Improving Yelp Ratings

Xiaocheng Hou, Tony Forlini and Mavjuda Nihmat

Illinois Institute of Technology

Abstract:
Improving yelp ratings can help customers find the most attractive aspects out of different types of restaurants. In this project, we used the latent semantic analysis (LSA) and latent dirichlet allocation (LDA) to find the "hidden concepts" from the customer reviews. In this report, only LDA will be discussed. Based on the review stars, we re-rate the business using these latent concepts. Then we analyze the performance of the LDA algorithm.

Introduction:
In a Yelp search, a star rating is arguably the first influence on a user's judgment. Located directly beneath business' names, the 5-star meter is a critical determinant of whether the user will click to find out more, or scroll on. In fact, economic research has shown that star ratings are so central to the Yelp experience that an extra half-star allows restaurants to sell out 19% more frequently.[0] However, what if a customer cares about only good food and service? In this case, the customer has to read tons of reviews to figure out this question on Yelp.

Thus, in this project, the new feature we intend to provide is to personalize the Yelp experience so that each user receives a new ranking that account for his or her preferences. The general approach is to find some key words from the reviews and label them as different aspects so that the customer can choose the restaurant they want as soon as possible. The most important thing is to find the key words and some appropriate labels.

Related work:
Hitesh Sajnani[1] who comes from UC Irvine manually inspected a few hundred reviews for restaurant businesses and found 5 important dimensions and use binary relevance to help them to improve ranking. But they just did some small datasets. James Huang et. al [2] use the LDA to point out demand of customers from a large amount of reviews, with high dimensionality and increase Yelp ratings based on the meaningful insights provided by these topics. But they still lack accuracy.
Julian McAuley [3] aimed to fuse latent rating dimensions with latent review topics. They just provided a good method for us to find the latent concepts.

Based on these pervious works, we want to use LDA/LSA to find the concepts the customer most cared about. With the improved ranking of the restaurant, people can pick up them directly as well as convenience.

Packages:
The R packages we used are the NLP, slam, car, cluster, rjson, tm, topicmodels, lda.

Data preprocessing:

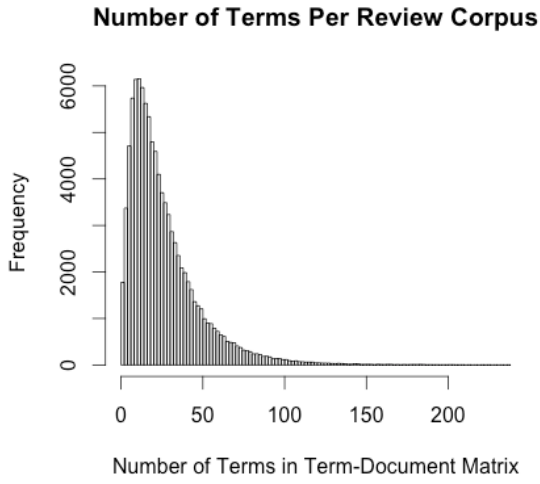
Yelp provide the academic datasets, it can be downloaded from http://www.yelp.com/dataset_challenge. The total size of the datasets is 1.77GB. There are some information about the business, checkin, review, tip, user and some instructions. We will use the review and business datasets to analyze and reach our goal. The files are json files.

The main data we used is review, which is 1.47GB with 1569264 entries. After we extracted restaurant reviews, the datasets size decreased to 890MB with 990627 entries. We code the python script to filter the datasets. The datasets are still too large to load to R so that we have to split them to smaller datasets. Each smaller datasets have 200000 reviews. After load the library we need in R, the review data we chose was read by the UnpackJSON function.
The datasets loaded to R showed below:

votes	user_id	review_id	text
list(funny = 0, useful = 5, cool = 0)	LWbYpcangjBMm4KPxZGOKg	6w6gMZ3iBLGcUM4RBIuifQ	This place was DELICIOUS!! My parents saw a recommendation to visit this p
list(funny = 0, useful = 0, cool = 0)	m1FpV3EAeggaAdfPx0hBRQ	jVVv_DA5mCDB6mediuwHAw	Can't miss stop for the best Fish Sandwich in Pittsburgh.
list(funny = 0, useful = 2, cool = 1)	8fApIAMHn2MZJFUiCQto5Q	3Es8GsjkssusYgeU6_ZVpQ	This place should have a lot more reviews - but I'm glad it doesn't, they d Its been there ages, and looks it. If you're all about ambiance, don't both The food is AWESOME! Worth any little complaints I might think up before it -Fish Sandwiich -Salmon (huge and delicious) -Flounder -Shrimp a few ways ("Norfolk" style is oily for my taste, and I never had i -Hawkins St Special -Prime Rib (sized for two, watch it) The prices are low, the portions are large, and just about everything on th
list(funny = 0, useful = 1, cool = 0)	uK8tzraOp4M5u3uYrqIBXg	KAkcn7oQP1xX8KsZ-XmktA	This place was very good. I found out about Emil's when watching a show cal
list(funny = 0, useful = 0, cool = 0)	6wvlM5L4_EroGXbnb_92xQ	BZNJkkP0bXnwQ2-sCqat2Q	Old school.....traditional "mom 'n pop" quality and perfection. The best fi
list(funny = 0, useful = 2, cool = 1)	345nDw0oC-jOcglqxmzweQ	VDTIbR3G5_IPkpXbo2MutA	Seen this restaurant on 25 best places in Pittsburgh with Rick Seback ack. Went there with my girlfriend she grew up with the owner. She's very n
list(funny = 0, useful = 0, cool = 0)	u9ULAsnYTdYH65Haj5LMSw	5uyYmniVyIB_wtKtyXDudQ	Wonderful reuben. Map shown on Yelp page is incorrect. It is actually a di
list(funny = 0, useful = 0, cool = 0)	pdHC0oAcG7gNdhuFRAUu0Q	zyn_Libz9VZTZ--OdC4-tQ	Good fish sandwich.
list(funny = 0, useful = 0, cool = 0)	tAKjY3bQXH51msJbOHYPmQ	uf61rPucuICXhSPXlZ1hIQ	After a morning of Thrift Store hunting, a friend and I were thinking of lu Well, seeing as how I'm kind of addicted to late 40's and early 50's, and t And yet another shot at finding a decent Reuben in da burgh...well, that's So off we go right at lunchtime in the middle of...where exactly were we? A We walked in the front door, and entered another world. Another time, and a This is about as old world bar/lounge/restaurant as it gets. Plain, with a Oh...but good food counts too. We each had a Reuben, and my friend had a side of fries. The Reubens were d My friend also mentioned that they have a Chicken Parm special one day of t The waitress did a good job, especially since there was quite a growing cro They only have Pepsi products, so I had a brewed iced tea, which was very f Emil's is no frills, good portions, very reasonable prices, VERY comfortabl You definitely want to hit Mapquest or plug in your GPS though. I am not su Addendum: 2nd visit for the fish sandwich. Excellent. Truly. A pound of fis

