Final Year Project

Leveraging Latent Factors for Predicting User Preferences from Restaurant Reviews

Suman Adhya
Roll Number: 2018SMCS008
Registration Number: 201802050102032

July 7, 2020

Supervisor : Dr. Partha Basuchowdhuri

Int.Master's-PhD/Master's Program in Mathematical and Computational Science

School of Mathematical and Computational Sciences

Indian Association for the Cultivation of Sciences

Kolkata-700032

Contents

I. Introduction

Crowd sourcing of information plays an important role in our lives. One such system is a restaurant review website, where people provide their evaluation of eateries on a point-based system as well as in text format. Although the liking and disliking of an individual may depend on several factors, the rating has no way of representing those components. But the text review often mentions several aspects, which are crucial for understanding the reviewer's judgment. Extracting such factors and suitably storing them to understand the preferences of each user can be a challenging task. Such restaurant review websites often recommend the reviewers the restaurants, where the reviewer has not yet visited (more accurately, has not yet written reviews for) but would probably like to go to based on the likings and dislikings they have expressed in their previous reviews. Such a system is known as a restaurant recommender system. A recommender system is used to suggest items such as product, music, movie, etc. to a user. One main reason for using a recommendation system is that on the internet a large amount of data generated daily and it is not easy to get meaning out of such a large volume of data. So we try to extract important aspects from this large amount of data, we try to visualize how the rating depends on other features of an item. There are already well-known machine learning algorithms available for recommendation system where the extract this hidden features and tried to learn the model for predicting the preference score but our approach is to extract those hidden factors and tried to uncover them i.e. to transform this unsupervised problem to a supervised one so that the model can be tunned well as per the requirement. In this project, we have collected restaurant review information from Zomato, to understand the preferences of the reviewers or users. Our approach is to build up a network of users and then detect user communities from that network. We studied the state-of-the-art models for finding the hidden factors (also known as latent factors) of user reviews using LDA(Latent Dirichlet allocation) model and use a matrix factorization method to predict unknown ratings. The goal is to create a novel method that would predict unknown rating with a better understanding of the aspects (entities mentioned in the review).

II. Problem Statement

Nowadays we have a lot of data available about user-item affiliation. To make revenue and to provide better user experience we need to process these data to get meaning full inferences. The only way to predict unknown ratings from previous experience is we need to understand what are the factors that make a user put preference scores about an item. These factors may be some features of the item or some personal choices of a user. A good prediction is possible only when we can extract these user preferences and the item characteristics and makes our model learn those factors. Now to look into the user preferences we realize that there are a lot of internet users who actively engaged in purchasing products, watching online streaming and listening music, but not all of them are distinct from each other, that is there are several user groups and every individual of the same group has their common preferences, likings, and dislikings. Our task is to build a users network where two users have a common edge if they have some similarity and our next task is to extract user's communities(clusters) from that network, where a community stands for similar users group. Now getting meaningful communities from a graph is always a complicated task. Then we tried to understand the hidden factors behind those communities, i.e. to uncover the latent factors which make the users form a community and the factors which make the communities dissimilar to each other.

III. Literature Review

SVD(Singular Value Decomposition) is a matrix factorization method where any rectangular matrix A can be decomposed into three factors $U, \Sigma, \ and\ V^T$. For any matrix A the best rank k approximation is given by SVD. In [2] which is the popular paper for challenging Netflix movie recommender system and won \$1 million prize for 10% improvement on Netflix, their approach is also similar but with a slight modification. Actually they take A as the rating matrix where $((a_{ij}))_{m \times n}$ is rating for i^{th} user for j^{th} movie. Now in SVD the missing entries treated as 0, which is not convenient for predicting ratings, that's why they come with another approach; they randomly initialize two matrices P and Q and calculate the RMSE value, i.e. let A be the actual rating matrix with a lot of missing entries(one user may not rate all movies) and A' be the matrix such that, $A' = P^T \cdot Q$ (Figure1)

$$\therefore RMSE = \frac{1}{|A|} \times \sqrt{\sum_{ij \in A} (a_{ij} - a'_{ij})^2}$$

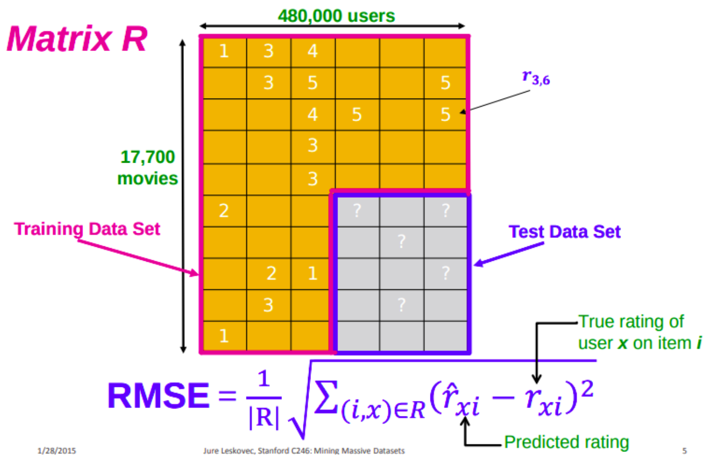

Matrix R
480,000 users
17,700 movies
$r_{3,6}$
Training Data Set
Test Data Set
True rating of user *x* on item *i*
Predicted rating
RMSE = $\frac{1}{|R|}\sqrt{\sum_{(i,x)\in R}(\hat{r}_{xi} - r_{xi})^2}$
1/28/2015
Jure Leskovec, Stanford C246: Mining Massive Datasets
5

Fig. 1: RMSE [3]

Therefore the optimization is,

$$min_{P,Q} \sum_{ij \in A} (a_{ij} - a'_{ij})^2$$

Now there is still an issue of over fitting so for regularization, the use they modified it such as,

$$min_{P,Q} \sum_{ij \in A} (a_{ij} - p_i^T \cdot q_j)^2 + [\lambda_1 \cdot ||p_i||^2 + \lambda_2 \cdot ||q_j||^2]$$

Then, for optimization purpose, they use stochastic gradient descent and the steps are: Say,

$$e_{ij} = a_{ij} - p_i^T \cdot q_j$$

- $p_i \leftarrow p_i + \gamma(e_{ij} \cdot p_i - \lambda_1 q_j)$
- $q_j \leftarrow q_j + \gamma(e_{ij} \cdot q_j - \lambda_2 p_i)$

Those they get the two matrices P and Q which product gives the predicted rating matrix.

