Exploration of Data Set

Snapshot of single message (data per line)


```
{
    "votes":{
        "funny":0,
        "useful":2,
        "cool":1
    },
    "user_id":"Xqd0DzHaiyRqVH3WRG7hzg",
    "review_id":"15SdjuK7DmYqUAj6rjGowg",
    "stars":5,
    "date":"2007-05-17",
    "text":"dr. goldberg offers everything i look for in a general practitioner.  he's nice and easy to talk to without
    being patronizing; he's always on time in seeing his patients; he's affiliated with a top-notch hospital (nyu) which my
    parents have explained to me is very important in case something happens and you need surgery; and you can get
    referrals to see specialists without having to see him first.  really, what more do you need?  i'm sitting here trying
    to think of any complaints i have about him, but i'm really drawing a blank.",
    "type":"review",
    "business_id":"vcNAWiLM4dR7D2nwwJ7nCA"
}
```


Details

Data Set: '*yelp_academic_dataset_review.json'*
Number of review data: Approximately **1.12 mil**
GitHub (for reference on implementation): https://github.com/atif-github-venture/data-mining-capstone (**Note**: The data and large files are excluded, they can be generated by execution)
Make sure to add 'data' and 'output' directory when cloning. Also add the unzipped data provided as is in the data folder as inputs.

Observation on the data

Even though the review data is provided as json, it is as a single json message per line. There is a need to convert the data in flat structure of tabular column. I have written a 'not so generic approach' which is faster but not reusable without making little changes in 'exploration_review_data.py', not bad but not better either. In my approach of conversion, assumed every message (data per line) contains same key value pair.

In my research, I came across an already developed solution which was out dated. So as per python 3 standards, I fixed the code and works perfectly fine in 'json_to_csv_converter.py'.

Run as "*python3 json_to_csv_converter.py ../data/yelp_academic_dataset_review.json*"

Approach taken

Get the superset of the column names in case there is a message with missing column name.

Flatten the nested dictionary set to be known as column name "key1.ke2". After that its simple, just saving the content in the CSV file well with a minute (conversion time).
There is also a recursive method to fetch the value from nested dict.

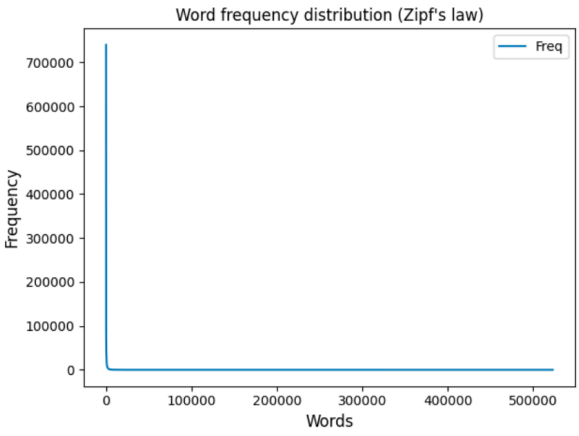
```
Example:
    d = {
        'a': {
            'b': 2,
            'c': 3,
            },
    }
    key = 'a.b'
    will return: 2
```

Generating Collection corpus for review text

Collected and squished all the review text from data. Refer to 'exploration_review_data.py'. Tokenized the string (tag suppression, lower case, stop words filter) and computed the frequency distribution and saved output in a text file. See details below for different volume of data considered, note that with all the data, there is lesser curvature compared to 50,000 review data.

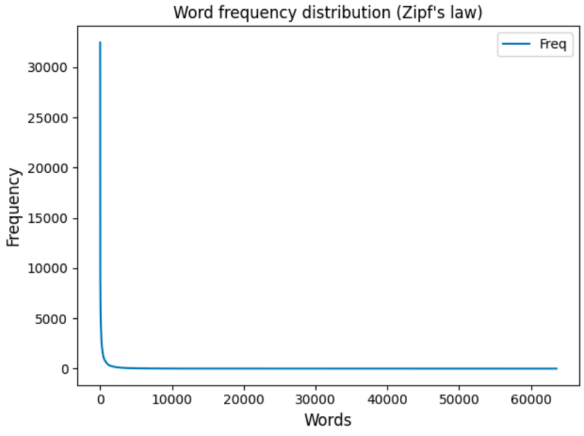
Execution time 193 s with all data.