The most important step is to do the text mining for the review data. Typically, we load the tm package and NLP package to deal with the text vector showed above. We did the regular way to remove the stop words, strip white space, transfer all letters to low case and remove numbers, punctuation.

After these steps, it was the time to construct the Term Document Matrix. It would help us to get a terms matrix with weight frequency.



Some frequent words appear more than 50 times. The word cloud was created after using weight tf-idf.

We may not only remove some stop words, but also remove the words like good and bad. The goal of us is to find some aspects, but not the positive or negative words. From the most frequents words, we can find that the customer care about the atmosphere, service dessert, food quality, offer and so on.

To specify the concepts, we need to apply some algorithm to find the latent topics.

Topics model:
Then we tested the latent meaning from these reviews and found the most representatives words. After we got the far-reaching concepts that concerned by customers.

1. LSA (Latent Semantic Analysis):
The first algorithm we thought is LSA (Latent semantic analysis). It is a retrieval method that uses a mathematical technique called singular value decomposition (SVD) to identify patterns in the relationships between the terms and concepts contained in an unstructured collection of text. A key feature of LSA is its ability to extract the conceptual content of a body of text by establishing associations between those terms that occur in similar contexts.[4]

The LSA approach is to perform a Singular Values Decomposition in order to gather the most important concepts of the data according to the singular values found within the diagonal matrix of the decomposition. Then we can look for the most used words for each concept in order to get to find a "label" for those concepts. We will only keep the most important concepts and reduce the dimension of the data since the data set is heavy and require high computation power.

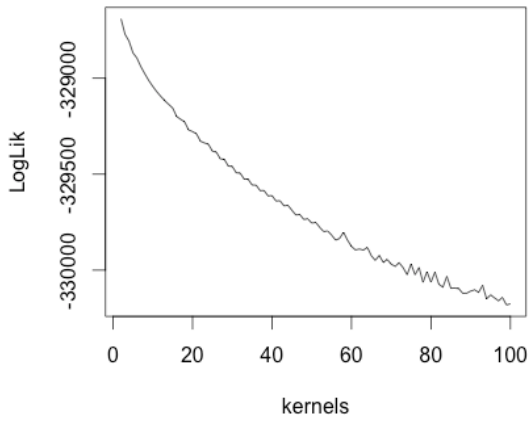
We created a term-document matrix which counts the number of occurrence of each term within a document. Here the documents are represented by our restaurants reviews. To be more accurate and obtain better results, we used the tf-idf weight for our term-document matrix. The tf-idf is a statistical measure which gives us the importance of a word in a review while taking in account

its overall number of occurrence within the whole corpus. Thus a very widely used word like a stem word won't have a big weight with the tf-idf matrix.

Then we perform the singular value decomposition. The SVD consists in the decomposition of the term-document matrix into three matrices, the term/concept matrix, the document/concept matrix and singular values matrix. We make the number of singular values vary in order to find the most suitable value for the rest of our algorithm. We also have to choose the number of dimension we keep for the reduced term-document matrix $M_k = U_k * D_k * V_k^T$ with k the number of dimension we keep from the SVD. Finally we need to cluster our data using the K-means algorithm. The K-means uses the Euclidean distance to assign the data to a centroid and then re-computes the centroids and iterates until convergence. We also have to choose the number of clusters in which we want to assign our data. In a further section we will perform experiments and evaluate the accuracy of our approach.

2. LDA (Latent Dirichlet allocation)

I used LDA to find out the main topic of these reviews so that we can decide the most influence aspects. LDA is a generative model that allows sets of observations to be explained by unobserved groups that explain why some parts of the data are similar. There are other similar models like independent component analysis, probabilistic latent semantic indexing, non-negative matrix factorization and Gamma-Poisson distribution. [5]



The first thing we want to decided is that make sure how many topics should we need. In other words, I want to find the best k value for LDA. To look up the methods, I find the Log-likelihood function to help me solve the problem. the log-likelihood can be used in place of the likelihood in maximum likelihood estimation and related techniques. To find the most reliable k value, I tried to use 2000 reviews to find it. (If I use a larger data, the computer would compute a very long time and get no result.) From the figure above, we can see that the LogLink decrease when the kernels increased. The reason may be that the datasets is small or the topics for reviews are hard to classify. From our opinion, at least the gradient of the curve decreased. It means that with the kernels increase, the similarity of the different topics is less.