Now In our project we treat the output of SVD as a tripartite graph structure and then from that tripartite graph structure we take the bipartite structure $U \; and \; \Sigma$ to create a user network, this idea is inspired from [1]. In their paper, they create a generative model named AGM which can convert any bipartite graph(may be weighted or may be unweighted) into a network. For unweighted bipartite graph i.e. a graph where the edges between the users and the community have a connected edge but not the strength of the connection, the did it this way: Create a probability set,

$$\{p_c : \forall c \in C \vee \sum_{c \in C} p_c = 1\}$$

where the p_c denotes the weighted probability of the community $c \in C$. Therefore the more the probability is the more the community has connected with users. Now, to build the user's network $G(V, E)$ where V is the set of all user nodes and E is the edge probability the did,

$$for\ some\ u, v \in V,\ p(u.v) = 1 - \prod_{c \in M_u \cap M_v} (1 - p_c)$$

where M is the set of all communities, M_u is the of communities of u and M_v is the set of all communities of v.
This $p(u, v)$ is the edge probability between the nodes u and v of V. If u and v shares no community between them then,

$$p(u, v) = \epsilon$$

This is denoted as ϵ-community.

But in our case, the users connected to some communities have membership strength. To convert a weighted bipartite graph to a network we use the idea from [4] where the implement the model BigClam which can convert large weighted bipartite graphs into a network. The idea is that, let U be the non-negative matrix of connection strengths, i.e. u_{ij} denotes the i^{th} user strength for j^{th} community. Therefore the edge probability between the user u and v is defined by:

$$p(u, v) = 1 - e^{-U_{uc} \cdot U_{vc}},\ for\ some\ c \in C$$

Therefore for all such common communities $c \in C$ we have,

$$\forall c \in C, p(u, v) = 1 - e^{-\sum_c U_{uc} \cdot U_{vc}}$$

Which can also be re-written as:

$$p(u, v) = 1 - e^{(-U_u \cdot U_v)}$$

IV. Data Set

A. Description of data set

Data set	#Unique users	#Unique restaurants
Full data set	185346	6488
Training data set	160394	6325
Testing data set	61258	5037

B. Visualization of data set

1) Rating Distribution

The user rating distribution of restaurant of Zomato, Kolkata are shown in the Figure2.

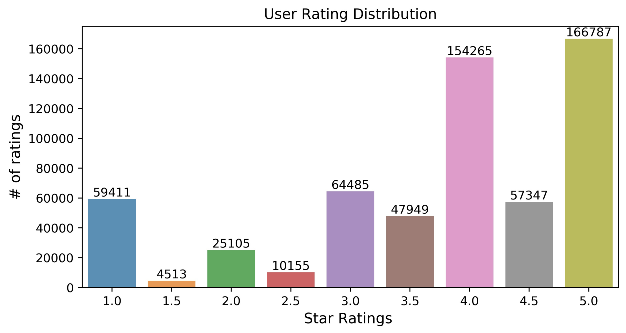


Fig. 2: Rating distribution

Inference:

From this distribution we can conclude that most of the restaurants has either 4.0 stars or 5.0 stars.

2) Popular Cuisines

The top 10 popular cuisines in Kolkata are shown in Figure 3. We generate heatmaps according to their latitude and longitude for first two most popular cuisines.

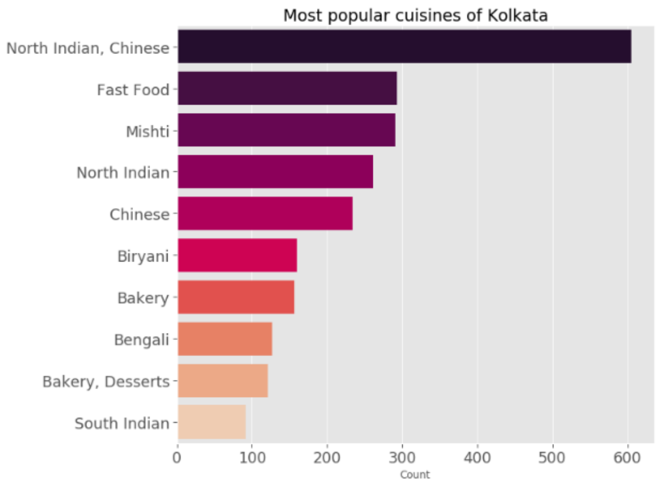


Fig. 3: Popular Cuisines

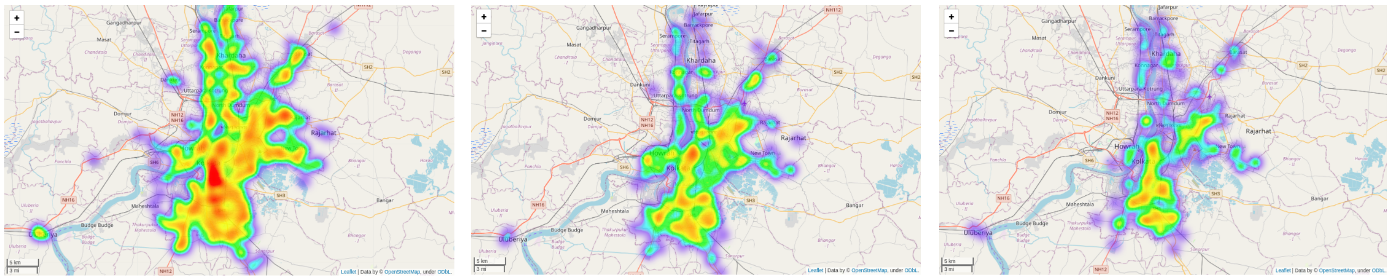


Fig. 4: (a)Heatmap of all restaurants in Kolkata; (b)Heatmaps of all restaurants North Indian and Chinese cuisines; (c)Heatmaps of all fast food restaurants

V. Methodology

A. SVD

Singular Value Decomposition is a data dimensionality deduction method. SVD is the fundamental idea behind PCA(Principal Component Analysis) where we identify the actual number of independent factors behind our data set to predict some components. The idea behind SVD is if we have data matrix $A_{m\times n}$, we can factorize it into three factors, $U_{m\times k}, \Sigma_{k\times k}, V_{n\times k}$, Where $\Sigma_{k\times k}$ gives the variance of the data about the first principal axes.

$$A = U\Sigma V^T$$

where,

- $U_{m\times k}$ is a column orthogonal matrix($U^TU = I_k$) containing the users to concept vectors.
- $\Sigma_{k\times k}$ is diagonal matrix containing the singular values $(\sigma_1, \sigma_2, ..., \sigma_k)$ of the matrix AA^T s.t. $\sigma_1 \geq \sigma_2 \geq .. \geq \sigma_k > 0$
- $V_{n\times k}$ is a column orthogonal matrix($V^TV = I_k$) containing the restaurants to concept vectors.