	Word	Freq
0	place	739891
1	good	696778
2	food	638094
3	time	520483
4	great	509059



Execution time 193 s with 50,000 reviews.

	Word	Freq
0	place	32468
1	good	28450
2	food	27156
3	time	22594
4	great	21701



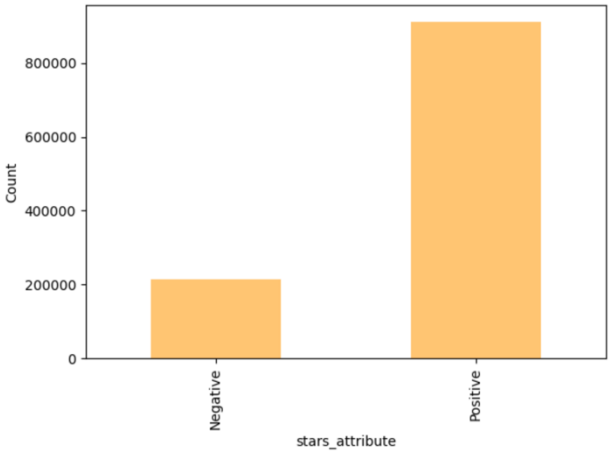
Stars Rating and its association with 'Positive' or 'Negative'

- Read the dataset using pandas library
- Drop the row which does not have comment.
- Describe the data by grouping per rating
- Classify the positive/negative based on ratings

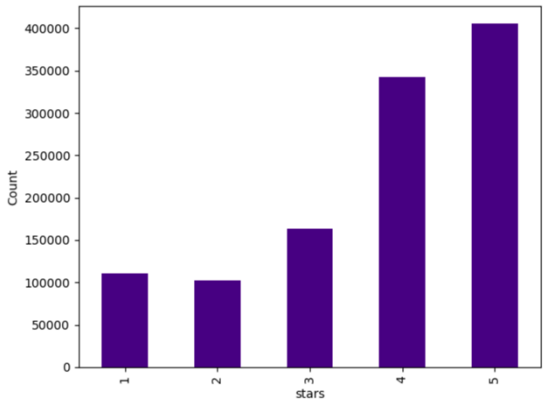
- For Analysis: -> Positive as 3, 4, 5; Negative as 1, 2

Shape of the data

```
Shape of the data set: (1125457, 10)
       votes.useful         stars    votes.cool   votes.funny
count  1.125457e+06  1.125457e+06  1.125457e+06  1.125457e+06
mean   1.132279e+00  3.737437e+00  6.533710e-01  5.250809e-01
std    2.125122e+00  1.299345e+00  1.712005e+00  1.633619e+00
min    0.000000e+00  1.000000e+00  0.000000e+00  0.000000e+00
25%    0.000000e+00  3.000000e+00  0.000000e+00  0.000000e+00
50%    0.000000e+00  4.000000e+00  0.000000e+00  0.000000e+00
75%    1.000000e+00  5.000000e+00  1.000000e+00  0.000000e+00
max    1.660000e+02  5.000000e+00  1.370000e+02  1.410000e+02
```




```
stars
1     110771
2     102737
3     163761
4     342143
5     406045
```



