|  |
| --- |
| **Yelp Trend Propagation: Finding like-minded Influential user to spread Information** |

**Akhil RavichandranGurumurthy Raghuraman**

Department of Computer Science Department of Computer Science

Arizona State University Arizona State University

Tempe, AZ 85281 Tempe, AZ 85281

[aravic12@asu.edu](mailto:aravic12@asu.edu)

**Narendra Kumar Subramanian Venkateswaran Ravikiran Tangirala**

Department of Computer Science Department of Computer Science Affiliation

Arizona State University Address Address

Tempe, AZ 85281

Nsampat1@asu.edusvenka71@asu.edu *email*

(if needed)

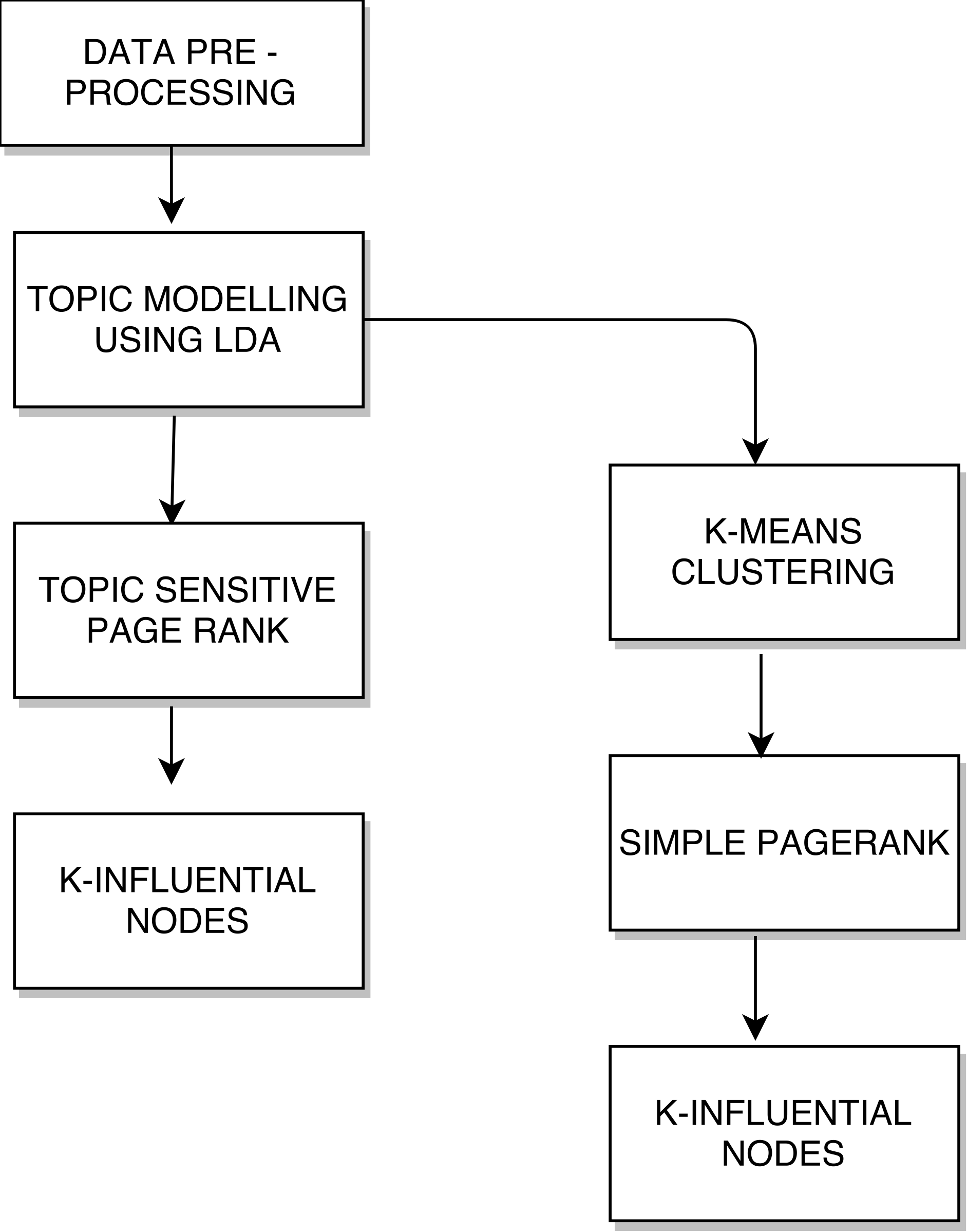
**ABSTRACT**

The number of online consumers who read and trust reviews are increasing day by day. Forbes reported that 88 percent of consumers trust online reviews as much as personal recommendation. Peers utilize the user experience and opinions to make decisions about where to go and what to buy. One of the major giants in recommending restaurants is Yelp. Many restaurants that made the list of top places to eat in 2017 are established only a year or two ago. A food or a restaurant, when newly introduced can suddenly become a sensation based on user reviews. In such cases, we want to maximize the spread of potential piece of information. This can be made by adding a given number of edges among the potential nodes. In a graph, the users are the nodes and the edges are the connections between users. We maximize the spread by recommending the information to the likeminded most influential node(user) in a graph. We extend the TwitterRank Algorithm [1] to a Topic Sensitive K-Means PageRank, a new methodology is proposed. This model ensures increased dissemination of information when compared to existing methodologies.