If,

$$A = U\Sigma V^T$$

Then,

$$A^TA = (U\Sigma V^T)^T(U\Sigma V^T)$$

$$\Rightarrow A^TA = (V\Sigma^TU^T)(U\Sigma V^T)$$

$$\Rightarrow A^TA = V(\Sigma^T\Sigma)V^T$$

Again,

$$AA^T = (U\Sigma V^T)(U\Sigma V^T)^T$$

$$\Rightarrow AA^T = (U\Sigma V^T)(V\Sigma^TU^T)$$

$$\Rightarrow AA^T = U(\Sigma\Sigma^T)U^T$$

AA^T and A^TA both are symmetric matrix as, $(AA^T)^T = (A^T)^TA^T = AA^T$
and, $(A^TA)^T = A^T(A^T)^T = A^TA$. Since they are symmetric, they are diagonalizable. Therefore we have,

$$A^TA = PDP^T$$

where:

- D is a diagonal matrix, such that : $D = ((\lambda_i))_{n\times n}$
- P is orthonormal matrix, i.e. $PP^T = P^TP = I_{n\times n}$

Therefore we have,

$$PDP^T = V\Sigma^T\Sigma V^T$$

$$\Rightarrow \lambda_i^2 = \sigma_i$$

Similarly,

$$AA^T = QD'Q^T$$

where:

- D is a diagonal matrix, such that : $D' = ((\lambda_i))_{m\times m}$
- P is orthonormal matrix, i.e. $QQ^T = Q^TQ = I_{m\times m}$

Therefore we have,

$$QDQ^T = U\Sigma\Sigma^TU^T$$

$$\Rightarrow \lambda_i^2 = \sigma_i$$

Though the two matrices AA^T and A^TA don't have same number of eigen values but they both have same eigen values and the remaining eigen values are zero; i.e. if $(\lambda_1, \lambda_2, ..., \lambda_n, 0, 0, ..., 0)$ is a set of eigen values for A^TA then the set of eigen values of AA^T will be $(\lambda_1, \lambda_2, ..., \lambda_n)$ where $m \geq n$. Again, as the matrices are symmetric, the eigen values are non-negative.

SVD to dimensionality reduction: Eckart-Young-Mirsky theorem for low rank approximation says that: if rank(B) = k, then $||A - B||_2 \geq ||A - A_k||_2$, where $||\cdot||_2$ is spectral norm. Therefore, the first k factors of Σ are most significant with corresponding columns of U and V.

How many principal axes should we keep?

The main thumb rule is that we should keep 90% energy of our data set. By dropping the principal axes we are reducing our total data set energy. At first, we need to calculate the total energy which is the sum of the square of the singular values of Σ.

An example, if $\Sigma = [12.4, 9.5, 1.3]$. Then the total energy will be, $12.4^2 + 9.5^2 + 1.3^2 = 245.70$, while the lowest singular value the reduced energy will be, $12.4^2 + 9.5^2 = 244.01$. Thus, we have remained over 99% of the energy. Now, if we eliminate the second lowest singular value, 9.5, the reaminig energy would be only $\frac{12.4^2}{245.70} \times 100$ or about 63% , which is $< 90\%$ show we can not eliminate this singular value.

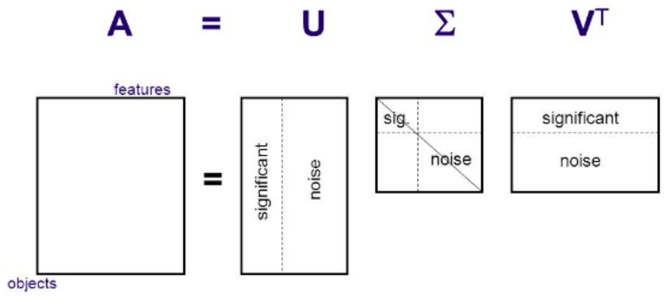


Fig. 5: Dimensionality reduction using SVD
https://images.app.goo.gl/x65RcjP9EE4vprgG7

Output of SVD gives a tripartite graph (Figure6; if we only take $U \, and \Sigma$ then we get a bipartite graph (Figure7). We then convert this bipartite graph to a users network by using the AGMfit algorithm and then use community detection to detect user communities in the network and similarly we can do the same for restaurants to group similar users and restaurants to a better understanding of the latent factors.

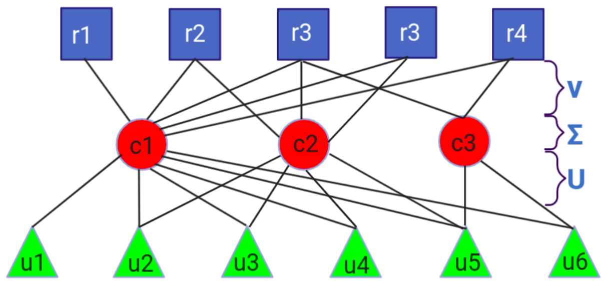


Fig. 6: Tripartite graph

B. Affiliation Graph Model

Affiliation Graph Model is generative model which is used to build a user's network from a large bipartite graph based on three components:

1) Communities from in a network when a group of users has same preference score. The bipartite graph represents this community membership where each users has a edge to a community.
2) The weight in the bipartite graph(Figure7) represents the users preference score for the community, so weights are non-negatives and the higher the weights are the higher the score for the user for that particular community.
3) The more two users share communities between them the larger the similarity between the two users.

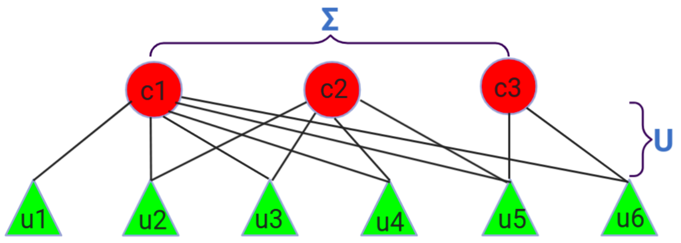


Fig. 7: Bipartite graph

How to define the edge probability of the network?
To convert a bipartite graph $B(V, C, M)$ into a graph $G(V, E)$ we need to define the edges E and for that we need to define the non-negative matrix F, where F_{uc} represents the weight between the node $u \in V$ and community $c \in C$. Now the edge probability between the node $u, v \in V$ will be(Figure9) :

$$p(u, v) = 1 - e^{-F_{uc} \cdot F_{vc}}$$

this is for a single community $c \in C$. Now for all $c \in C$ we have(Figure10),

$$\forall c \in C, p(u, v) = 1 - e^{-\sum_c F_{uc} \cdot F_{vc}}$$

Clearly from this equation, the more the u, v have common communities, the large the edge probability will be.

DEFINATION
IF F be a matrix such that $F_{uc} \geq 0$, where F_{uc} represents the weight between the node $u \in V$ and community $c \in C$. BigClam generates a network $G(V, E)$ from the given matrix F by creating edge (u, v) between a pair of nodes $u, v \in V$ with probability $p(u, v)$:

$$P(u, v) = 1 - exp(-F_u \cdot F_v)$$

where F_u is a weight vector for node u representing the weights for the community set.