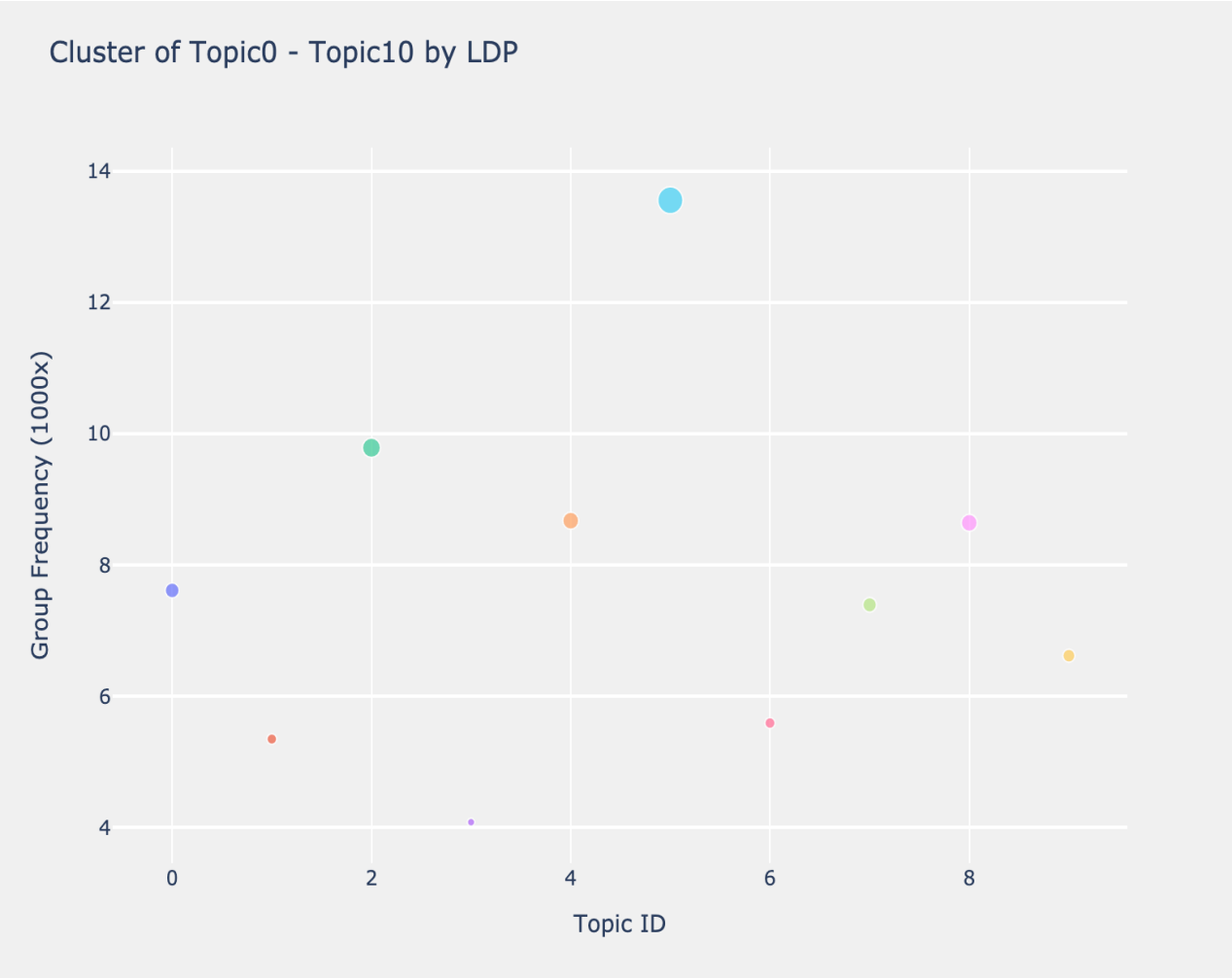
Task 1.1

Use a topic model (e.g., PLSA or LDA) to extract topics from all the review text (or a large sample of them) and visualize the topics to understand what people have talked about in these reviews.