**1 INTRODUCTION**

Yelp is a Mobile/Desktop Application which publishes the crowdsourced reviews about local businesses. Yelp has gained huge popularity in 2010 when they saw the rise in people’s interest to write reviews. By 2015, The number of business reviews on Yelp has reached more than 90 million reviews [2]. Yelp offers a social network feel to its users who can become friends with each other, rate the review, comment about it, even become a fan of a reviewer. The “Fan” count of a reviewer particularly describes about how many users are fans of that reviewer. This helps in identifying whether the user is a fan of the reviewer’s reviews or an automatically linked friend from Facebook, Twitter etc. We consider only the food and restaurant reviews based on our project’s relevance. There are previous works to study about finding the most influential users in Twitter, that allows the search results to be sorted by the authority/influence of the users. Currently, majority of the research papers are focusing on the influence of the user based on the number of friends he/she has. This is not really a good metric to indicate the influence. Based on this methodology, the information when disseminated to the influential users, it is likely that the user skips the information in his/her timeline. In that case, it’s a direct miss. We propose a new methodology where the information will spread to likeminded influential users. This methodology has multifarious benefits. Firstly, it disseminates the information to the user who is likely to accept the information and disseminate to his friend’s circle. Secondly, when an influential user accepts this information, it is likely that his friend’s circle would be of similar taste to the user, consequently increasing the hit ratio in the consecutive propagation. Thirdly, it is also likely that the most influential user for a topic say, “Indian food”, would be socially connected to another influential user, so the information is given to the proper set of users with whom we can achieve the maximum information dissemination.

We propose an approach over the existing TwitterRank algorithm with the idea of layered set of information dissemination. The framework for our proposal is as shown in Figure 1. This clearly shows the existing methodology on the left and the new proposal towards the right.



**Figure.1.a Existing methodology and Our Approach**

We obtain the dataset from Yelp dataset challenge. We apply certain constraints with which we preprocess the data. We took the entire collection of reviews written by all the users in the dataset to form a single corpus. Then, the set of topics (10 Hidden topics) using Latent Dirichlet Allocation and a bag of 200 words for every topic was obtained. The process ran for 500 iterations till it got converged to set of words and their probabilities. Then, we obtain the list of topic and words for every user based on their reviews written and formed a matrix with users for rows and topics for columns. For each user, we count the number of words that fall into each topic. This gives the feature vector for every user. The users are clustered using K-means algorithm with the user’s feature vector to obtain the clustered users. The users in the same clusters are likeminded. Then Page Rank algorithm is used to identify the most influential likeminded user. Various similarity measures are applied between the reviewer and the influential user, obtained from the Twitter rank algorithm and our K-means PR (Page rank) algorithm. This provides proof that our algorithm provides the increased hit ratio which in turn produces increased information dissemination in a large graph database.

**1.1 RELATED WORKS**

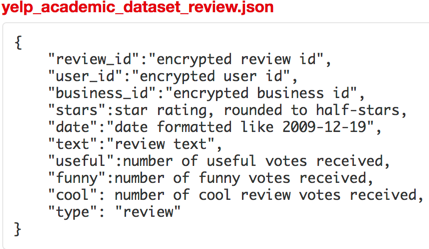
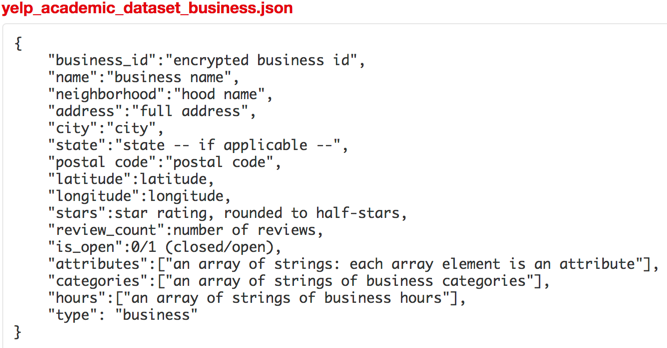
PageRank is the core concept of modern search engine. In the PageRank [3], the rank of a page is equivalent to the probability that a random surfer reaches this page. The random surfer choses a page with a probability P, then at each step, it choses another page to visit with probability P or uniformly at random choses one of the out-links of the previous page. A new approach, [4] Topic Sensitive PageRank has been the topic of research few years back. This method categorizes the documents into c classes and for each class, computes the rank vector based on the PageRank formula. Then during the query time, it computes the probability that input query lies in each category, to obtain the vector. The final rank vector is computed by linear combination of the obtained vectors and the rank vector.

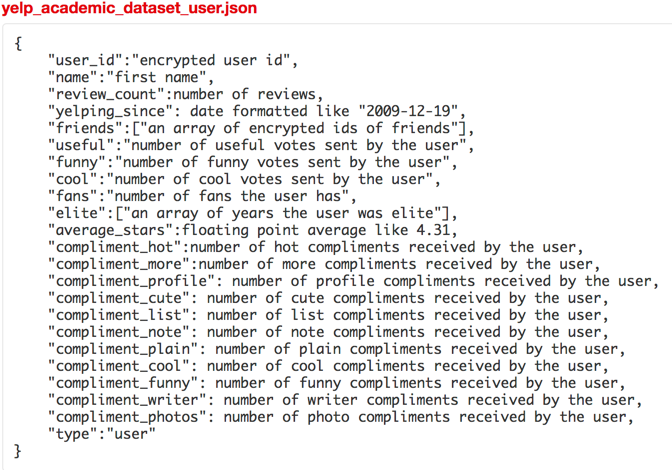
Latent Dirichlet Allocation (LDA) is a generative statistical model that allows set of observed topics to be explained by unobserved topics. A document usually contains series of topics which can be described by specific frequency or probability distribution. Several hidden topics is assumed to be present in the set of documents [5][6][7]. Each topic will have a probability distribution over the set of words. Latent Dirichlet Allocation (LDA) is a probabilistic generative model, which was first introduced by Blei et all. [7]. Latent Dirichlet Allocation shows quite good results of automatic identification of the subjects of documents. LDA uses additional assumption that the vectors and the documents are generated by Dirichlets distribution [8]. Texts are presented in the form of “bag of words” when the order, syntax and punctuation in the text are ignored. Topic combinations are iterated during the training process of the model. These combinations have the highest likelihood for those texts, which are input for this method. The result of the method is the number of topics. The method doesn’t create topics, it only allocates.