The probabilistic interpretation of the above equation is: say an undirected graph with weighted edges, in which the pairs of nodes $u, v \in V$ have a dormant interaction of strength $X_{uv} (\geq 0)$, now suppose that nodes $u, v \in V$ originate an interaction $X_{uv}^{(c)}$ within each community $c \in C$ is a community using a *Poisson distribution* where the mean is$F_{uc} \cdot F_{vc}$. Then the total amount of strength of interaction X_{uv} between nodes u and v is the sum of $X_{uv}^{(c)}, \forall c \in C$:

$$X_{uv} = \sum_c X_{uv}^{(c)}, X_{uv}^{(c)} \sim Pois(F_{uc} \cdot F_{vc})$$

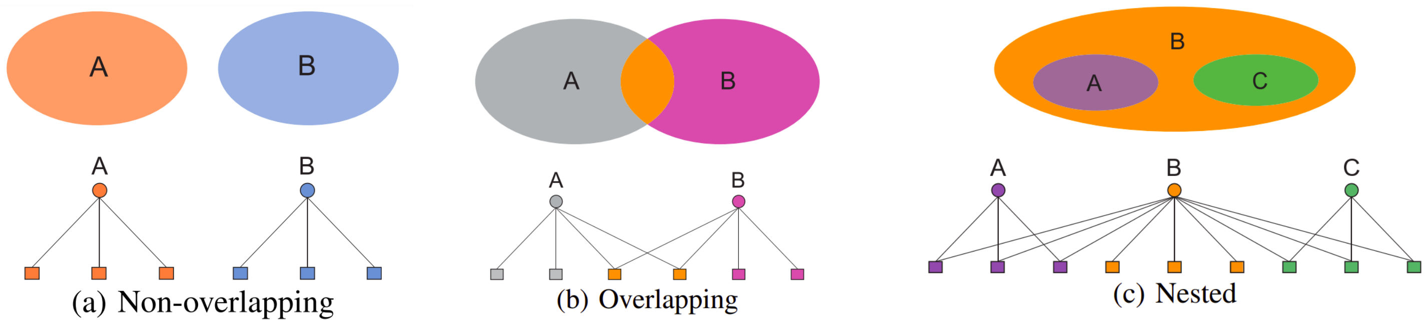


Fig. 8: (a)Non-overlapping community ; (b)Overlapping community; (c)Nested community [4]

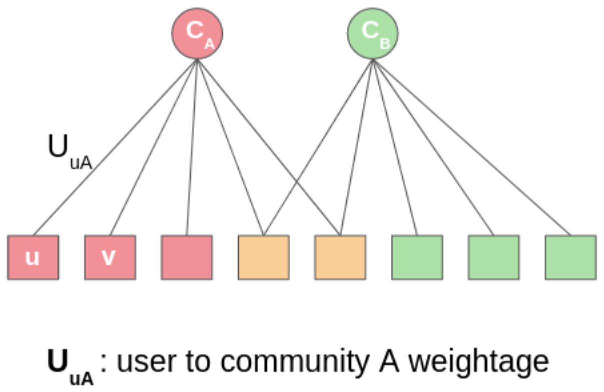


Fig. 9: u,v are two nodes of the network

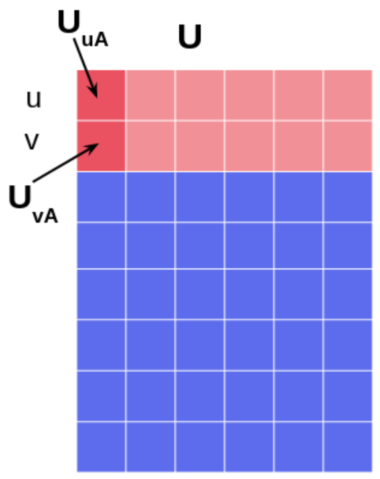


Fig. 10: U matrix from SVD

Then, applying the additive property of the Poisson random variable, we have $X_{uv} \sim Pois(\sum_c F_{uc} \cdot F_{vc})$, and the above equation will be:

$$P(X_{uv} > 0) = 1 - P(X_{uv} = 0) = 1 - e^{-\sum_c F_{uc} \cdot F_{vc}}$$

Now the over-all computation can also be done for the directed graph, with a little modification.

Adaptability of the AGM:

The Affiliation graph model can be used to model a broad-spectrum of large networks, such as:

1) **Non-overlapping communities** : In non-overlapping communities the ϵ-community allows for for edges between two nodes A and B(Figure 8(a)).
2) **Hierarchical communities** : Let's say we have three communities A, B and C, where community A and C are nested inside the community B. AGM can also use for this kind of community structure(Figure 8(b)).
3) **Overlapping communities** : In overlapping communities where two communities say A and B has some common nodes, it can be modeled easily in AGM(Figure 8(c)).

DETECT COMMUNITIES

Our aim is to detect communities from a unlabeled undirected graph $G(V, E)$ by fitting AGM. In other words our task is to fit AGM to $G(V, E)$ and get an another graph B with parameters $\{p_c\}$ by maximizing the likelihood $L(B, \{p_c\}) = P(G|B, \{p_c\})$ of the underlying graph G.

$$Arg\ max_{B,\{p_c\}} L(B, \{p_c\}) = \prod_{u,v \in E} p(u, v) \prod_{u,v \notin E} (1 - p(u, v))$$

Now we can apply gradient ascent by iterating the two steps:

1) Fixed B and update $\{p_c\}$
2) Fixed $\{p_c\}$ and update B

To start the process we first initialize B and to do this we generate a bipartite graph B on K communities ($K = |C|$) by using the model.

How to update $\{p_c\}$?

Atfirst we need to keep fixed the affiliation network B, our aim is to find $\{p_c\}$ by solving the given maximization problem:

$$Arg\ max_{\{p_c\}} \prod_{u,v \in E} (1 - \prod_{k \in C_{u,v}} (1 - p_k)) \prod_{u,v \notin E} (\prod_{k \in C_{u,v}} (1 - p_k))$$

where the constraint is, $0 \leq p_c \leq 1$. But this is now a non-convex optimization problem, so we need convert it into a convex optimization problem, and for that we have to maximize the *logarithm* of the *likelihood* by changing the variables $e^{-x_k} = 1 - p_k$:

$$Arg\ max_{\{x_c\}} \sum_{u,v \in E} log(1 - exp(\sum_{k \in C_{u,v}} -x_k)) \sum_{u,v \notin E} \sum_{k \in C_{u,v}} x_k$$

therefore the constraint, $0 \leq p_c \leq 1$ transformed into $x_c \geq 0$. This problem is now transformed into a convex optimization problem in x_c,and therefore we can find the global optimal solution of the above equation by applying stochastic gradient descent.