However, this way may not be the best to label the concepts. The purpose of us is to find 3 or more main idea to represent reviews aspects. Let's see when k=5 and seed = 1000, the results I got after LDA.

```
      Topic 1   Topic 2   Topic 3   Topic 4  Topic 5
 [1,] "place"   "good"    "food"    "order"  "food"
 [2,] "good"    "realli"  "good"    "place"  "order"
 [3,] "great"   "get"     "place"   "great"  "like"
 [4,] "like"    "place"   "servic"  "back"   "realli"
 [5,] "chicken" "time"    "great"   "food"   "servic"
 [6,] "love"    "just"    "beer"    "servic" "will"
 [7,] "get"     "restaur" "like"    "like"   "time"
 [8,] "sauc"    "pizza"   "just"    "one"    "much"
 [9,] "went"    "back"    "even"    "time"   "alway"
[10,] "fri"     "night"   "restaur" "make"   "also"
```

The above topics showed that the place, time, food, some drink and service, which maybe the most important aspect the customer consider about.

LDA evaluation and improve rating:
The topics above showed that the most important words I appeared in some of the topics. It demonstrated that there are several topics in each review from the other aspects. There are two questions we want to figure out. 1) Is it possible that we can get a better result if we use tf-idf to the Document Terms Matrix? 2) Is it necessary to split the text to paragraph?

```
      Topic 1    Topic 2   Topic 3     Topic 4     Topic 5
 [1,] "sandwich" "roll"    "pizza"     "burger"    "sandwich"
 [2,] "amaz"     "egg"     "amaz"      "soup"      "fish"
 [3,] "happi"    "pasta"   "soup"      "breakfast" "taco"
 [4,] "steak"    "wonder"  "vega"      "sushi"     "pizza"
 [5,] "roll"     "happi"   "happi"     "thai"      "sushi"
 [6,] "burger"   "decent"  "wine"      "salsa"     "soup"
 [7,] "beer"     "shrimp"  "atmospher" "awesom"    "egg"
 [8,] "slice"    "super"   "beer"      "decent"    "roll"
 [9,] "italian"  "famili"  "awesom"    "husband"   "atmospher"
[10,] "excel"    "chines"  "excel"     "mexican"   "fast"
[11,] "famili"   "italian" "roll"      "cheap"     "select"
[12,] "spot"     "town"    "chef"      "shrimp"    "excel"
[13,] "spici"    "slice"   "steak"     "chip"      "mexican"
[14,] "fast"     "chip"    "sandwich"  "spici"     "amaz"
[15,] "sushi"    "pho"     "spot"      "famili"    "burrito"
[16,] "clean"    "eaten"   "tea"       "wing"      "breakfast"
[17,] "today"    "sushi"   "sushi"     "pork"      "wine"
[18,] "town"     "extrem"  "buffet"    "cold"      "beer"
[19,] "wonder"   "watch"   "pork"      "spot"      "salsa"
[20,] "trip"     "fast"    "crust"     "wish"      "phoenix"
```

For the first question, comparing to topics without tf-idf, the most significant difference is that more kind of foods appear in terms. It's little help to
tell the different topics. Meanwhile, I found a paper whose author Biel et.al [6] make a statement that tf-idf is unnecessary to LDA model. So it is nothing to do with the tf-idf.
The second question seems reasonable to solve the problem. One project did by our classmates use this way to find the topics of the
reviews. But they used the LSA. As we know, each document may be viewed as a mixture of various topics in LDA. And each document is assumed to be
characterized by a particular set of topics. This is akin to the standard bag of words
model assumption, and makes the individual

words exchangeable. [7] Meanwhile, Jack Linshi [0] did the data using LDA and also get the result which are similar to us. He then develops LDA to a so-called “CodeWord” LDA. Also Hitesh Sajnani [1] used a combination concept to define their topics.

I explored the topics variance with the increasing number of the topics.

	Topic 1	Topic 2	Topic 3	Topic 4
[1,]	"good"	"food"	"place"	"good"
[2,]	"just"	"place"	"chicken"	"food"
[3,]	"servic"	"order"	"restaur"	"like"
[4,]	"like"	"got"	"price"	"place"
[5,]	"great"	"can"	"order"	"get"
[6,]	"time"	"one"	"friend"	"time"
[7,]	"get"	"meat"	"like"	"great"
[8,]	"one"	"friend"	"great"	"eat"
[9,]	"order"	"realli"	"best"	"much"
[10,]	"will"	"great"	"back"	"love"
[11,]	"peopl"	"nice"	"time"	"back"
[12,]	"tri"	"come"	"one"	"pizza"
[13,]	"also"	"littl"	"realli"	"servic"
[14,]	"chees"	"servic"	"salad"	"salad"
[15,]	"sandwich"	"will"	"nice"	"meal"
[16,]	"want"	"restaur"	"just"	"side"
[17,]	"tabl"	"even"	"come"	"bar"
[18,]	"delici"	"just"	"look"	"pretti"
[19,]	"wait"	"sauc"	"tri"	"tri"
[20,]	"sure"	"get"	"even"	"realli"

	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
[1,]	"good"	"food"	"place"	"good"	"food"
[2,]	"servic"	"place"	"restaur"	"like"	"love"
[3,]	"just"	"got"	"chicken"	"place"	"just"
[4,]	"get"	"order"	"friend"	"get"	"order"
[5,]	"time"	"one"	"like"	"food"	"tri"
[6,]	"like"	"friend"	"great"	"time"	"bar"
[7,]	"great"	"can"	"order"	"great"	"price"
[8,]	"one"	"great"	"back"	"back"	"best"
[9,]	"will"	"will"	"time"	"much"	"can"
[10,]	"order"	"meat"	"one"	"pizza"	"sauc"
[11,]	"also"	"servic"	"get"	"servic"	"eat"
[12,]	"wait"	"realli"	"salad"	"meal"	"realli"
[13,]	"chees"	"restaur"	"price"	"pretti"	"place"
[14,]	"want"	"get"	"realli"	"salad"	"littl"
[15,]	"sandwich"	"come"	"night"	"eat"	"servic"
[16,]	"peopl"	"nice"	"look"	"fresh"	"great"
[17,]	"delici"	"make"	"just"	"first"	"nice"
[18,]	"made"	"even"	"come"	"server"	"meat"
[19,]	"tabl"	"dinner"	"nice"	"make"	"one"
[20,]	"sure"	"thing"	"best"	"bread"	"come"

