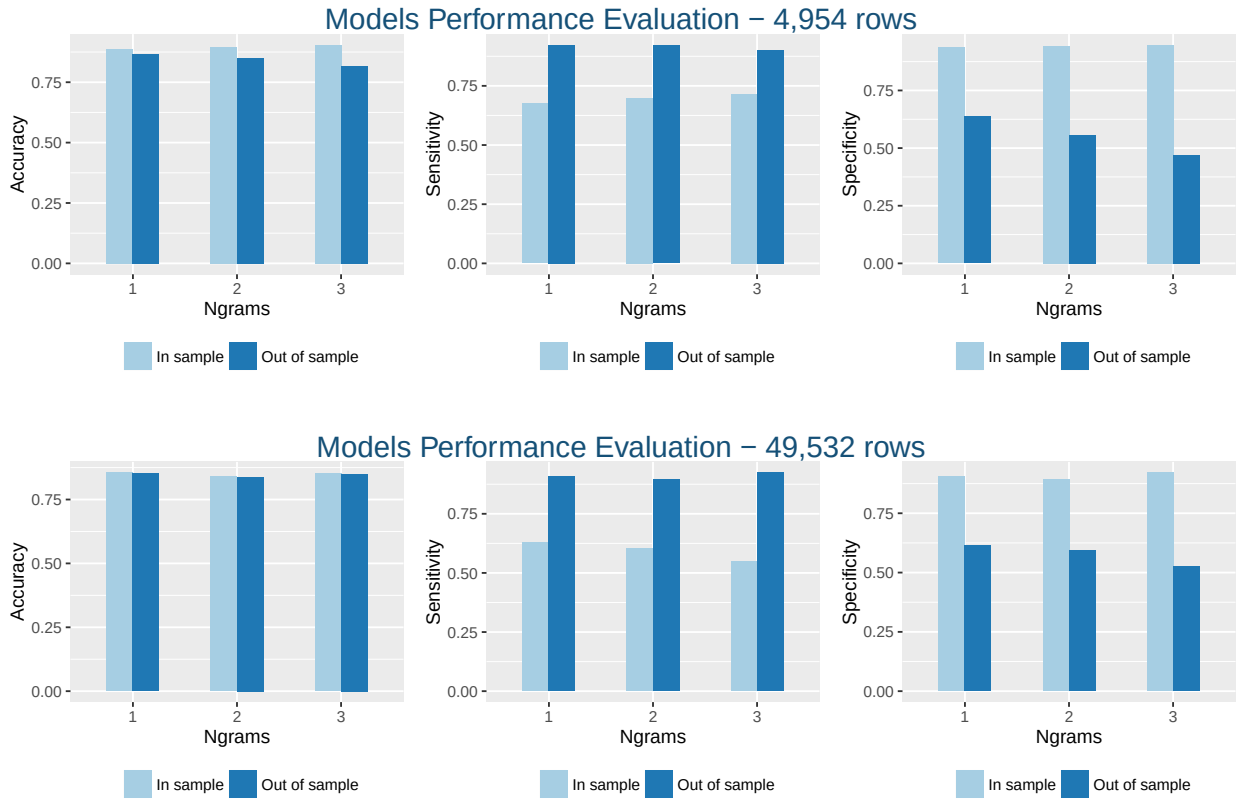
| NGrams | Reviews | Accuracy | Sensitivity | Specificity |
|---|---|---|---|---|
| 1 | 49532 | 0.8525285 | 0.9069363 | 0.6152597 |
| 2 | 49532 | 0.8378924 | 0.8940315 | 0.5930736 |
| 3 | 49532 | 0.8492985 | 0.9233155 | 0.5265152 |

It was interesting to see that the performance variation between the models trained with the two sets wasn't significative. I was also surprised to see that with the higher ngram count the performance of the model decreased.

I chose the unigram model built with the 4,954 rows as my final model since it had the best performance with an 87% accuracy. This accuracy confirms that it is possible to train a model using yelp reviews data to determine the sentiment of the review.

The following is a visual comparisson of the in sample and out of sample performance metrics for the models.

[1]Due to hardware limitations, I could only process 5% of the reviews data.




For this model, these were the most relevant features for both positive and negative reviews.


Model top 50 positive features


Model top 50 negative features

## Discussion

As the models show, it is possible to train a model to determine the sentiment of a restaurant review with a relatively high accuracy using the yelp reviews dataset as a training set. If a restaurant were to use this model to interpret the thousands of tweets, facebook posts, blog posts or articles posted online they could get a sense if people are saying negative or positive things about their stablishment online.