C. Topic Modeling by LDA

One of the important goals to deal with text data in machine learning is to understand the topics from the text data and LDA just does the same. Today we have a tremendous amount of text data in the format of reviews, transcripts, opinions, comments, feedback from the various sources like social media, newspaper, articles, books, customer complaints in e-commerce websites, etc. detecting topics efficiently from such data is always a challenging task, knowing what people talk about and users feedback about items we can take efficient steps easily. These opinions and customer reviews are highly valuable in any business model to improve user experience. As the data volume is too large to manually look at each text documents and find out the topics is not an efficient way. Therefore we need some algorithm where input is the text document set and output will be the topics about which the documents discussed. Topic modeling does the same. It looks at the document and learned the topics in a unsupervised way. Latent Dirichlet Allocation is used to generate topics from a set of documents. The idea behind LDA is, *"each document can describe as distribution of topics, and each topic can be described as a distribution of words"*. The working principle of LDA has been shown in the 11

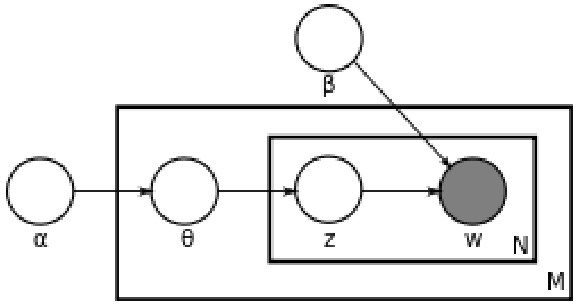


Fig. 11: Plate notation for Latent Dirichlet allocation. https://en.wikipedia.org/wiki/File:Latent_Dirichlet_allocation.svg

- M: is the number of documents.
- N: is the number of words in the given document.
- α: is the parameter of the Dirichlet prior on the per-document topic distributions. A high value of α implies every document is likely to contain a mixture of most of the topics and a low value implies every document is represented by a few of the topics.
- β:is the parameter of the Dirichlet prior on the per-topic word distribution. A high value of β implies every topic contains a mixture of most of the words and low value implies a topic contains a mixture of fewer words.
- θ_i: is the topic distribution for document i.
- z_{ij}: is the topic for the j^{th} word in document i.
- w_{ij} : is the specific word. The fact that W is grayed out means that words w_{ij} are the only observable variables and the other variables are latent variables.

LDA can recognize words appearing in documents and there is lots of document, other parameters are hidden (latent), one of them is **z**, a topic that is assigned with each word in the document. Thus the model understands topic, context, and words. LDA is a bag of words model so there are no syntactical rules here but still it can give the topics only by taking care of words.

1) Dirichlet distribution

Dirichlet distribution is the multivariate generalization of the Beta distribution. Here we discuss an example of a 3-dim problem, where we have 3 parameters in α that affect the shape of θ (distribution). For an N-dimensional Dirichlet distribution we have an N length vector as α. We can see how the shape of θ changes with different α values. For example, we can see how the top middle plot shows a similar shape to the α ground.

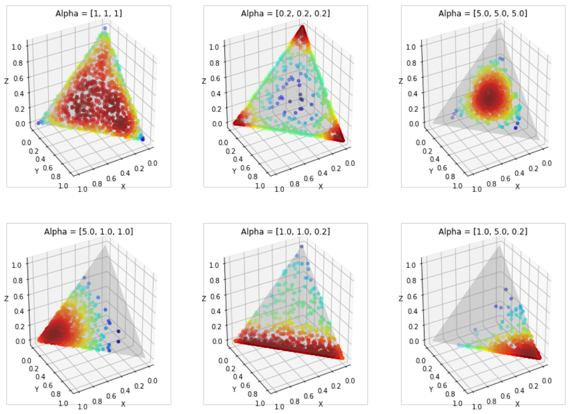
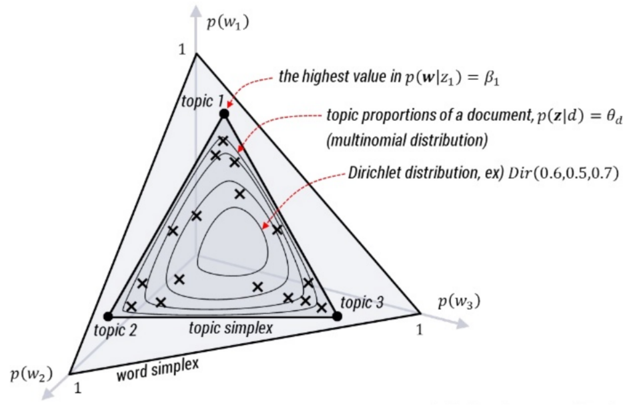


Fig. 12: Large α values push the distribution to the middle of the triangle, where smaller α values push the distribution to the corners. https://miro.medium.com/max/1400/1*3oOHy1tUfUT9Z379Alb9nA.png

2) How to evaluate the result?

The output of LDA gives topics and each topic is a distribution of words, but the topics have no name other than a unique id. But if we don't want to label them just want to compare them we can put them into space(LDA space). The space of LDA is simplex space, as all the topics have some probability distribution of the collection of words. The dimensionality of this space depends on the number of topics. The stronger



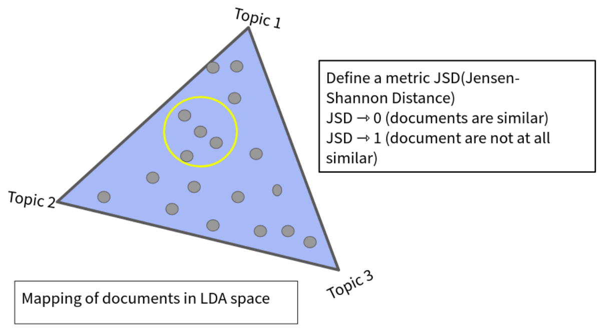
$p(w_1)$
1
the highest value in $p(\mathbf{w}|z_1) = \beta_1$
topic 1
topic proportions of a document, $p(\mathbf{z}|d) = \theta_d$
(multinomial distribution)
Dirichlet distribution, ex) $Dir(0.6,0.5,0.7)$
topic 3
$p(w_3)$
1
topic 2
topic simplex
$p(w_2)$
1
word simplex

Fig. 13: LDA space.
https://miro.medium.com/max/1400/1*_NdnljMqi8L2_
lAYwH3JDQ.gif

a certain topic is represented in a document, the closer the document is to the document corner. LDA space may have hundred of topics as a dimension which can not be visualized. Here for simplicity we give an example of an LDA space with only 3 topics. Now a single point doesn't make any sense on the space; mapping all documents into the space and defining a metric on the space we can imagine a certain threshold within which, each document will consider similar to each other. As a metric we can use **JSD(Jensen Shannon Distance)** here because it is used to measuring similarity between two probability distributions.

$$JSD = \sqrt{JensenShannonDivergence} \in (0,1)$$

- $JSD \rightarrow 0$ (Documents are similar)
- $JSD \rightarrow 1$ (Documents are not at all similar)



Topic 1
Define a metric JSD(Jensen-Shannon Distance)
JSD → 0 (documents are similar)
JSD → 1 (document are not at all similar)
Topic 2
Topic 3
Mapping of documents in LDA space

Fig. 14: Mapping documents to LDA space

3) Converting Unsupervised Output to a Supervised Problem by using LDA

LDA can be used to uncover the hidden semantic structure of an unsupervised problem and can convert it into a supervised one. First, we need to learn an LDA model and after that when we used the learned model to detect topics we will get output in such format: 100% Topic 0 or 20% Topic 0 +30% Topic 1 +40% Topic 2 +10% Topic 3. This output shows the distribution of topics for a given document. We used the LDA model to understand the hidden semantic structure of the communities and convert the unsupervised problem to a supervised problem.