In order to find the optimization topic number, I used different k values and compare them. These four pictures are k=4,5,6,7 and 10. We assign these words into several hidden concepts. From the four pictures above, although the most of the words appear in most of the topics, the latent words are not the same. When the number of topic increased from 4 to 5, the concept of health came out. But there is no new concept when the k =6, 7 and 10. From the picture that the number of topic equals 10, 100% are almost the same as the one with 5 topics. So there is a minimum topic number with most concepts.

	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6
[1,]	"good"	"food"	"place"	"good"	"food"	"great"
[2,]	"just"	"place"	"restaur"	"like"	"love"	"place"
[3,]	"servic"	"order"	"chicken"	"get"	"just"	"food"
[4,]	"get"	"got"	"order"	"time"	"bar"	"like"
[5,]	"time"	"one"	"time"	"food"	"price"	"friend"
[6,]	"like"	"can"	"like"	"place"	"best"	"servic"
[7,]	"one"	"restaur"	"friend"	"much"	"order"	"will"
[8,]	"order"	"come"	"one"	"eat"	"eat"	"back"
[9,]	"great"	"meat"	"back"	"great"	"tri"	"make"
[10,]	"also"	"realli"	"price"	"back"	"can"	"order"
[11,]	"peopl"	"nice"	"salad"	"salad"	"sauc"	"time"
[12,]	"want"	"friend"	"come"	"pizza"	"littl"	"get"
[13,]	"delici"	"get"	"great"	"first"	"realli"	"tri"
[14,]	"will"	"servic"	"get"	"servic"	"place"	"one"
[15,]	"restaur"	"even"	"realli"	"meal"	"nice"	"chees"
[16,]	"sandwich"	"littl"	"nice"	"bread"	"come"	"got"
[17,]	"wait"	"dinner"	"best"	"alway"	"meat"	"end"
[18,]	"well"	"great"	"just"	"realli"	"servic"	"good"
[19,]	"menu"	"menu"	"look"	"love"	"also"	"dish"
[20,]	"around"	"just"	"even"	"menu"	"went"	"pretti"

	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6	Topic 7
[1,]	"place"	"good"	"food"	"good"	"food"	"food"	"like"
[2,]	"great"	"food"	"just"	"love"	"good"	"place"	"good"
[3,]	"get"	"order"	"place"	"servic"	"time"	"good"	"place"
[4,]	"just"	"place"	"like"	"order"	"order"	"servic"	"back"
[5,]	"time"	"like"	"eat"	"great"	"get"	"restaur"	"food"
[6,]	"servic"	"realli"	"get"	"realli"	"one"	"like"	"time"
[7,]	"can"	"restaur"	"come"	"littl"	"great"	"price"	"one"
[8,]	"restaur"	"fri"	"menu"	"lunch"	"come"	"friend"	"just"
[9,]	"tri"	"will"	"pizza"	"sauc"	"servic"	"realli"	"eat"
[10,]	"will"	"love"	"think"	"tast"	"alway"	"great"	"chicken"
[11,]	"lunch"	"tri"	"want"	"get"	"friend"	"time"	"got"
[12,]	"make"	"time"	"order"	"got"	"back"	"tabl"	"tri"
[13,]	"pizza"	"just"	"best"	"back"	"dinner"	"nice"	"order"
[14,]	"definit"	"friend"	"tabl"	"can"	"tri"	"bar"	"menu"
[15,]	"one"	"pretti"	"two"	"even"	"best"	"chees"	"price"
[16,]	"realli"	"drink"	"love"	"meal"	"never"	"peopl"	"noth"
[17,]	"also"	"meal"	"star"	"fresh"	"staff"	"even"	"even"
[18,]	"roll"	"best"	"chees"	"chicken"	"salad"	"salad"	"can"
[19,]	"wait"	"one"	"well"	"first"	"make"	"right"	"alway"
[20,]	"know"	"bad"	"look"	"made"	"nice"	"star"	"come"