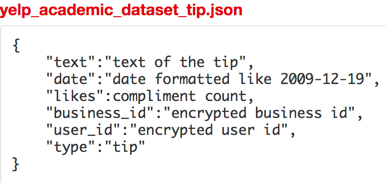
Jianshu et all. came up with a new algorithm called TwitterRank [9] which focuses on the problem of identifying the influential users of micro blogging services like Twitter. Twitter currently uses PageRank and Topic sensitive PageRank. The TwitterRank algorithm, an extension of PageRank algorithm which measures the influence of users by taking topic similarity between the users and the link structure. It is proven that the TwitterRank algorithm outperforms the current algorithms used by Twitter.

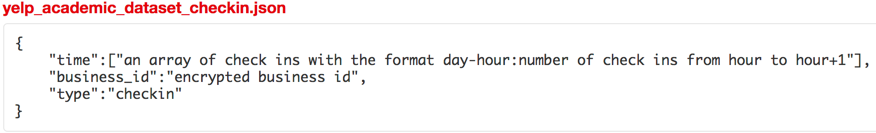
**2.1 DATASET**

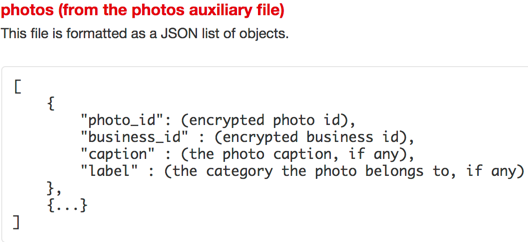
We have taken the masked dataset from Yelp. Yelp provides their academic dataset for the Yelp Dataset Challenge [8]. The data contains several tables with UserId, UserName, UserLocation, BusinessId, BusinessName, BusinessReviews and other required data. The sensitive data like UserId, UserName is masked to protect sensitive information that can be traced back to the Users or Businesses. The academic dataset provided by yelp contains many tables they are: Users, Business, Review, Check In, Tip and Photos. Each of these tables provide us with wide array of information regarding Users and Business. The Tables and the columns in the data set are listed below in JSON format [8][9]. We have considered some of the tables from the provided dataset, they include: Users, Business, Review, and Tips.





