Yelp Reviews Topic Extraction: Creating Category-based Scores For Restaurants

Terry Shih
Undergraduate
Computer Engineering Student
Santa Clara University
Email: tshih@scu.edu

InHo Rha
Undergraduate
Computer Science Student
Santa Clara University
Email: irha@scu.edu

ABSTRACT

Yelp reviews are a great place for a restaurant owner to gain valuable feedback with regards to this restaurant. In this paper, we detail our solution to parsing through reviews and finding the most insightful advice for business owners using Latent Dirichlet Allocation. We present our findings from using LDA as well as difficulties faced running other models.

1 INTRODUCTION

Yelp is one of the most popular platforms on which people leave reviews for restaurants and other businesses. On Yelp, people can leave a review along with a rating between 1-5 stars for a given restaurant. These reviews, which typically contain from a couple sentences to several paragraphs, detail a customer's experience at an establishment and provide valuable insights into the reasonings behind their star ratings. The average of these star ratings have been shown to have direct correlations to the success of a business, with "a one-star increase in Yelp rating lead[ing] to a 5-9 percent increase in revenue" (Michael, 2016).

However, the average rating can sometimes be misleading. Without going through each individual review, a restaurant owner might struggle to find a solution to their low Yelp score. To remedy this issue, we sought to develop a program that goes through reviews for a restaurant and generates a rating for different performance categories. These categories would include: Food Quality, Service Quality, and Pricing.

By providing separate ratings for each of these categories, an owner of the business might more easily tell which area of his or her restaurant needs improvement.

2 RELATED WORK

Our work is largely inspired by the work of Huang, Rogers, and Joo. They ran LDA on a subset of Yelp Reviews to see hidden topics in their set of reviews. It was their work in topic classification that inspired us to look into Word2Vec to do our own unsupervised learning algorithm. Unfortunately, as detailed in section 3.2.1, we ran into issues getting the clustering algorithms to work with our vectors so we had to drop the Word2Vec model, instead falling back on using LDA ourselves.

3 METHODOLOGY

3.1 Dataset

The data used in this project was provided by Yelp through their Dataset Challenge program (Yelp). In the dataset, which consists of JSON files containing review data and business informations, there were over 5.2 million reviews for over 174,000 businesses. For this project we chose to extract only the businesses with the "restaurant" tag, resulting in 54,000 businesses and 3.4 million reviews for these establishments. However, as will be explained later in this paper, we had to further reduce our dataset to 300,000 reviews due to the time constraint and our hardware limitations.

3.1.1 Data preparation

First step for our data preparation process was to extract only the "restaurant" reviews. This was done by using a Pandas Dataframe to load the "business.json" data and indexing only the businesses with the "restaurant" tag in their "categories" column. We then used the list of ids from these restaurants to find the corresponding reviews from the "reviews.json" dataset.

After the restaurant reviews were extracted, we had to prepare the text data for topic modelling. We created a custom function called `Preprocessing` to do the following steps. For each of the review text, we first removed all punctuations and stop words from the data. We then lemmatized the individual words and turned them into a list of tokens. The processed data was then saved back into the Dataframe before being saved into a JSON file.

3.2 Tools

We used Python2, with heavy use of Sklearn library, as our main tool for the project. Other important libraries we used includes Pandas, Numpy, NLTK, and gensim.

3.2.1 Attempt 1: Word2Vec

Our first attempt at topic modelling involved the use of Word2Vec algorithm. Without going into the specifics, Word2Vec is basically an algorithm for processing text and converting a word or a document it into a n-dimensional vector. It achieves this by using a shallow "two-layer neural net" that are "trained to reconstruct linguistic contexts of words" (DL4J). Simply put, it converts a text into a vector. A well-known example of this algorithm is the "king + (man - woman) = queen" demonstration (Figure 1). The theory is that the vector distance between the point "king" and "queen" is roughly equal to the vector between "man" and "woman" so subtracting the vector of "man-woman" from "king" results in a vector close to the "queen" vector.

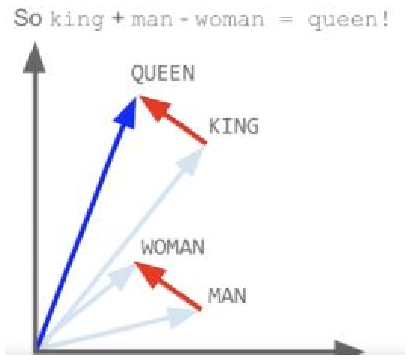


Figure 1

We theorized that after we use the Word2Vec algorithm to convert the reviews into n-dimensional vectors, we could run a clustering algorithm on the vectors to extract the latent topics from the data.