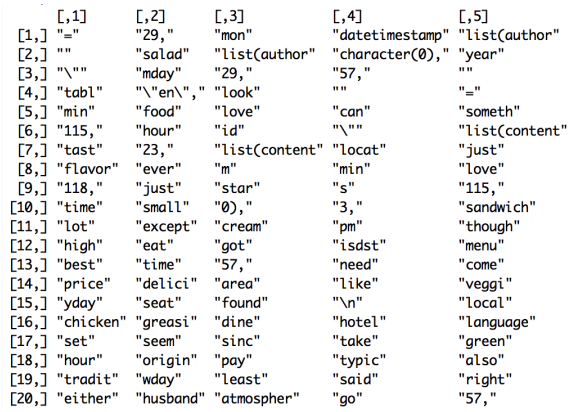
	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6	Topic 7	Topic 8	Topic 9	Topic 10
[1,]	"good"	"food"	"place"	"good"	"food"	"place"	"like"	"order"	"food"	"great"
[2,]	"just"	"place"	"restaur"	"like"	"bar"	"food"	"great"	"good"	"place"	"good"
[3,]	"servic"	"one"	"like"	"get"	"just"	"great"	"restaur"	"food"	"love"	"best"
[4,]	"get"	"order"	"one"	"place"	"price"	"like"	"order"	"will"	"make"	"time"
[5,]	"time"	"servic"	"order"	"food"	"order"	"servic"	"friend"	"restaur"	"come"	"love"
[6,]	"one"	"can"	"friend"	"time"	"tri"	"friend"	"place"	"just"	"realli"	"got"
[7,]	"like"	"get"	"just"	"servic"	"love"	"back"	"tri"	"get"	"good"	"sauc"
[8,]	"also"	"realli"	"time"	"much"	"eat"	"will"	"chicken"	"like"	"alway"	"tri"
[9,]	"well"	"dinner"	"get"	"pretti"	"realli"	"get"	"delici"	"time"	"night"	"can"
[10,]	"order"	"even"	"back"	"well"	"littl"	"make"	"night"	"chicken"	"price"	"pizza"
[11,]	"delici"	"got"	"price"	"back"	"place"	"one"	"back"	"menu"	"one"	"chicken"
[12,]	"peopl"	"just"	"even"	"first"	"servic"	"time"	"one"	"can"	"best"	"order"
[13,]	"wait"	"meat"	"realli"	"just"	"can"	"pretti"	"eat"	"come"	"much"	"dish"
[14,]	"great"	"come"	"come"	"eat"	"went"	"order"	"fri"	"price"	"time"	"want"
[15,]	"sandwich"	"friend"	"salad"	"great"	"well"	"tri"	"want"	"lunch"	"nice"	"salad"
[16,]	"tabl"	"littl"	"great"	"fresh"	"best"	"good"	"nice"	"back"	"eat"	"friend"
[17,]	"will"	"restaur"	"chicken"	"salad"	"one"	"just"	"time"	"one"	"dish"	"menu"
[18,]	"want"	"good"	"fresh"	"realli"	"even"	"chees"	"better"	"salad"	"restaur"	"meal"
[19,]	"tri"	"nice"	"look"	"alway"	"meat"	"wait"	"meat"	"thing"	"high"	"also"
[20,]	"lunch"	"wait"	"lunch"	"side"	"also"	"tabl"	"also"	"even"	"fri"	"get"

Evaluation	**Food**	**place**	**Service**	**Together**	**No meaning**
Good	food	Restaurant	Service	friend	Really
Like	meat	Place	staff		just
Well	salad				
Try	chicken	**Price**	**health**	**order**	**Bar**
Delicious	fresh	price	fresh	Order	Bar
Great	fry	worth		menu	drink
nice	pizza	high			

I used a topic result comes from the LDA with Gibbs sampling as the model to find the 5 concepts. Another partner analyze the performance of different sampling methods. She got the result that it was better to use Gibbs sampling. Figure shows below:

	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
[1,]	"place"	"food"	"food"	"good"	"food"
[2,]	"great"	"good"	"like"	"servic"	"good"
[3,]	"time"	"like"	"place"	"order"	"time"
[4,]	"get"	"place"	"just"	"great"	"one"
[5,]	"just"	"order"	"eat"	"love"	"order"
[6,]	"servic"	"restaur"	"menu"	"realli"	"get"
[7,]	"restaur"	"realli"	"come"	"got"	"great"
[8,]	"can"	"time"	"tabl"	"back"	"back"
[9,]	"tri"	"friend"	"get"	"sauc"	"come"
[10,]	"will"	"bad"	"want"	"even"	"friend"
[11,]	"one"	"tri"	"chicken"	"tast"	"like"
[12,]	"realli"	"fri"	"price"	"first"	"servic"
[13,]	"make"	"pretti"	"two"	"littl"	"alway"
[14,]	"rice"	"drink"	"star"	"chicken"	"tri"
[15,]	"anoth"	"will"	"friend"	"get"	"never"
[16,]	"lunch"	"just"	"think"	"lunch"	"dinner"
[17,]	"pizza"	"one"	"best"	"can"	"best"
[18,]	"also"	"love"	"chees"	"thing"	"salad"
[19,]	"wait"	"best"	"pizza"	"fresh"	"price"
[20,]	"cook"	"back"	"order"	"also"	"dish"

I also used another package called LDA to apply LDA to the data. And it's figure of the top words below:



	[,1]	[,2]	[,3]	[,4]	[,5]
[1,]	"="	"29,"	"mon"	"datetimestamp"	"list(author"
[2,]	""	"salad"	"list(author"	"character(0),"	"year"
[3,]	"\""	"mday"	"29,"	"57,"	""
[4,]	"tabl"	"\"en\","	"look"	""	"="
[5,]	"min"	"food"	"love"	"can"	"someth"
[6,]	"115,"	"hour"	"id"	"\""	"list(content"
[7,]	"tast"	"23,"	"list(content"	"locat"	"just"
[8,]	"flavor"	"ever"	"m"	"min"	"love"
[9,]	"118,"	"just"	"star"	"s"	"115,"
[10,]	"time"	"small"	"0),"	"3,"	"sandwich"
[11,]	"lot"	"except"	"cream"	"pm"	"though"
[12,]	"high"	"eat"	"got"	"isdst"	"menu"
[13,]	"best"	"time"	"57,"	"need"	"come"
[14,]	"price"	"delici"	"area"	"like"	"veggi"
[15,]	"yday"	"seat"	"found"	"\n"	"local"
[16,]	"chicken"	"greasi"	"dine"	"hotel"	"language"
[17,]	"set"	"seem"	"sinc"	"take"	"green"
[18,]	"hour"	"origin"	"pay"	"typic"	"also"
[19,]	"tradit"	"wday"	"least"	"said"	"right"
[20,]	"either"	"husband"	"atmospher"	"go"	"57,"

From the figure, we can see that there are still some numbers, whitespace and symbols. Actually I did the same preprocessing to the data.

Improving ranking:
The previous work from Hitesh Sajnani who comes from UC Irvine showed that they use the binary relevance to label the topics. Based on the label, they improve the rating.[5]

There are around 6 type of the restaurant which are ethnic, fast food, fast casual, casual dining, casual dining, family style, find dining.[8] There are some important characteristics for them. For example, the fast food are known as fast and its effectively order. The fast casual may concern more about worthiness and the menu. And the find dining has something to do with the health and service.

