Christian Roos
May 9th, 2020
CSCI 6250: Big Data
Final Project Report

Project Overview

My project makes use of an open-source Yelp dataset that consists of restaurant information and reviews. Broadly, my aim was to develop two models that would use a restaurant review (with its words as tokens) to determine the restaurant's rating defined in terms of stars. Businesses should be concerned with such results. A study done by Michael Luca, an Assistant Professor at Harvard Business School, determined that Yelp reviews do impact revenue.[1] He concluded that every added star received on the website corresponded to a 5-9% increase in revenue, with locally-owned restaurants being the most affected and large chain restaurants being the least affected.

Data Set

The data set used in this project is:

Yelp Academic Dataset:

https://www.yelp.com/academic_dataset

The Yelp Academic Dataset is a subset of Yelp's complete data. It contains five JSON files: business, user, review, tip, and check-in. I focused on the review schema, primarily for text analysis. The schema for the review objects was:

```
'type': 'review':
        {
        'business_id': (the identifier of the reviewed business),
        'user_id': (the identifier of the authoring user),
        'stars': (star rating, integer 15),
        'text': (review text),
        'date': (date, formatted like '20110419'),
        'votes': {
                'useful': (count of useful votes),
                'funny': (count of funny votes),
                'cool': (count of cool votes)
                }
        }
```

The following information relates to the size, in bytes, of the data used in this project, the number of entries (or rows) in each dataset, and the number of attributes per row.

Dataset	File Size	Number of Entries	Number of Attributes
yelp_academic_dataset_review.json	1.4 GB	1569264	JSON(10)

Tools

[1] https://www.hbs.edu/faculty/Publication%20Files/12-016_a7e4a5a2-03f9-490d-b093-8f951238dba2.pdf

My toolset included:
- Java 8
- Hadoop 3.2.1

Technical Goals

In my project, I wanted to accomplish two technical goals:

1. Explore the complicated JSON structure of the Yelp Academic dataset.

2. Learn the Hadoop distributed file system and the MapReduce framework.
Employ MapReduce algorithms to construct machine learning models for large
textual datasets.

MapReduce

The project looks at text classification. The goal is to use the words in Yelp reviews to predict the number of stars int the star rating given by the users. I loaded the data set onto a Hadoop distributed file system in order to use MapReduce to perform text classification.

Two models: Naive Bayes Multinomial and Bernoulli.

Naive Bayes counts every word, while Bernoulli looks for presence of each word in a document.

Naive Bayes Multinomial:

Formulas:
$P(C_k \mid x_i) = \log[P(C_k)] + \sum \log[P(x_i \mid C_k)]$

$P(C_k)$ = number of reviews in class k / total number of reviews in all classes

$P(x_i \mid C_k)$ = number of occurrences of word i in class k / total number of words in class k

Map Reduce Steps:

1. ClassCounter

This class is primarily used to calculate $P(C_k)$. The function of the mapper is to associate each review with its corresponding star class. So, each review that rated a restaurant four stars with be affixed with the “four star” class label. The reducer then sums the total number of four star class reviews, e.g.

LabelMapper:
    Input: Text (from yelp_academic_dataset_review.json)
    Output: stars 1

LabelReducer:
    Input: stars [1, 1, 1, ...]
    Output: stars total

2. WordStarsCounter

In this class, the mapper produces tuples of words that appear in reviews with the corresponding classification (in stars) of those reviews. The reducer then sums up the total count of appearances of each word, given (again) its corresponding star classification.

WordStarsMapper:
    Input: Text (from yelp_academic_dataset_review.json)
    Output: word stars 1

WordStarsReducer:
    Input: word stars [1, 1, 1, 1, …]
    Output: word stars total

3. StarsWordProb

This class calculates relevant statistical data about word appearance, namely the frequency of a particular word (given its star classification) relative to the number of all other words given the same star classification. The mapper re-maps the results from the previous, making stars the key. The reducer analyzes the frequency.

StarsWordMapper:
    Input:Text (from WordStarsCounter)
    Output: stars word:total

StarsWordReducer:
    Input: stars [word:total, word:total, word:total, ...]
    Output: stars [word:total:prob, word:total:prob, ...]

4. ClassifyBayesian

This class calculates the final results: the prediction that, given some review, we can predict based on the words contained therein what star class that review should be associated with. We can compare the predicted value to the actual value.

ClassifyMapper:
    Input:Text (from yelp_academic_dataset_review.json)
    Output: review_id actual predicted

Commands:

bin/hadoop jar Hadoop-1.0.jar yelp.ClassCounter
bin/hadoop jar Hadoop-1.0.jar yelp.bayesian.WordStarsCounter
bin/hadoop jar Hadoop-1.0.jar yelp.bayesian.StarsWordProb
bin/hadoop jar Hadoop-1.0.jar yelp.bayesian.ClassifyBayesian
bin/hadoop jar Hadoop-1.0.jar yelp.CalculateError

Results:

	predicted				
actual	1	2	3	4	5
1	4	49	0	3	13
2	109	1	0	0	1
3	236	16	2	2	3
4	289	20	3	10	9
5	210	14	1	2	3

bayesian_accuracy
[1] 0.02

There is likely a bug somewhere in the above classifier causing it to think most of the reviews are 1-star. Further investigation is required.

Multivariate Bernoulli Naive Bayes:

Formula:

$P(C_k \mid x) = \log[P(C_k)] + \sum \log[(1 - P(x_i \mid C_k)^{(1 - x)}]$

$P(C_k)$ = number of reviews in class k / total number of reviews in all classes

$P(x_i \mid C_k)$ = reviews in class k that contain the word i / total number of review in class k

MapReduce Steps:

1. ClassCounter

LabelMapper:
Input: Text (from yelp_academic_dataset_review.json)
Output: stars 1

LabelReducer:
Input: stars [1, 1, 1, ...]
Output: stars total

2. StarsDocsCounter

StarsDocsMapper:
Input: Text (from yelp_academic_dataset_review.json)
Output: stars word

StarsDocsReducer:
Input: stars [word, word, word, word]
Output: stars [word:count, word:count, word:count,...]

3. ClassifyBernoulli

ClassifyMapper:
Input: Text (from yelp_academic_dataset_review.json)

Output: review_id actual predicted

Commands:

bin/hadoop jar Hadoop-1.0.jar yelp.ClassCounter
bin/hadoop jar Hadoop-1.0.jar yelp.bernoulli.ClassCounter
bin/hadoop jar Hadoop1.0.jar yelp.bernoulli.StarsDocsCounter
bin/hadoop jar Hadoop1.0.jar yelp.bernoulli.ClassifyBernoulli
bin/hadoop jar Hadoop-1.0.jar yelp.CalculateError

Results:

Bernoulli Confusion Matrix (sample size: 1000)

		predicted				
actual		1	2	3	4	5
	1	10	1	11	47	0
	2	0	31	13	67	0
	3	0	0	145	114	0
	4	0	9	0	322	0
	5	1	0	0	117	112

bernoulli_accuracy
[1] 0.62 = 62%

Future Work

Although the Bernoulli method significantly improved the accuracy of text-based classification, further improvements can still be made. For instance, many words appear across all of the documents, but have little effect on the classification of the review. These are called "noisy" terms. I attempted to remove a reasonable set of common words such as "a", "an", "the" by filtering each line of text through a stop-words list. However, a good deal of care must be taken regarding which terms are on the stop-words list. Normative words such as “good”, “bad”, “better”, “worse”, and so on should probably not be ignored because they form the meat-and-bones of the review, even though they are commonly used words. Things like articles, prepositions, and demonstrative pronouns should probably be removed though.