- Execution time to generate the 10 topics, with 50,000-word features, 15 words per topic: 528s, with all the data set

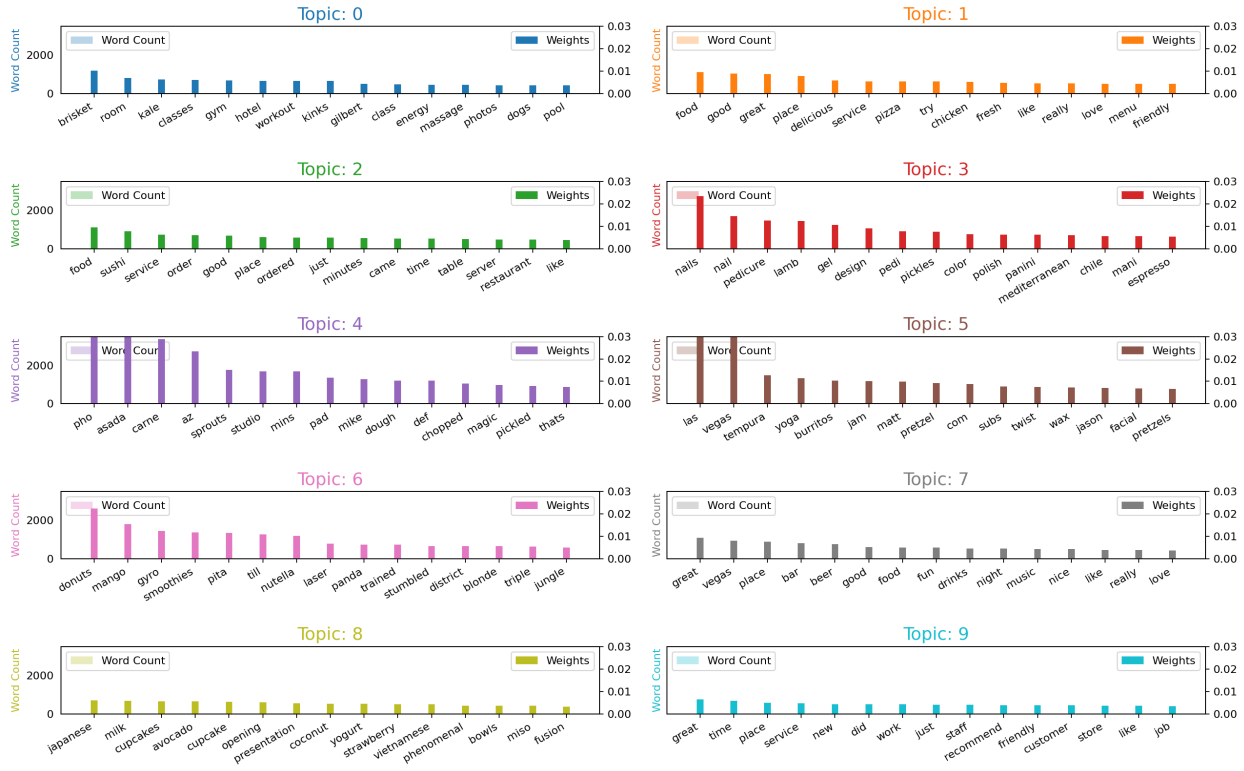
- Python class: 'topic_model.py' (*use the main method to alter the parameters*)
- Use *'requirements.txt'* to install the packages needed for this project.

Below diagram shows the clustering weightage per topic based on the diameter of each topic.



A word cloud with the size of the words proportional to the weight is shown below.

The importance (weights) of the keywords matters. Along with that, how frequently the words have appeared in the documents is also interesting to look at. Below graph depict the word count Vs weight plot for different topics. This is another way to emphasize the weightage of words within a topic.