In the real world, when you say the price is good is not mean that the price is cheap but the reasonable. On yelp.com, you can easily find a restaurant cheap and expensive in search page. We have to find the most specific aspects and also as the most frequent words in the topics. From the topic figure above, I labeled the topic 1 as the fast and taste (several types of food), topic 2 as the bar and drink, topic 3 as the worth and menu, topics 4 as the health and service, and topic 5 as the together and easy-order.

Now, let's rating the business using the 5 concepts. Firstly, I have to create a matrix whose row is the ith review and column is topic. After that, filled the star of the review to the topic they have. Moreover, I calculate the mean rating of the same topics for same business id regardless the NA.

	row.names	fast and taste	bar and drink	worth and menu	health and service	together and easy-order
1	mVHrayjG3uZ_RLHkLj-AMg	3.645489	3.630118	3.614350	3.634440	3.622337
2	KayYbHCt-RkbGcPdGOThNg	4.500000	3.666667	NaN	3.333333	4.000000
3	wJr6kSA5dchdgOdwH6dZ2w	NaN	NaN	3.500000	NaN	3.666667
4	fNGIbpazjTRdXgwRY_NIXA	4.000000	5.000000	NaN	NaN	NaN
5	b9WZJp5L1RZr4F1nxclOoQ	4.375000	5.000000	4.375000	4.800000	4.800000
6	zaXDakTd3RXyOa7sMrUE1g	4.000000	5.000000	5.000000	NaN	2.000000
7	WETE_LykpcnrC1sFcQ5EGg	4.000000	2.000000	5.000000	1.000000	4.000000
8	rv7CY8G_XibTx82YhuqQRw	NaN	3.000000	4.000000	NaN	NaN
9	SQ0j7bgSTazkVQlF5AnqyQ	3.000000	1.500000	3.000000	2.000000	2.500000
10	wqu7ILomIOPSduRwoWp4AQ	5.000000	5.000000	NaN	4.000000	1.000000
11	P1fJb2WQ1mXoiudj8UE44w	3.400000	3.076923	3.285714	3.636364	4.111111
12	PK6aSizckHFWk8i0oxt5DA	NaN	1.000000	1.000000	NaN	1.000000
13	sRqB6flj3GtTZIZJQxf_oA	1.000000	2.000000	5.000000	3.000000	NaN
14	6ilJq_05xRgek_8qUp36-g	1.000000	2.181818	2.500000	2.500000	1.000000
15	MKyk4F4HSzHF8v-4cYe3Ww	3.583333	3.400000	3.800000	4.500000	3.000000
16	1qCuOcks5HRv67OHovAVpg	3.500000	4.000000	3.200000	4.000000	2.666667
17	sbW8qHJgzEIH42B0S-3New	2.428571	2.714286	3.333333	2.777778	3.250000
18	McikHxxEqZ2X0joaRNKlaw	2.571429	3.548387	3.363636	3.230769	4.000000
19	EoAY1JSVeJriBzId8lUQwQ	2.600000	2.666667	2.500000	3.200000	2.666667
20	TfvWSIAMFZ9zI3y2-K6wOA	NaN	2.000000	NaN	NaN	3.000000

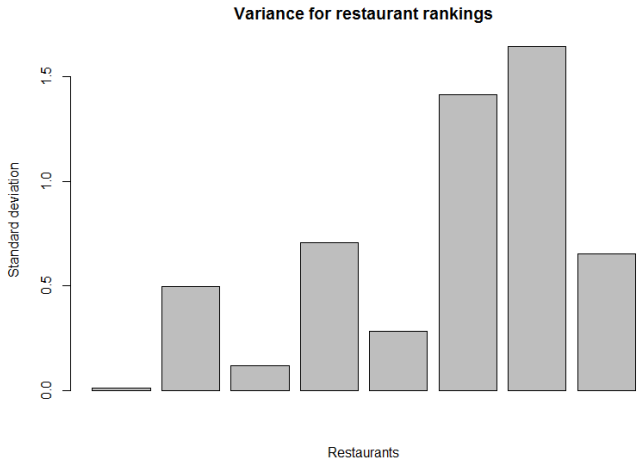
Finally, I got the matrix with the 5 new concepts rating for each business.
From the table, we can see that there are some missing values. It shows that the some restaurants have few reviews and the reviews have no concepts in particular topics. It's a simple algorithm to improve the ranking. Also my partner Tony did several experiments to find the threshold to recalculate the rating for the business. However we got similar rating.

Error analysis:
Since we are doing unsupervised learning we do not have any validation data. We have no idea to establish a confusion matrix to see whether our rating is reasonable or not. The only way we could do is to see some reviews and compare them to our improve rating.
We analyze some reviews according to the five previously introduced topics for nine different restaurants and see whether it's reasonable:

The first restaurant provided with rankings on the above table has ten reviews. This restaurant has a ranking of nearly 3.6 out of 5 stars for all the topics which is a relatively good grade. If we look at the food quality some comments give us some insight such as:
"best Fish Sandwich in Pittsburgh" and other 4 reviews mentioned the sandwich and launch means it quite fit for the fast and taste.
But for bar and drink doesn't make sense a little. However, the worthiness and menu looks good for the 3.6/5. Several customer reviews showed the concept.
Concerning the healthiness and service, here is some feedback:
"The service is sometimes hit or miss. Most of girls are good, one is very slow, and one is amazing. They are all friendly and usually a few different people will check in to make sure that you're happy"
This sentence indicates that the service is good overall, even though one of the waitress is slow. Hence the grading of 3.6/5 seems accurate for this restaurant.
In this way, we checked dozens of restaurants rankings by looking at the reviews. Although most of the rating make sense, the approach is not sufficient to verify the reasonable. I think the future work we have to do is to make a improvement to this approach that can use some services such as