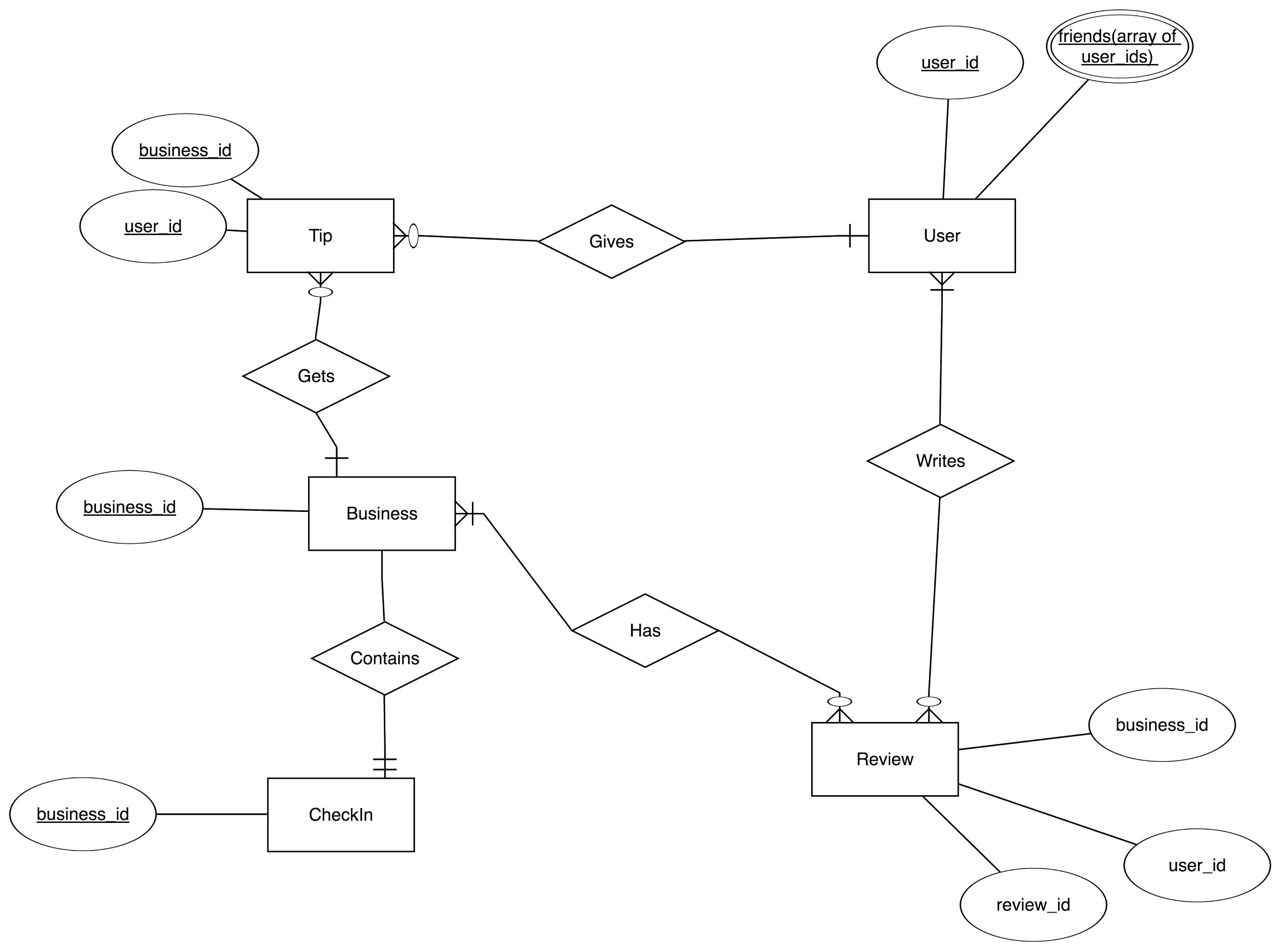






**2.1.1 DATA PREPROCESSING**

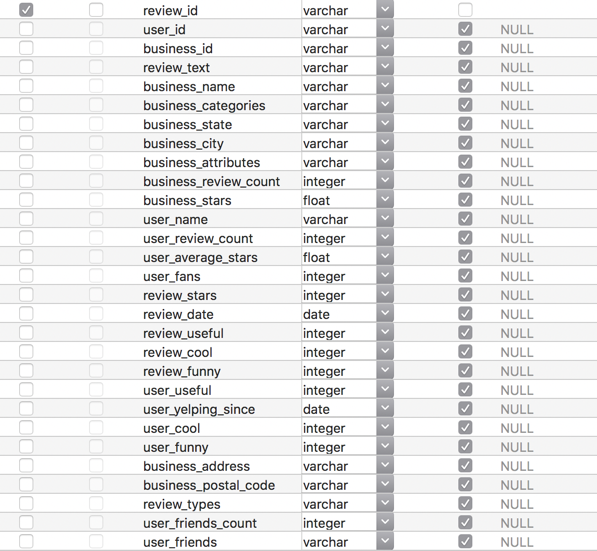
We used SQLAlchemy to load the data into an SQLite Database and performed the required data pre-processing and cleaning. We processed and converted all the JSON data provided by Yelp into a csv by invoking the data transformation code provided by Yelp [9]. We loaded all the required tables that is Users, Business, Reviews and Tips into a SQLite Databases using SQLAlchemy. Our most important requirement was to perform a join in regards to User Id and BusinessId with respect to the Reviews Table, Users Table and Business Table as depicted in the above ER Diagram as shown in Figure 2.1.1.a



**Figure 2.1.1.a Entity Relationship Diagram**

With each review, we need the user who posted it and in turn determine user data related to his friends, followers, rating, and fans. We also need the business data in regard to its location, business type, attributes, categories and other parameters to be used as label data to test our clustering techniques in regard to topic clustering. Hence, we performed a join with these three tables and created a new Table named Reviews\_Business\_Users and pruned the unnecessary columns. We have also calculated the user\_review\_count(total number of reviews for a given user) ,business\_review\_count (total number of reviews for a given business) and user\_friend\_count(total number of friends for a given user).

We used these calculated columns for further data pruning and considered data that is only relevant and required in our model. A more detailed explanation is provided on the constraints we used on the dataset we created in the next section. Additionally, we have provided a snapshot of the created table(Reviews\_Business\_Users) from our database as shown in Figure 2.1.1.b.



**Figure 2.1.1.b Database Schema**

**2.12 Constraint - Akhil**

Why did you chose this constraint (explain reasons) (1/4 page)

**2.2 TWITTER RANK**

Intuitively, the influence of twitterer from TwitterRank Algorithm can be applied to the influence of a yelp user. The Topic-sensitive measure proves to be the most widely used technique to provide recommendation based on the user’s interest. Considering these criteria, we have implemented the topic sensitive TwitterRank algorithm using Yelp dataset.