We were able to successfully create a Word2Vec model that converted our reviews into 50-dimensional vector using the *gensim* library but quickly faced trouble when we attempted to run clustering algorithms on the outputted 3.4 million vectors. We attempted to use various unsupervised clustering algorithms but they all failed because it:

1. Required more memory than was feasible (kMeans, MiniBatchKMeans)
2. Too long to compute (Mean-shift)
3. Skewed or meaningless clusters (DBSCAN)

After spending a lot of time and resources on this attempt, we realized that we were stuck and decided to abandon this method.

3.2.2 Attempt 2: LDA

We used Latent Dirichlet Allocation (LDA) to cluster the reviews based on hidden topics. LDA is

Bayesian generative model that returns the underlying topics found in a group, or corpus, of documents, or in the case of this project, reviews. The model assumes that a topic is a distribution of words pulled from the dictionary of possible words, V. Any word can appear in multiple topics.

In LDA, a document, D, is assumed to be a bag-of-words comprised of N words. D can have one or many topics out of K possible topics. To generate a document, you label each word in D using the weight of the topics of D. Then, you pick each word from the labelled topic using the distribution in the topic. The finished product is a document that has a jumbling of words that loosely relate to the topics of D. LDA reverse engineers this process. Given the corpus of documents and K, LDA determines the distribution of words in each of the K topics such that the corpus of documents could have been made using the above method with the generated topic distributions.

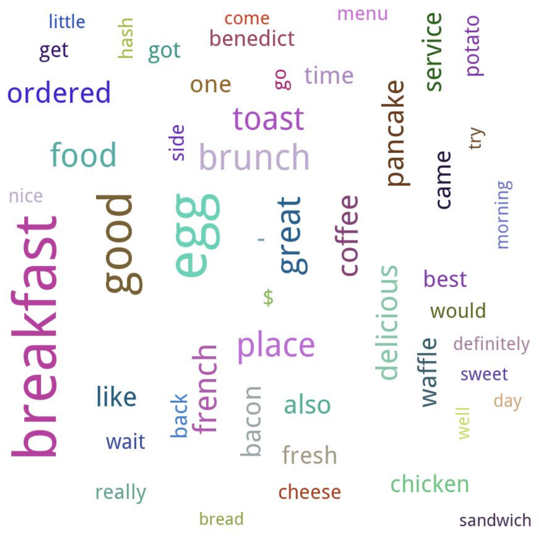
We used sklearn's LDA implementation to make an LDA model. With our 300,000 reviews, we tested for K = [10, 15, 20, 25, 30, 40] with |V| = 32244. In Table A1 (appendix table), we see the count of reviews whose top topic is the cluster given by the row number. Note that while we used only the most probable topic for this visualization of clustering, we still have information regarding the other hidden topics in a given review. Given our constraint on time, we decided to use K = 20 as it provided a decent spread of reviews among topics while still having some low count topics.

3.3 Labelling Clusters

Because LDA is an unsupervised learning algorithm, we had to go through the process of labelling each of the 20 topics returned by the LDA model. Since each topic contains a list of words and their respective probability within the topic, we could extract from each topic the 50 most likely to occur words. We brought the top 25 into a text file for our reference and turned the top 50 into a word cloud. We also looked at random samplings of lemmatized reviews without stop words in each cluster, and with the top 50 words, determined an appropriate label for the topic

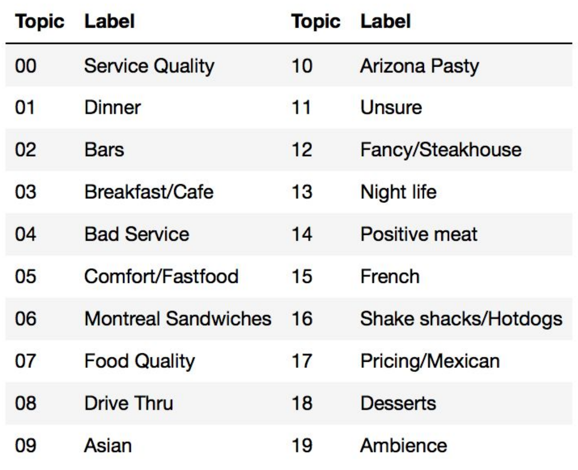
For instance, some of the top words of topic 3 include: breakfast, egg, brunch, coffee, french, pancake, toast, waffle. This can be visualized in the word cloud displayed in Figure 1. The obvious choice for the label then would be breakfast, but that alone is insufficient. Looking through some of the reviews in this cluster, we find that not only are there many reviews about breakfast and breakfast places, but there are also reviews of coffee shops in topic 3, which makes sense. People drink coffee for breakfast, but also at cafes, so it's probable that they would be clustered together. So, with these observations, we came to the label, 'Breakfast/Cafe', for topic 3.