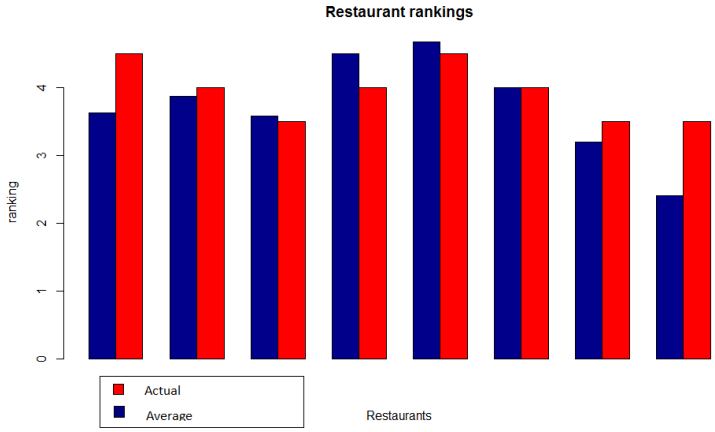
Amazon mechanical turk in order to label the reviews with a ranking for each topic and then test our data with training and testing.

We still want to find the standard deviation of the rankings in order to see if the ranking distribution has high variance meaning big differences of grading among the topics or uniform



grading among the topics.

Also we measured the average of the rankings through the different topics and compared it to the actual ranking.



Some standard deviations are very similar grades for different topics such a restaurants 1 and 3. The restaurants 6 and 7 have a different overall rating with the vary topics. This might indicate that some restaurants did several kinds of business. Restaurants with high variance may told us the business has clearly advantage and disadvantage reviewed by customers.

Furthermore, for my algorithm, based on the similarity of the topics after LDA, I can improve the rating calculation. I think it should be better.

Conclusion:

We basically accomplish our goal to improve the yelp ranking after do the LDA/LSA to find the latent concepts. Finally, we evaluate the improved ranking manually to see whether it is reliable. The ranking we measured are almost good for the improved ranking.

For our project, the most important work is to label the latent topics. LSA and LDA is quite powerful algorithm to implement especially in text mining. Although we didn't discuss LDA in class, I find a lot of information from papers and other resources. Moreover, text mining is a promising area. In real life, after the text retrieval by search engine, how to do text mining and get the information we want is very important. During the text mining procedure, we compared the different influence of weight frequency documents terms matrix and tf-idf matrix and get a deeper understand for these two concepts.

There are still some problems in our project and how to continue the future work. Firstly, the data we use is very large. My computer cannot deal with all these data so that we can make full use of cloud computing like AWS EC2 and other tools. Secondly, the topics we found maybe not very clearly, but we still try to make it more reasonable rely on previous work others did. Finally, we have to find a method to evaluate the improved ranking automatically.
.
The project provided us with a good opportunity to understand and apply data mining algorithms with a hand on experience using both python and R language

Reference:
[0] Jack Linshi, Personalizing Yelp Star Ratings: a Semantic Topic Modeling Approach

[1] http://www.ics.uci.edu/~vpsaini/

[2] James Huang et. al, Improving Restaurants by Extracting Subtopics from Yelp Reviews

[3] Julian McAuley, Hidden Factors and Hidden Topics: Understanding Rating Dimensions with Review Text

[4] http://en.wikipedia.org/wiki/Latent_semantic_indexing

[5] http://en.wikipedia.org/wiki/Latent_Dirichlet_allocation

[6] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. Journal of Machine Learning Research

[7] David M. Blei, Correlated Topic Models, 2006

[8] http://en.wikipedia.org/wiki/Types_of_restaurant

Appendix:

Python code: restaurantfl.py

```
import json

def sortRestaurants():
```

```
      #f = open('C:/Users/tofor_000/Desktop/Cours_IIT/Data
Mining/yelp_dataset_challenge_academic_dataset/xaa.json','r')


      data = []
      with open('Restaurants.json') as f:
        for line in f:
              data.append(json.loads(line))
      #print type(data[6])
      #json_obj = json.load(data)
      #print data
         output = open('RestaurantsReview.json','w')
         data1 = []
      with open('yelp_dataset_challenge_academic_dataset/yelp_academic_dataset_review.json') as
data_file:
             for line1 in data_file:
                 data1.append(json.loads(line1))


      for item in data:
            for item1 in data1:
        if item1['business_id'] in item['business_id']:
               #print 'YES Works'
               output.write(json.dumps(item1))
               output.write("\n")
        else:
               #print 'NOT WORKING :('
               continue

      output.close()
      f.close()
      data_file.close()

sortRestaurants()
```