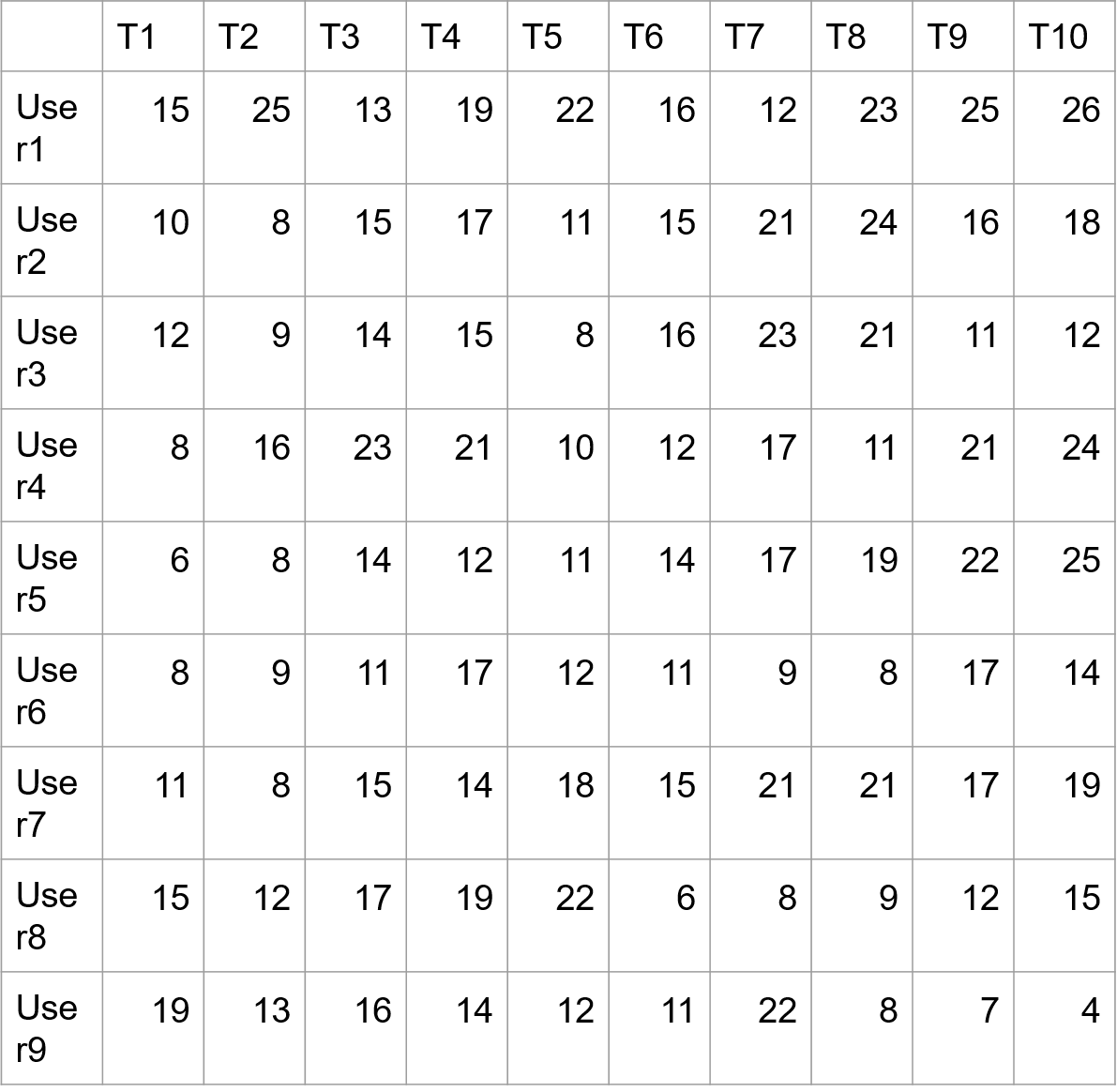
**2.2.1 TOPIC DISTILLATION**

The main aim of topic distillation is to identify a set of hidden topics and associate the bag of words to those topics obtained. For this purpose, we have used Latent Dirichlet Allocation (LDA), a generative statistical model to identify a set of topics and words until it converges to a fixed list so that there are no more changes in words associated to the topics. For simplicity, we have taken 10 topics and we have used the entire reviews collection from the Yelp dataset as the corpus. We used LDA model to run for 500 iterations so that it converges to a maximum extent. For each topic, we obtained 200 words. The list of topics and bag of words are stored separately. Similarly, we collate all the reviews posted by each user and apply LDA to the collective document. We obtain the bag of words for each user.

**2.2.2 FEATURE MATRIX**

Once we have the bag of words for every user as well the overall corpus, we build a User-Topic Matrix where the row stands for user and column stands for topics (t1...t10). Based on the set of words for each user, we find the number of unique words that has fallen into topic and increment the count by one. The words in the corpus is arranged based on the probability, so if a user’s word falls into topic t1 as well as topic t3, we obtain the index of the word match and find the smallest index such that it has higher probability and the corresponding word should belong to that topic.

If then , increment the count of t3



**Figure 2.2.2.a User-Topic Matrix**

Based on this method, we form the User-Topic matrix. A sample is shown in figure 2.21.a. In this matrix, each row signifies the user’s feature vector

**2.2.3 ADJACENCY MATRIX**

We take the Yelp Dataset and filter the list of users and their friends list. Based on the list obtained, we formulate a graph where the nodes represent the user and the edge represents the friend relationship between user. The graph is formulated in the form of adjacency matrix of (N x N) where N is the number of users. Each edge is given a weight which is calculated using the formula as follows

Given a topic t, the transition probability of random surfer from friend to is defined as

= 1 - ||

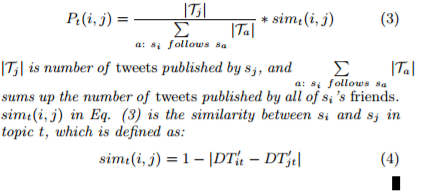
|| - Number of Reviews posted by

- Sums up the number of Reviews published by all ’s friends

Based on the formula stated, we obtain the adjacency matrix for all 10 topics. These topics sensitive adjacency matrix represents the strength of likeliness of the contents posted by user *i* and *j*. The adjacency matrix is then introduced to Page Rank Algorithm.

**2.2.4 TOPIC SENSITIVE PAGE RANK**

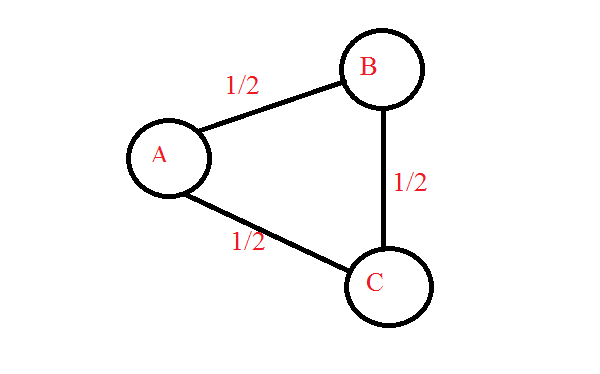
Based on [1], we form a network(graph) of the users with the nodes being the users and the edge signifies a friend relationship. we use the below formula to decide the edge weight between users:



The DT’ is a row normalized matrix of DT where DT is the user-topic matrix. It signifies the influence of each user on each topic. As we can see |DTit’ – DTjt’| is the difference of distance between user i and user j and 1-|DTit’ – DTjt’| signifies the similarity. So, if the topic influence of two users on two topics is almost the same then we can see that they have a high similarity. We formed a graph with user-friends relationship.

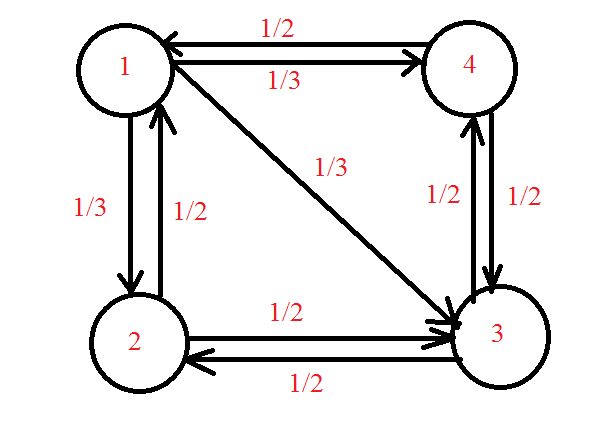
**2.2.4.1 PAGERANK, BRIEF EXPLANATION**

Before moving to the next section, we briefly discuss the PageRank algorithm for finding out the most influential node in a graph. First let’s go through PageRank in an undirected unweighted graph. Google first invented PageRank to bring ranking concept to the web. It is a random walk model. So, consider a node and then we follow any of the edges randomly, then the probability of following each edge will be the next question.