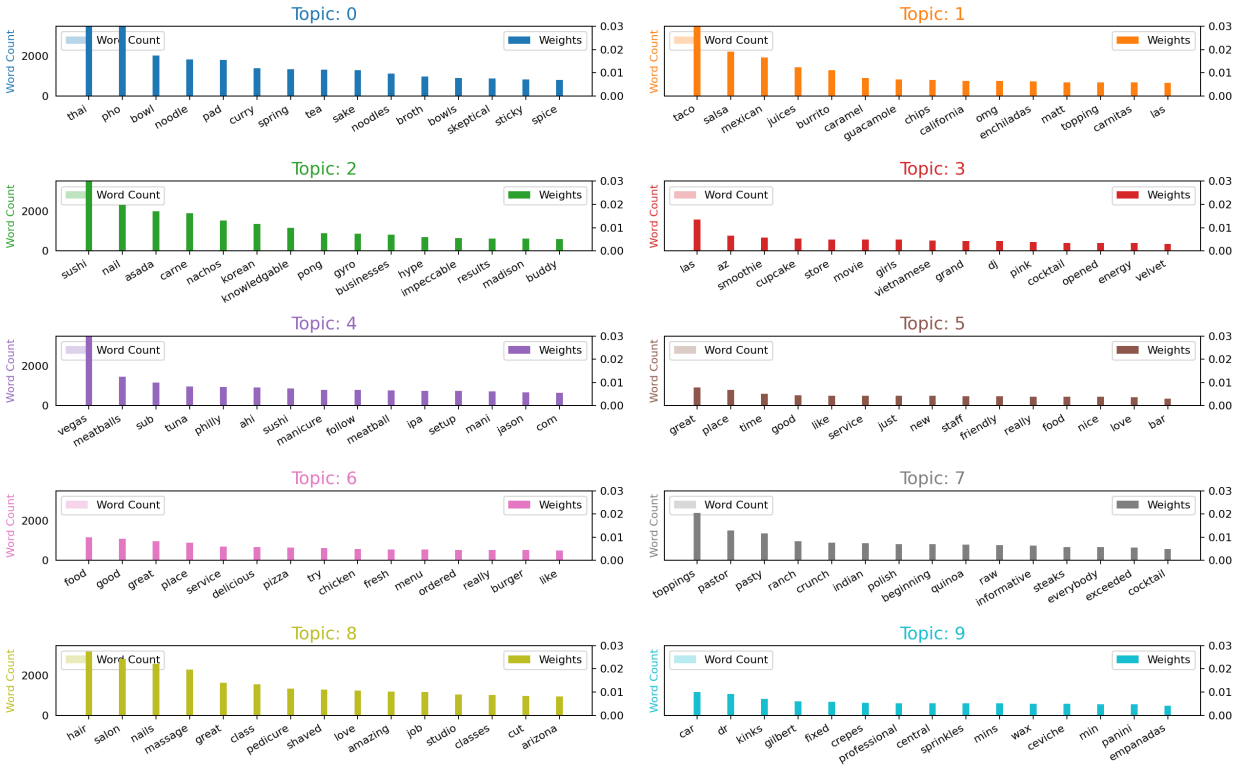


Task 1.2

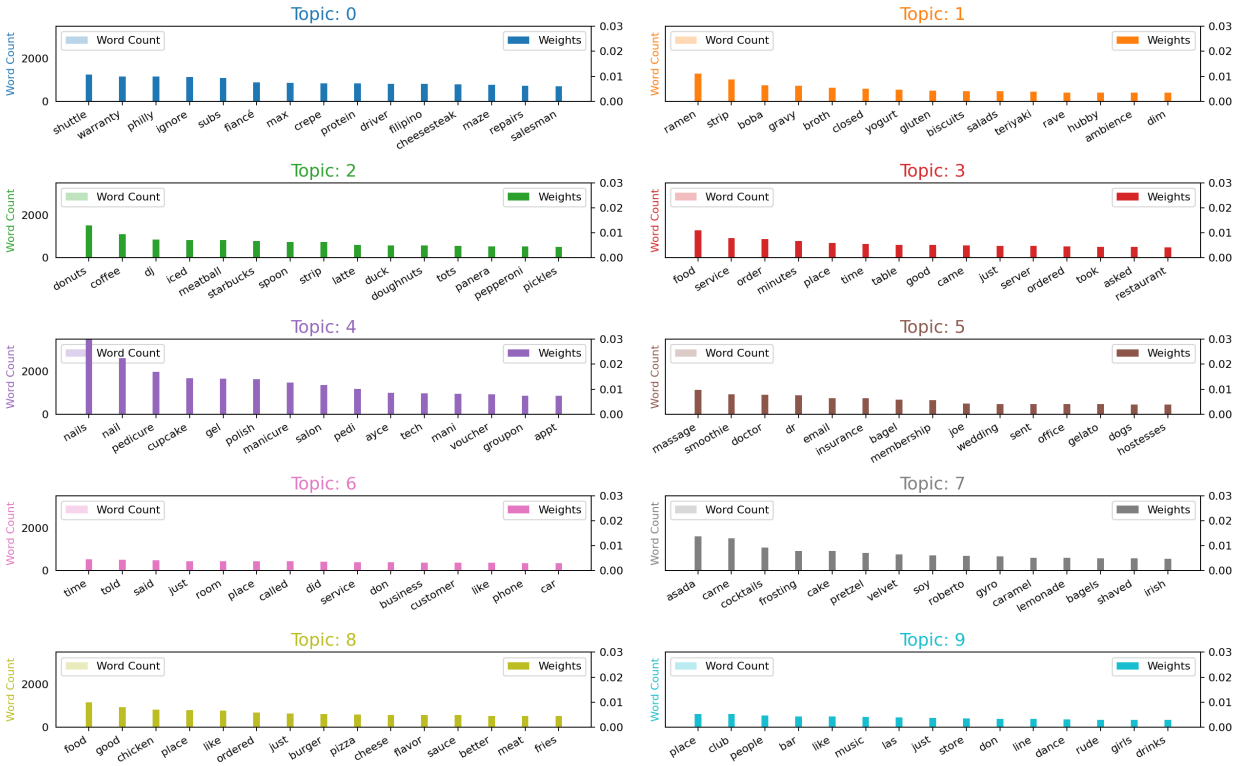
Do the same for two subsets of reviews that are interesting to compare (e.g., positive vs. negative reviews for a particular cuisine or restaurant), and visually compare the topics extracted from the two subsets to help understand the similarity and differences between these topics extracted from the two subsets. You can form these two subsets in any way that you think is interesting. Here we show a sample visualization for a sample of reviews with high and low ratings.

- Python class: 'topic_model.py' (*use the main method to alter the parameters*)
- Please uncomment by following direction in the python class. The approach for producing the data based on the positive Vs Negative is by applying all the same code and just by filtering the review comments i.e., the data set based on the star rating (as explained above). This can be achieved by concept of data-frame in pandas.
- 911,949 data set for 'positive' reviews, execution time – 429 s (rest same parameters)
- 213,508 data set for 'negative' reviews, execution time – 218 s (rest same parameters)

For positive reviews based on stars rating 3, 4 & 5


Topic: 0
Word Count
Weights
thai pho bowl noodle pad curry spring tea sake noodles broth bowls skeptical sticky spice
Topic: 1
taco salsa mexican juices burrito caramel guacamole chips california omg enchiladas matt topping carnitas las