Fig. 1: Topic 3 Word Cloud



4 RESULTS

4.1 Extracted Subtopics



Topic	Label	Topic	Label
00	Service Quality	10	Arizona Pasty
01	Dinner	11	Unsure
02	Bars	12	Fancy/Steakhouse
03	Breakfast/Cafe	13	Night life
04	Bad Service	14	Positive meat
05	Comfort/Fastfood	15	French
06	Montreal Sandwiches	16	Shake shacks/Hotdogs
07	Food Quality	17	Pricing/Mexican
08	Drive Thru	18	Desserts
09	Asian	19	Ambience

Fig. 2:Labels for Topics

We performed our labelling process on each of the 20 topics given by our LDA model and came up with these labels. Because some of the reviews in our data came from relatively few restaurants, some of the topics became those restaurants, as in the case with topics 6 and 10. While topics 0 and 4 both tended to talk about service, the reviews in topic 4 were largely negative, which is where the distinction comes from. Using the model, we can transform reviews and get an array of weights of each topic in a review. For instance, take this review:

> *Melanie is a treasure that must be protected at all costs. Seriously though, she was super-cool and funny and gave us some pretty fun and adventurous recommendations that we would otherwise not have tried. Really attentive, she even helped us cook our food - it felt like we were getting an experience that other restaurant-goers were not getting. Like v.i.p. status. That's how awesome she is. Food was really good. Excellent recommendations. Next time I come here, I'm just going to tell them to surprise me. I like the music here. Will definitely come back.*

Running this review through our LDA model gives us topic 0 (Service Quality) as the most weighted topic, followed by topic 7 (Food Quality) as the second most weighted topic. This makes sense as this particular review appears to give praise to a specific waitress and also talks at length about how great the food is.

4.2 Category-based Restaurant Scores

```
# Example Code
[IN]
getHiddenRatings(business_id='Gt4z3Ayl
NTsEPDkzkaC7HA', df=df)
[OUT]
Restaurant: Stacks & Yolks
Total # of reviews: 660
Overall Average: 3.25606060606
Service Average: 3.5406162465
Food Average: 2.46195652174
Pricing Average: 4.05555555556
```

In order to get the hidden ratings for Food, Service, and pricing from the reviews of restaurant, we first grab all the reviews for a restaurant and run the reviews through our LDA model to get the topics of each review. Since certain topics are related to these topics (topics 0 and 4 talk about service for example), the getHiddenRatings function sifts through all the restaurant's reviews and considers a review as talking about any of the three topics if the review has a high weight for that topic. In the example above, we see that although Stacks & Yolks could use a lot of work improving the quality of their food, we also notice that their pricing is scored rather high.

5 FUTURE WORK

Though we were able to parse out promising information we recognize that our model was not made in the best of conditions.

As LDA is an unsupervised learning algorithm, we would like to have been able to verify the accuracy of our model. By tagging new reviews with the probable categories then running them through the model, we could achieve this. But, in order to get an acceptable test set for accuracy checking, we would need more manpower and time to go through and manually tag a bunch of reviews.

Additionally, deeply analyzing the outputs of the other clustering sizes might give us more insight into the accuracy of our choice of K=20. One quirk of LDA that we did not take into consideration until fairly late in the process is that you do not need to use all of the topics given back by LDA. In fact, having a couple of "trash" topics might theoretically give us cleaner, more useful topics in the others. Much like getting a better read on accuracy though, we would need more time or manpower to run our manual labelling process for the different K values.

We also want to take another, closer look at Word2Vec. Though we ran into issues with Word2Vec and training vector clustering algorithms, we believe that there is potential in using Word2Vec to cluster reviews.

6 REFERENCES

James Huang, Stephanie Rogers, and Eunkwang Joo. 2013. *Improving Restaurants by Extracting Subtopics from Yelp Reviews*. Yelp, San Francisco, CA.