**Fig 2.2.4.1.A PageRank Illustration**

In Fig 2.2.4.1.A we see that the probability of moving from A to B and A to C is ½ as there are two edges of A. So, if we do a random walk from A then we have equal probability of moving to B or C. Also, PageRank has a damping factor. The damping factor signifies whether the user does continue the random walk or the user again starts the random walk. We have taken the damping factor as 0.85. The probability of continuing the random walk is 0.85 and the probability of restarting the random walk is 0.15. When we do this random walk a multiple time we get the PagRank scores of all the nodes. The PageRank score is the number of times a random surfer visits the node. If a node has high value then the random surfer visits that node multiple times and it is one of the most significant node.



**Fig 2.2.4.1.B PageRank Example**.

Consider another graph in Figure 2.2.4.1.B, Now the transition matrix for the above graph is as follows:

0 ½ 0 ½

1/3 0 ½ 0

1/3 ½ 0 ½

1/3 0 ½ 0

In the Matrix as depicted in Fig 2.2.4.1.C. A [2][3] signifies the 4th column, 3rd row signifies the probability of moving from node 4 to node 3 in the random walk. The formula for calculating the PageRank scores for the above transition matrix is as follows:

Algorithm:

Let the transition matrix be A.

Let the Teleportation vector be B. Initially all the nodes have equal teleportation probability. Thus B for fig 2 is:

0.25

B= 0.25

0.25

0.25

B = A\*B

Perform the above step until B remains constant.

If we closely see here B is nothing but an eigen value and this page rank score is the eigen vector corresponding to the eigen value 1.

One of the key problems of the above matrix is that for an unconnected graph some of the nodes will never be reached. To achieve this we introduce the damping factor. So the equation for PageRank can be written as

B= (1-p)A\*B + p\*T ---------------------------------------------(1)

The p in the above equation is

T is nothing but the Teleportation vector , in this example all the nodes have equal probability so it is a n X 1 column matrix with each value 1/n. n is the nodes in the graph.

**2.2.4.2 TWITTERRANK**

Twitter rank is similar to PageRank but there is a change in the teleportation vector. The equation for twitter rank is as follows:

TRt  = (alpha)\*Pt +(1-alpha)Et

Here Et is the teleportation vector.

Here Et  = DT’’ where DT’’ is a column normalized matrix of DT.

So if a user has a large contribution in a specific topic then there is a higher chance of teleporting to him in the random walk.

**2.2.4.3 AGGREGATION OF TOPIC SPECIFIC PAGERANK**

So now to get the most influential user for a specific user what we do is that we take the bag of words of the user and count the number of words in each topic. So we get the contribution of a user on each topic is given by:

Ct = (no of words in a topic t)/(total number of words by the user)

So the ration Ct is multiplied to the pagerank score of that topic and then we get the most influential users.

TR = ∑ Ct TR -----------------------------------------------------------(3).

Now the users with the highest score in TR are the mot influential user specific to the reviewer.

In the next section we see our approach to solve the similar problem.

**2.3 K-MEANS PAGE RANK**

One of the key points of observation in the above approach was that the influential users largely dependent on the number of connections in the network rather than the topic i.e. the top influential users remained the same for many test users even though their topic interests are the same. To try and reduce the effect of the network and give more weightage to the topic interests we propose a clustering based method. The flowchart in Figure 1.a has already shown the difference between our approach and the previous approach.

In our approach, we apply k means clustering after forming the Document Term Matrix DT. One of the advantages of applying the clustering method is that the users with the same interest are likely to fall in the same cluster. For e.g. if a user is interested in Mexican food and a quite atmosphere then such users will be in the same cluster. It takes users who may have multiple similar interest and puts them in the same cluster. One of the key issues of k means clustering is that we have to find an optimum value of k i.e. the number of clusters. To find the optimum number of clusters we use the elbow method. The elbow method plots k vs the objective function of k means clustering.

Objective Function of K Means = ∑i=1n|uj – xi |2 where n is the number of points and uj is the cluster center corresponding to the point xi. Now we plot a graph of k vs the objective function.\

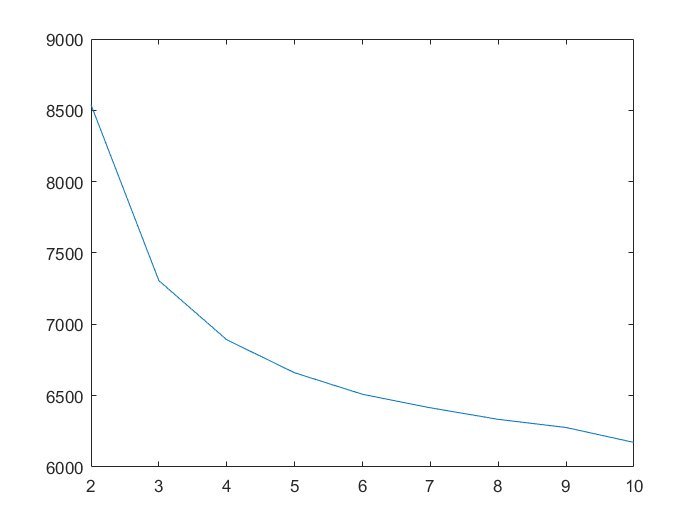
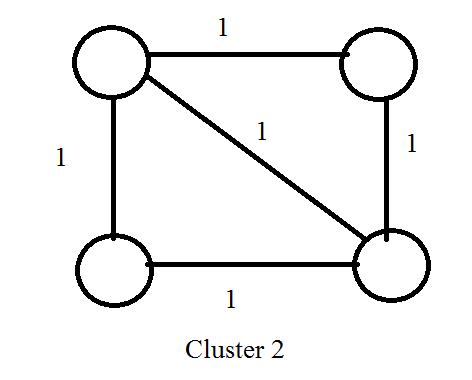
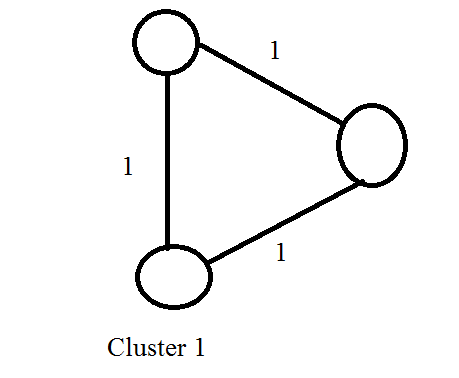


Fig 4: K vs Objective Function

Now we see that at k=5 the objective function quickly starts tapering down. So using the elbow method we can infer that k=5 is the optimum number of clusters.