R code:

```
library("rjson")

library("NLP")

library("tm")

library("class")

library("slam")




UnpackJSON <- function(filePath) {

 con <- file(filePath, "r")

 input <- readLines(con, -1L)

 jsonData <- sapply(input,fromJSON)

 close(con)

 df <- data.frame(jsonData)

 temp <- rownames(df)
```

```
 df <- as.data.frame(t(df))

 colnames(df) <- temp

 rownames(df) <- NULL

 return(df)

}


reviewdata <- UnpackJSON("~/Downloads/yelp_dataset_challenge_academic_dataset/reviewaa.json")


text <- reviewdata["text"]

text <- as.matrix(text)


textc <- Corpus(VectorSource(text), readerControl = list(reader = readPlain))


textc <- tm_map(textc, stripWhitespace)

textc <- tm_map(textc, content_transformer(tolower))

textc <- tm_map(textc, removeWords, stopwords("english"))

textc <- tm_map(textc,removePunctuation)

textc <- tm_map(textc, removeNumbers)

textc <- tm_map(textc, stemDocument)


reviewdtm <- DocumentTermMatrix(textc)

reviewdtm.tfidf <- tapply(reviewdtm$v/row_sums(reviewdtm)[reviewdtm$i], reviewdtm$j, mean) *
log2(nDocs(reviewdtm)/col_sums(reviewdtm > 0))

summary(reviewdtm.tfidf)

reviewdtm.new1 <- reviewdtm[,reviewdtm.tfidf >= 0.1]

reviewdtm.new1 <- reviewdtm.new1[row_sums(reviewdtm.new1) > 0,]

summary(col_sums(reviewdtm.new1))


reviewdtm <- removeSparseTerms(reviewdtm, 0.95)

rowTotal <- apply(reviewdtm, 1, sum)

reviewdtm.new <- reviewdtm[rowTotal>0, ]
```

```
hist(apply(reviewdtm.new, 1 , sum),xlab="Number of Terms in Term-Document Matrix", main="Number of Terms Per Review Corpus",
breaks=100)


library(SparseM)

library(RTextTools)

library(topicmodels)

library(lda)

library(ggplot2)

library(reshape2)


#Deciding best K value using Log-likelihood method

best.model <- lapply(seq(2, 100, by = 1), function(d){LDA(reviewdtm.new, d)})

best.model.logLik <- as.data.frame(as.matrix(lapply(best.model, logLik)))

plot(2:100, best.model.logLik$V1, type = "l", xlab = "kernels", ylab = "LogLik")


k <- 5

SEED <- 1000


reviewTM <-

  list(VEM = LDA(reviewdtm.new, k = k, control = list(seed = SEED)),

       VEM_fixed = LDA(reviewdtm.new, k = k,

                       control = list(estimate.alpha = FALSE, seed = SEED)),

       Gibbs = LDA(reviewdtm.new, k = k, method = "Gibbs",

                   control = list(seed = SEED, burnin = 1000,

                                  thin = 100, iter = 1000)),

       CTM = CTM(reviewdtm.new, k = k,

                 control = list(seed = SEED,

                                var = list(tol = 10^-4), em = list(tol = 10^-3)))

       )


ldadata <- lexicalize(textc)
```

```
keep <- ldadata$vocab[word.counts(ldadata$documents, ldadata$vocab) >= 3]

ldadata1 <- lexicalize(textc, lower=TRUE, vocab=keep)


result <- lda.collapsed.gibbs.sampler(ldadata$documents,k,ldadata$vocab,25,0.1,0.1)


top.words <- top.topic.words(result$topics, 20, by.score=TRUE)

top.words


N <- 10 #Choose the number of documents to display

topic.proportions <- t(result$document_sums) / colSums(result$document_sums)

topic.proportions <- topic.proportions[sample(1:dim(topic.proportions)[1], N,replace = TRUE), ]

topic.proportions[is.na(topic.proportions)] <-  1 / k

colnames(topic.proportions) <- apply(top.words, 2, paste, collapse=" ")

topic.proportions.df <- melt(cbind(data.frame(topic.proportions),

                                   document=factor(1:N)),

                             variable_name="topic",

                             id.vars = "document")

qplot(, value, fill=document, ylab="proportion",

      data=topic.proportions.df, geom="bar") +

  theme(axis.text.x = element_text(angle=90, hjust=1)) +

  coord_flip() +

  facet_wrap(~ document, ncol=5)

#To compare the fitted models we first investigate the values of the models fitted with VEM and estimated and with VEM and fixed

sapply(reviewTM[1:2], slot, "alpha")

sapply(reviewTM, function(x) mean(apply(posterior(x)$topics, 1, function(z) - sum(z * log(z)))))



Topic <- topics(reviewTM[["VEM"]], 1)


#top 5 terms for each topic in LDA

Terms <- terms(reviewTM[["VEM"]], 20)
```

Terms

```
my_topics <- topics(reviewTM[["VEM"]])

most_frequent <- which.max(tabulate(my_topics))

terms(reviewTM[["VEM"]], 10)[, most_frequent]

library ("ggplot2")
qplot(Topic, value, fill=document, ylab="proportion",
      data=topic.proportions.df, geom="bar") +
  opts(axis.text.x = theme_text(angle=90, hjust=1)) +
  coord_flip() +
  facet_wrap(~ document, ncol=5)

reviewTopicma <- matrix(, nrow = 2000, ncol = 5)
for (i in 1:length(reviewdata$text)) {
  reviewTopicma[i,Topic[i]] = reviewdata$stars[i][[1]]
}
first <- 1
last <- 1
buzRam <- c()
for (i in 2: length(reviewdata$business_id)) {
  if (as.character(reviewdata$business_id[i]) == as.character(reviewdata$business_id[(i-1)])){
    }
  else {
    last = i-1
    subRT <- reviewTopicma[first:last,  , drop = FALSE]
    buzRating <- colMeans (subRT, na.rm = TRUE, dims = 1)
    dim(buzRating) <- c(1,5)
    rownames(buzRating) <- as.character(reviewdata$business_id[(i-1)])
```

```
    buzRam <- rbind(buzRam, buzRating)
    first = i
  }
}

colnames(buzRam) <- c("fast and taste", "bar and drink", "worth and menu", "health and service","together and easy-order")

reviewdtm.new1 <- weightTfIdf(reviewdtm.new)

findFreqTerms(reviewdtm.new1, lowfreq=50)

library (RColorBrewer)
library ("wordcloud")
m = as.matrix(reviewdtm.new1);
v = sort(colSums(m), decreasing=TRUE);
myNames = names(v);
k = which(names(v)=="miners");
myNames[k] = "mining";
d = data.frame(word=myNames, freq=v);
wordcloud(d$word, colors=c(3,4), random.color=FALSE, d$freq, min.freq=50);
```