VI. Experimental Setup

A. Data scraping

To collect data from the web we use these python libraries:

- **Requests:** The most elegant and simplest way to make an HTTP request to a server. The request package generates a request to the server from where data are needed to be scrapped.
- **Parser:** If the request to the server is granted then we will get the HTML content of the server. Now the HTML data is in nested format and it is not easy to retrieve our required data only by string matching. That is why we use this library to convert this complex nested data format to a tree-type structure and then collect the required data. We used html5lib as a parser. It is a Python library for pulling data out of HTML and XML files
- **BeautifulSoup:** Now from the tree-type structure data to navigating and searching the required data we convert into a BeautifulSoup object.
- **re:** Regular expression is used to collect the desired data from BeautifulSoup object by using string operations.
- **Selenium:** Selenium is used to create an automated web driver. As a web driver, we use a chrome driver with some manual customizations. It is used for automated clicks, logins and to collect the HTML data

At first, with the help of the request package, we send HTML requests to the server of zomato and then go through the page by page and extracts restaurant links. The in another code we go through each link to gather restaurant information like rating, number of reviews, the cost for two, address, latitude, longitude, the cost for two, cuisines, and rating and tags for ambiance, staff behavior, food quality, etc. Then using selenium we crawl the review text from each restaurant in Kolkata.

B. SVD

The size of our training set of rating matrix is 160331×6325. Before doing SVD we have to first compute the total energy of the data set which is sum of the eigen values of the matrix $A^T A$, where A is the rating matrix. Then we compute 70% of the total energy and choose the top k eigen values such that sum of the top k eigen values is $\leq 70\%$ of total energy and the same for top $k+1$ eigen values is $> 70\%$ of total energy. This k is the reduced dimension of our training data matrix. In our case we get $k = 610$. Then using svd package of scipy in python we perform SVD for $k = 610$ factors and as output we get a tripartite graph[5] U, Σ, V , where:

- U is a 160331×610 order matrix
- Σ is a 610×610 order diagonal matrix
- V^T is a 610×6325 order matrix

Here we get our tripartite graph as Figure 6, where $U_{160331 \times 610}$ gives the user to factor matrix for i^{th} user, $U[i]$

that is i_{th} column of matrix $U_{160331 \times 610}$ gives the i^{th} user to factor component. Similarly for matrix $V_{6325 \times 610}$ gives the restaurant to factor matrix and $\Sigma_{610 \times 610}$ gives the weight of the factors. Our next task is to create a network from this tripartite structure.

C. Affiliation Network and Community Detection

The matrix $U_{160331 \times 6325}$ is gives a bipartite graph which is a Users to factors matrix and Σ has the weights of the factors. To compute the network $G(V, E)$ as described earlier we first perform $U \cdot U^T$ and then compute $1 - e^{-value}$ for each elements in the matrix. Thus we get the affiliation network for users, our next task is to community detection. Here use the package Networkx in python and then to plot the data in gephi we then convert the the data format.

D. LDA

Used packages:

- **simple_pre-process**: A module in utils of Gensim
- **STOPWORDS**: a module in gensim·parsing·pre-processing; to preprocess the data using common stop-words in english grammar.
- **WordNetLemmatizer**: a NLTK tool to lematize the words.
- **SnowballStemmer**: another NLTK tool for stemming purpose.
- **porter**: another NLTK tool for stemming purpose.
- **WordPunctTokenizer**: for tokenization.
- **re**: re stands for Regular Expression, used to preprocess the data.

Steps:

- Create a corpus of all text reviews.
- Pre-process the text data to remove the emojis, special characters, confusing spellings of a same word and other unpleasant materials.
- Create a document-term matrix from whole review text data. This is vectorize the the text data use tf-idf vectorizer and count vectorizer.
- Create a dictionary of words and their unique ids.
- Using LdaModel of Gensim package learn our model on the our data set.

Our goal is to run the LDA model to our user communities and check what is similarity between the each members of a particular community, and which factors make these communities distinct from each other.

VII. Results

A. Affiliation graph and Community Detection

The user network that we build is plotted in Gephi [16] here the various colors denote the various communities. The steps for the data processing in Gephi are:

- At first we convert our network $G(V, E)$ which is of type Networkx in python into Gephi format and then import the file from Gephi.

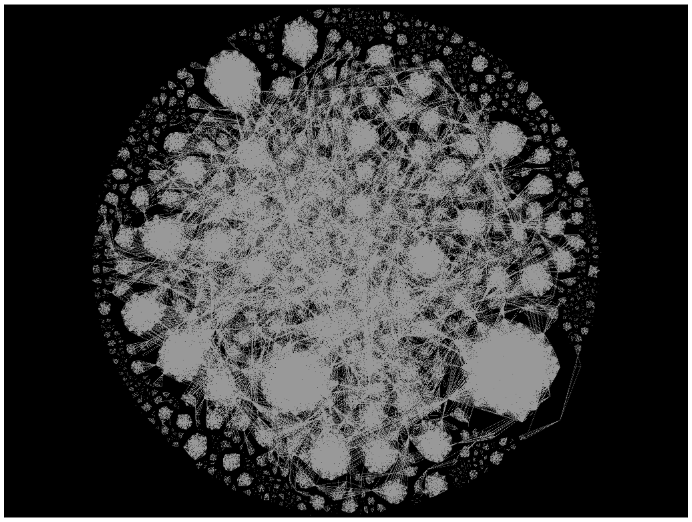


Fig. 15: User's network plotted in Gephi

- Then we select the visualization algorithm in Gephi to use. There are various algorithms in Gephi to be used and the algorithm selection affects the formation of the network. We used the Force Atlas Layout.
- After that personalize the network according to the needs by changing the colors, adding the node labels, etc.
- Then compute the network properties. In our research, the value which is calculated as network property attributes is the number of total nodes, the number of total edges, the average Degree, the average weighted values, the diameter of the network, and the number of distinct communities.

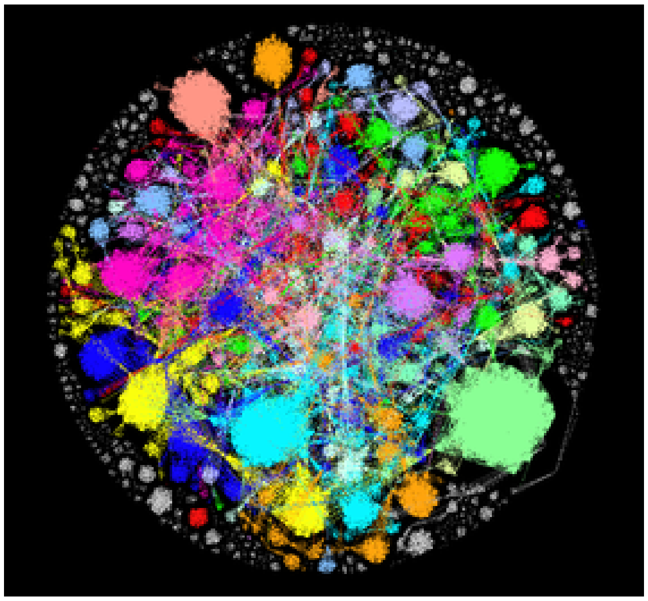


Fig. 16: User's community in gephi

Modularity report:

- Modularity: 0.901

- Modularity with resolution: 4.742
- Number of Communities: 1326

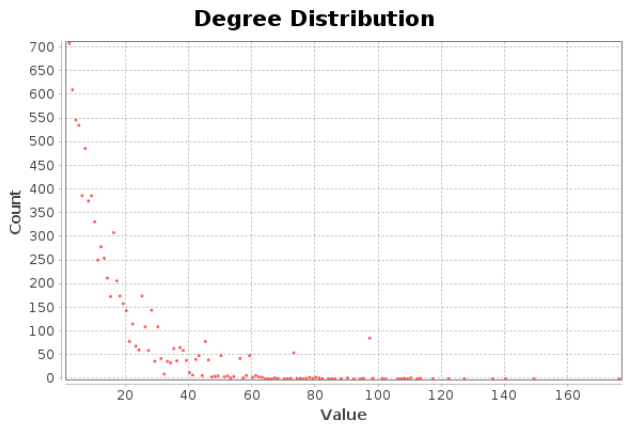


Fig. 17: Degree Distribution. Average degree: 15.880

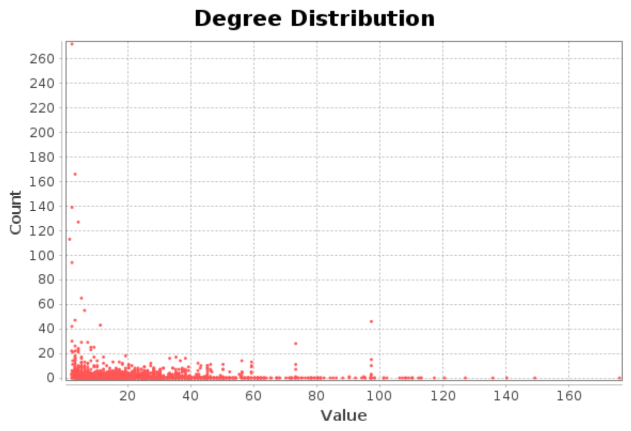
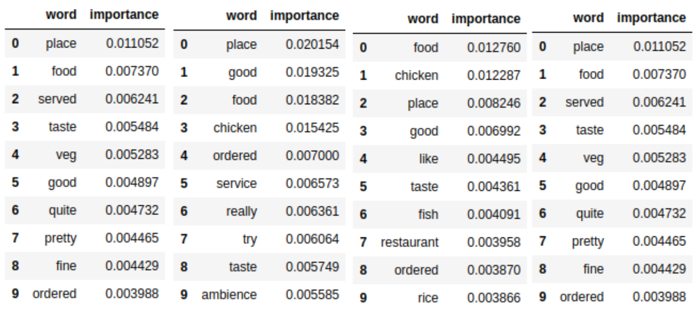


Fig. 18: Average Weighted Degree Distribution. Average Weighted Degree: 15.674

B. Topic Modeling

1) 1st ATTEMPT

- At first we create term to document matrix; i.e. if b_{ij} is in the matrix represent whether i_{th} term appears in j_{th} document or not. If appears $b_{ij} = 0$ o.w. 1. This is a very sparse matrix containing a lot of zeros.
- Then we create a dictionary of the all terms and their respective location in the term-document matrix.
- We put the term-document matrix into a new gensim format, i.e dataframe to term-document matrix and then convert it into gensim corpus.
- Gensim also requires dictionary of the all terms and their respective location in the term-document matrix.
- Now that we have the corpus (term-document matrix) and id2word (dictionary of location: term), we need to specify two other parameters as well, the number of topics and the number of passes.
- Now we have the corpus(term-document matrix) and id2word (dictionary of location: term). Now in python3 we import LdaModel from gensim run the model with number of topics 2,3 and 4 and number of number of passes = 10.



	word	importance
0	place	0.011052
1	food	0.007370
2	served	0.006241
3	taste	0.005484
4	veg	0.005283
5	good	0.004897
6	quite	0.004732
7	pretty	0.004465
8	fine	0.004429
9	ordered	0.003988

	word	importance
0	place	0.020154
1	good	0.019325
2	food	0.018382
3	chicken	0.015425
4	ordered	0.007000
5	service	0.006573
6	really	0.006361
7	try	0.006064
8	taste	0.005749
9	ambience	0.005585

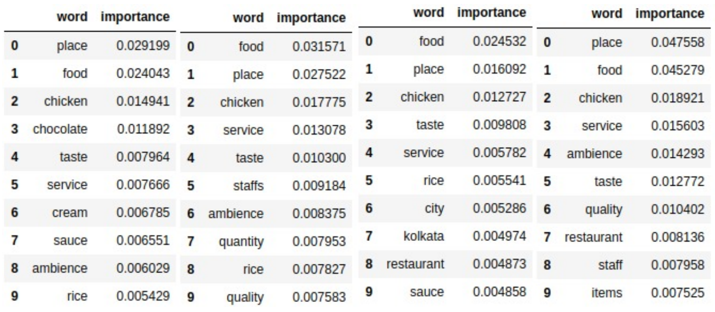
	word	importance
0	food	0.012760
1	chicken	0.012287
2	place	0.008246
3	good	0.006992
4	like	0.004495
5	taste	0.004361
6	fish	0.004091
7	restaurant	0.003958
8	ordered	0.003870
9	rice	0.003866

	word	importance
0	place	0.011052
1	food	0.007370
2	served	0.006241
3	taste	0.005484
4	veg	0.005283
5	good	0.004897
6	quite	0.004732
7	pretty	0.004465
8	fine	0.004429
9	ordered	0.003988

Fig. 19: Topic Distribution in 1^{st} attempt

2) 2nd ATTEMPT (Nouns only)

- To make the topics better we use with nouns only.
- We use nltk to pull out all nouns from the text and apply the nouns function to the transcripts to filter only on nouns.
- Create a new document-term matrix using only nouns.
- Re-add the additional stop words since we are recreating the document-term matrix.
- Recreate a document-term matrix with only nouns.
- Create the gensim corpus.
- Create the vocabulary dictionary.
- Again we import LdaModel from gensim run the model with number of topics 2,3 and 4 and number of number of passes = 10.



	word	importance
0	place	0.029199
1	food	0.024043
2	chicken	0.014941
3	chocolate	0.011892
4	taste	0.007964
5	service	0.007666
6	cream	0.006785
7	sauce	0.006551
8	ambience	0.006029
9	rice	0.005429

	word	importance
0	food	0.031571
1	place	0.027522
2	chicken	0.017775
3	service	0.013078
4	taste	0.010300
5	staffs	0.009184
6	ambience	0.008375
7	quantity	0.007953
8	rice	0.007827
9	quality	0.007583