**2.3.1 GRAPH FORMATION FOR K-MEANS**

In this approach, we have already clustered the users with similar interest. Now we form a graph and the edges of the graph are only within each cluster i.e. there is no edge between nodes of cluster1 and any other cluster. The edges of the graph take only two values i.e. 0 or 1. If 2 users have a friend relationship and both are in the same cluster we give the value of 1 between them otherwise we give a value of 0. This can be better explained with the diagram below:



In the above diagram we show two sample clusters and as described below an edge has a default value of 1.

**2.3.2 FINDING MOST INFLUENTIAL USER**

For finding the most influential user we take a test user and find the cluster in which he falls. This is found by comparing his word vector to each of the cluster centers and the one with the smallest distance is the one where he’s placed. Now in that cluster we form the graph as described above and then apply a simple page rank algorithm. The simple PageRank algorithm is similar to the one described above but there the weights here are 0 or 1 and also the teleportation vector starts with equal probability for all nodes in that cluster. The simple PageRank can be better explained with the equation below as shown in equation (1) before:

B= (1-p)A\*B + p\*T ---------------------------------------------(1)

The p in the above equation is

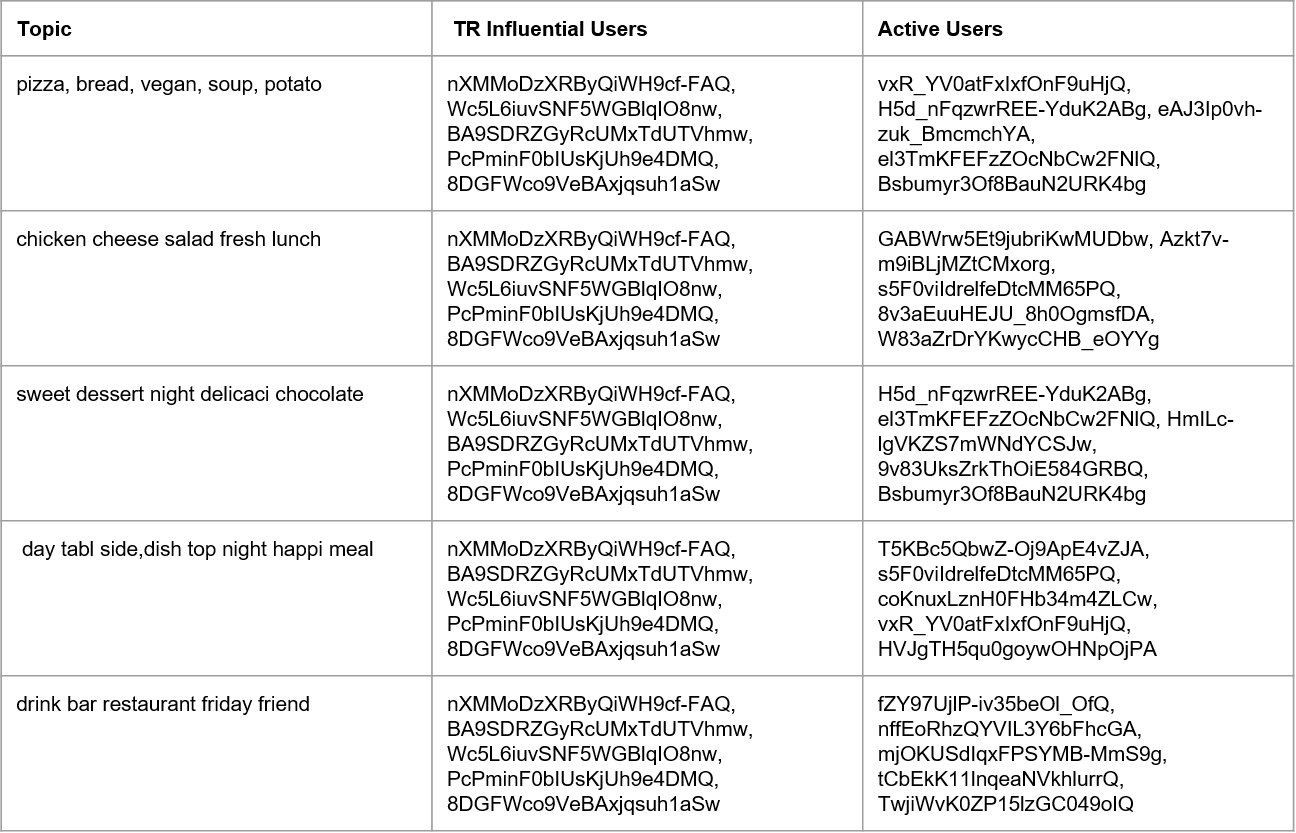
T is nothing but the Teleportation vector , in this example all the nodes have equal probability so it is a n X 1 column matrix with each value 1/n. n is the nodes in the graph.

Now the users with the highest PageRank scores are the most influential users relevant to the network.