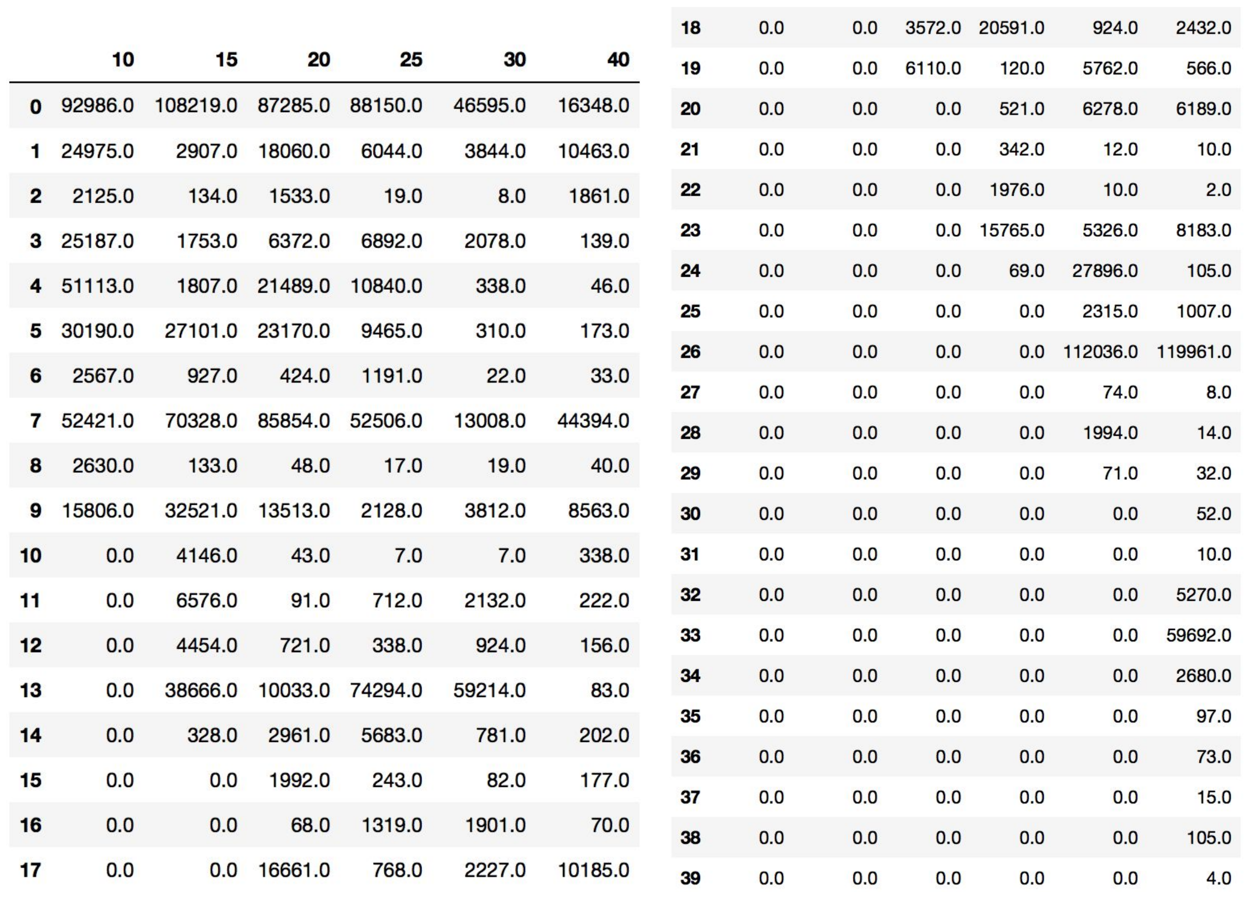
Yelp. *Yelp Dataset Challenge.* https://www.yelp.com/dataset/challenge

Michael Luca. 2016. *Reviews, Reputation, and Revenue: The Case of Yelp.com*. Harvard Business School, Cambridge, MA.

David M. Blei. 2009. *Topic Models Lecture*. Machine Learning Summer School, Cambridge, NJ.