Topic: 2
sushi nail asada carne nachos korean knowledgable pong gyro businesses hype impeccable results madison buddy
Topic: 3
las az smoothie cupcake store movie girls vietnamese grand dj pink cocktail opened energy velvet
Topic: 4
vegas meatballs sub tuna philly ahi sushi manicure follow meatball ipa setup mani jason com
Topic: 5
great place time good like service just new staff friendly really food nice love bar
Topic: 6
food good great place service delicious pizza try chicken fresh menu ordered really burger like
Topic: 7
toppings pastor pasty ranch crunch indian polish beginning quinoa raw informative steaks everybody exceeded cocktail
Topic: 8
hair salon nails massage great class pedicure shaved love amazing job studio classes cut arizona
Topic: 9
car dr kinks gilbert fixed crepes professional central sprinkles mins wax ceviche min panini empanadas

For Negative data set based on stars rating 1 and 2



Inspiration

- https://www.machinelearningplus.com/nlp/topic-modeling-visualization-how-to-present-results-lda-models/
- https://github.com/yfliu87/DataMining_Capstone/blob/master/task1/task1.1/task1.1_generate_graph.py
- **LDA**: code refactored from provided toolkit by the course.

- https://github.com/Yelp/dataset-examples/blob/master/json_to_csv_converter.py

Packages Used

- nltk
- metapy
- pandas
- matplotlib
- gensim
- wordcloud
- sklearn
- plotly