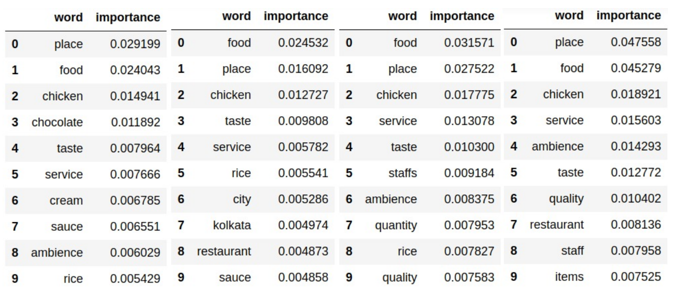
	word	importance
0	food	0.024532
1	place	0.016092
2	chicken	0.012727
3	taste	0.009808
4	service	0.005782
5	rice	0.005541
6	city	0.005286
7	kolkata	0.004974
8	restaurant	0.004873
9	sauce	0.004858

	word	importance
0	place	0.047558
1	food	0.045279
2	chicken	0.018921
3	service	0.015603
4	ambience	0.014293
5	taste	0.012772
6	quality	0.010402
7	restaurant	0.008136
8	staff	0.007958
9	items	0.007525

Fig. 20: Topic Distribution in 2^{nd} attempt

3) 3rd ATTEMPT (Nouns and Adjective only)

- For more clarity now we pull the adjectives along with the nouns.
- Apply the nouns and adjective function to the transcripts to filter only on nouns and adjectives.
- Create a new document-term matrix using only nouns and adjectives, also remove common words.
- Create the gensim corpus.
- Create the vocabulary dictionary.
- Once again we import LdaModel from gensim run the model with number of topics 2,3 and 4 and number of number of passes = 10.

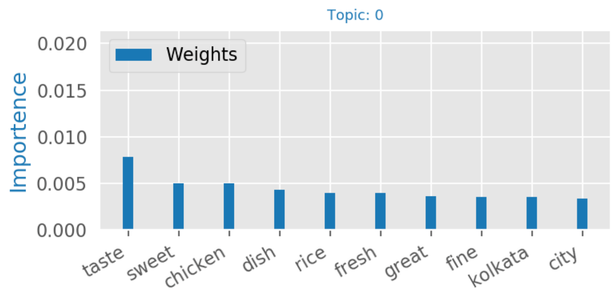


	word	importance		word	importance		word	importance		word	importance
0	place	0.029199	0	food	0.024532	0	food	0.031571	0	place	0.047558
1	food	0.024043	1	place	0.016092	1	place	0.027522	1	food	0.045279
2	chicken	0.014941	2	chicken	0.012727	2	chicken	0.017775	2	chicken	0.018921
3	chocolate	0.011892	3	taste	0.009808	3	service	0.013078	3	service	0.015603
4	taste	0.007964	4	service	0.005782	4	taste	0.010300	4	ambience	0.014293
5	service	0.007666	5	rice	0.005541	5	staffs	0.009184	5	taste	0.012772
6	cream	0.006785	6	city	0.005286	6	ambience	0.008375	6	quality	0.010402
7	sauce	0.006551	7	kolkata	0.004974	7	quantity	0.007953	7	restaurant	0.008136
8	ambience	0.006029	8	restaurant	0.004873	8	rice	0.007827	8	staff	0.007958
9	rice	0.005429	9	sauce	0.004858	9	quality	0.007583	9	items	0.007525

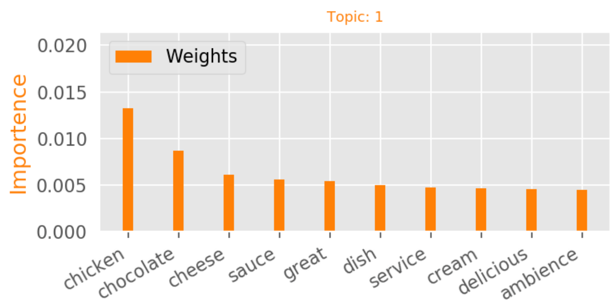
Fig. 21: Topic Distribution in 3^{rd} attempt

4) *FINAL ATTEMPT*

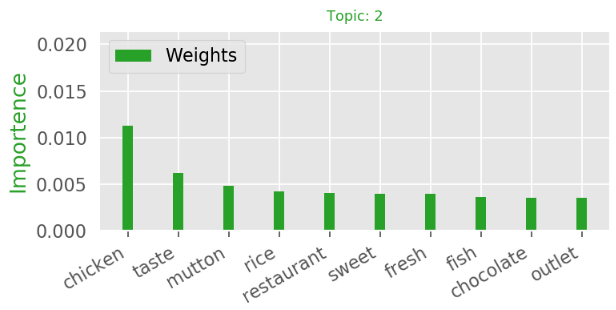
Out of the 9 topic models we looked at, the nouns and adjectives, 4 topic one made the most sense. So let's pull that down here and run it through some more iterations to get more fine-tuned topics. Here we import LdaModel from gensim run the model with number of topics 2,3 and 4 and number of number of passes = 80.



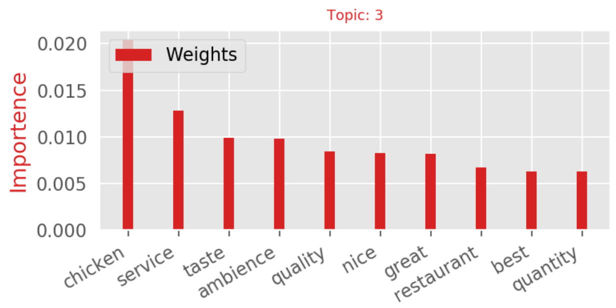
(a) Topic 0



(b) Topic 1



(c) Topic 2



(d) Topic 3

Fig. 22: Learned topics of LDA

REFERENCES

[1] V. CHESNOKOV, *Overlapping community detection in social networks with node attributes by neighborhood influence*, 06 2017, pp. 187–203.
[2] Y. KOREN, R. BELL, AND C. VOLINSKY, *Matrix factorization techniques for recommender systems*, Computer, 42 (2009), pp. 30–37.
[3] J. LESKOVEC, *Recommender systems: Latent factor model*, 2015.
[4] J. YANG AND J. LESKOVEC, *Overlapping community detection at scale: A nonnegative matrix factorization approach*, in Proceedings of the Sixth ACM International Conference on Web Search and Data Mining, WSDM '13, New York, NY, USA, 2013, Association for Computing Machinery, p. 587–596.

Acknowledgments

I would like to thank my supervisor, Dr. Partha Basuchowdhuri, for giving me the opportunity to work under him in this project and for the patient guidance, encouragement and advice he has provided throughout the semester.

I am also thankful to CCRES for their computational facilities.

I would also like to thank to Gautam Khanna who helped me in the data crawling process.

I am very grateful to Dr. Sanjay Kumar Saha to allow me to use the computational facilities of his own lab in Jadavpur University.

I would like to appreciate the support my lab members for their help and support.

School of Mathematical and Computational Sciences
Indian Association for the Cultivation of Science
Kolkata, Jadavpur, 700032

CERTIFICATE

This is to certify that the project report entitled **"Leveraging Latent Factors for Predicting User Preferences fromRestaurant Reviews** has been submitted by **Suman Adhya**, an Integrated Masters-PhD/Masters student of IACS, **Roll No.:2018SMCS008, Registration No.: 201802050102032**, in his Mater's project work under my supervision.

....
Signature of the supervisor with date and seal