DL4J. https://deeplearning4j.org/word2vec.html

APPENDIX



	10	15	20	25	30	40
0	92986.0	108219.0	87285.0	88150.0	46595.0	16348.0
1	24975.0	2907.0	18060.0	6044.0	3844.0	10463.0
2	2125.0	134.0	1533.0	19.0	8.0	1861.0
3	25187.0	1753.0	6372.0	6892.0	2078.0	139.0
4	51113.0	1807.0	21489.0	10840.0	338.0	46.0
5	30190.0	27101.0	23170.0	9465.0	310.0	173.0
6	2567.0	927.0	424.0	1191.0	22.0	33.0
7	52421.0	70328.0	85854.0	52506.0	13008.0	44394.0
8	2630.0	133.0	48.0	17.0	19.0	40.0
9	15806.0	32521.0	13513.0	2128.0	3812.0	8563.0
10	0.0	4146.0	43.0	7.0	7.0	338.0
11	0.0	6576.0	91.0	712.0	2132.0	222.0
12	0.0	4454.0	721.0	338.0	924.0	156.0
13	0.0	38666.0	10033.0	74294.0	59214.0	83.0
14	0.0	328.0	2961.0	5683.0	781.0	202.0
15	0.0	0.0	1992.0	243.0	82.0	177.0
16	0.0	0.0	68.0	1319.0	1901.0	70.0
17	0.0	0.0	16661.0	768.0	2227.0	10185.0
18	0.0	0.0	3572.0	20591.0	924.0	2432.0
19	0.0	0.0	6110.0	120.0	5762.0	566.0
20	0.0	0.0	0.0	521.0	6278.0	6189.0
21	0.0	0.0	0.0	342.0	12.0	10.0
22	0.0	0.0	0.0	1976.0	10.0	2.0
23	0.0	0.0	0.0	15765.0	5326.0	8183.0
24	0.0	0.0	0.0	69.0	27896.0	105.0
25	0.0	0.0	0.0	0.0	2315.0	1007.0
26	0.0	0.0	0.0	0.0	112036.0	119961.0
27	0.0	0.0	0.0	0.0	74.0	8.0
28	0.0	0.0	0.0	0.0	1994.0	14.0
29	0.0	0.0	0.0	0.0	71.0	32.0
30	0.0	0.0	0.0	0.0	0.0	52.0
31	0.0	0.0	0.0	0.0	0.0	10.0
32	0.0	0.0	0.0	0.0	0.0	5270.0
33	0.0	0.0	0.0	0.0	0.0	59692.0
34	0.0	0.0	0.0	0.0	0.0	2680.0
35	0.0	0.0	0.0	0.0	0.0	97.0
36	0.0	0.0	0.0	0.0	0.0	73.0
37	0.0	0.0	0.0	0.0	0.0	15.0
38	0.0	0.0	0.0	0.0	0.0	105.0
39	0.0	0.0	0.0	0.0	0.0	4.0

Table A1: [columns] The number of each cluster using LDA with K= column label. [rows] The cluster number. [values] The number of reviews in that cluster.

Steps to run our code:

Using: Python 2 and Jupyter Notebook
Required packages: numpy, pandas, sklearn, matplotlib, gensim, nltk, pytagcloud (optional tool for creating word clouds from https://github.com/atizo/PyTagCloud)

Setup:

1. Clone https://github.com/edwardrha/YelpReviewPrediction

2. Download Yelp data from https://www.yelp.com/dataset/challenge and place the JSON files into the /dataset directory.

3. Run the preprocessing.py from inside the /src directory. This will create and save the processed restaurant review files into the /dataset directory.

WARNING: This step requires very high amount of RAM(64GB or higher). We do not recommend this step to be run and instead use the processed JSON uploaded to Google Drive here:
https://drive.google.com/drive/folders/1gYgWxNDK_78notWqKFuRiujawdkB640r?usp=sharing

4. Run the ClusterModel.py from inside the /src directory. This will create a CountVectorizer object, feature names object, train 6 LDA models for k=[10, 15, 20, 25, 30, 40] and pickle them into /models directory. It will also save the label predictions for the reviews into /dataset directory as txt files.
NOTE: Training each LDA model takes around 30 minutes each per core. Time consuming.

5. Now the Main.ipynb in the main directory is ready to be run.

Main.ipynb:
By using the models and data created from the Setup process, demonstrates how we can predict the category of a review and use it to give the categorical rating for a chosen restaurant.

Labeling.ipynb:
Contains the codes we used to examine the reviews from a cluster so we can manually label them.

Slides.ipynb:
Contains the codes and slides used to create our presentation. Run in terminal:
```
jupyter nbconvert Slides.ipynb --to slides --post serve
```
To open the presentation slides.