**3. PROOF AND ANALYSIS**

**(1/2 page intro)**

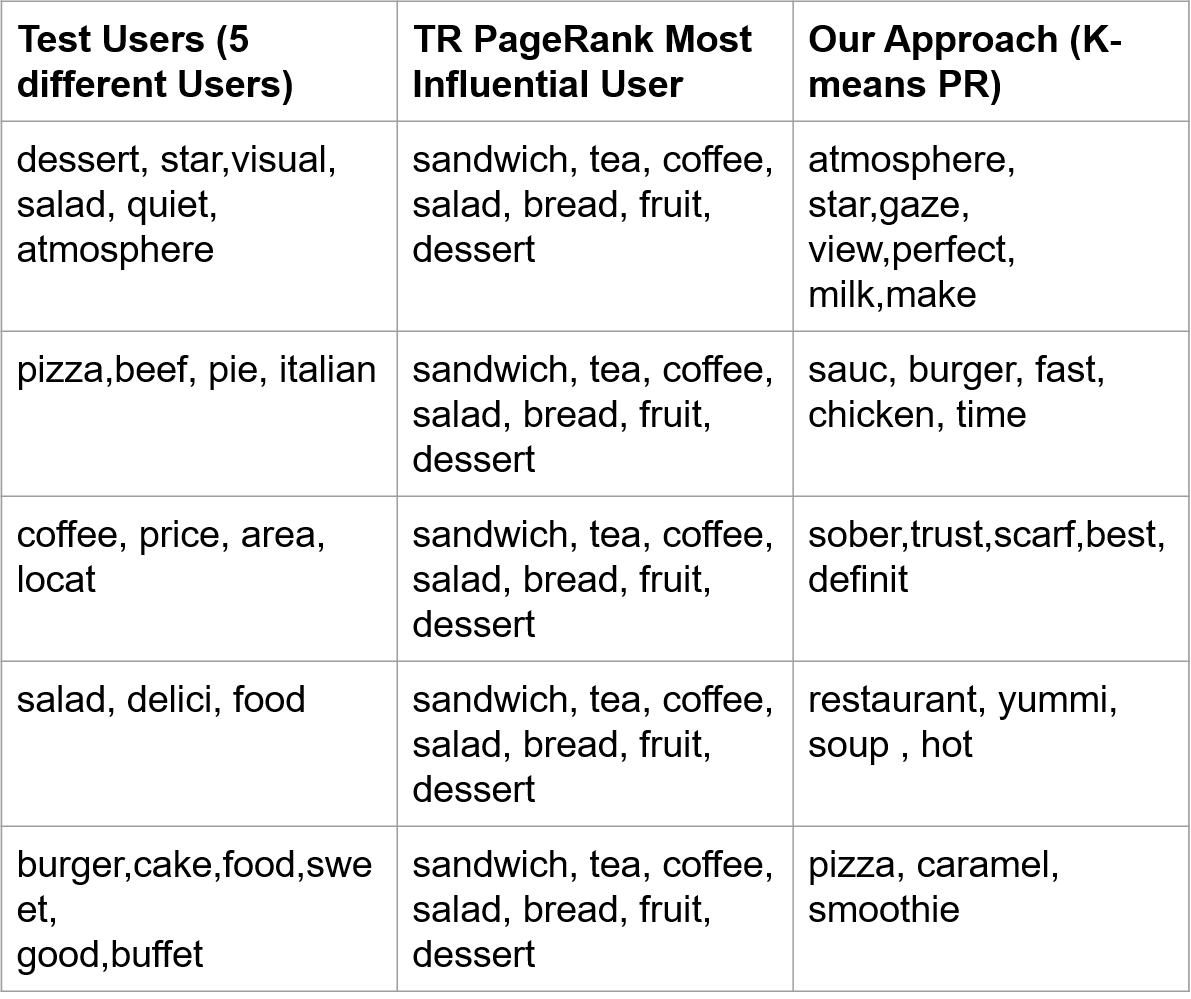
**3.11 RESULTS**



Active Users vs Influential Users

Here we show the active users vs the influential users, the active users are the users who have the most number of words in that topic whereas the influential users are the most influential users obtained via our first approach. We see that it is not necessary that the most influential user and the active users remain the same. Also we see that the most influential user remains the same in Influential users and this is because of the connections in the network and the teleportation vector.

Now we take 5 random test user and compare the bag of words obtained by approach 1 and approach 2.



Comparison of Approach 1 and Approach 2

In the above figure we see that the most influential user via the twitter rank approach remains the same but the influential user according to our approach changes as each of the user falls in a different cluster.

Now lets compare the bag of words obtained from approach 1 and approach 2 for a random test user

**Test data Topic words:**

flawless,dessert,fire,love,star,visual,sinc,info,good,tast,amaz,quick,gorgeou,place,dinner,person,friend,quiet,area,glass,salad,challeng,almost,atmospher,simpli,via,salon,half,keep,fill,immedi,vibey,call,filet,hill,fountain,easili,group,better,top,view,

**Topic Sensitive PR:**

food,order,friend,great,salad,healthi,littl,back,menu,sandwich,pretti,price,tea,say,option,use,mom,reason,enjoy,friendli,coffee,fri,flavor,fruit,bread,thai,rate,server,vegan,bar,asian,feel,recommend,glad,fish,last,wish,home,much,work,total,select,excel,meal,chicken,wasn,stevia,tast,dessert,pictur

**K-means PR:**

cappuccino,better,blue,heart,atmospher,tad,shop,consist,liter,master,fresh,time,spend,drive,short,tradit,want,stinkin,someon,though,star,gaze, view,perfect,sure,nice,color,milk,make,fell,love,thing,steam,back,seat,oz,afternoon,s,hello,shini,path,espresso,beaten,nutti,bake,close,expect,fri,awesom

If we see the bag of words here we see that the test data bag of words and the influential user 1 bag of words match on food tastes like dessert, salad(marked in red) whereas using our approach the bag of words match on finer details like the atmosphere of the restaurants the user likes.

In order to fully understand the outputs from each algorithm, the results obtained from each algorithm are compared using various similarity measures. Given the final data from each algorithm the chosen similarity measures are Euclidean distance, Jaccard coefficient and Dice coefficient. By taking the Jaccard coefficient and Dice coefficient values on an average from random users, we obtained values that represent the likelihood that a user would accept incoming information or (in this case) review from an influential user. These values are termed as the “hit ratio” as they help us to gauge the impact the influential user is having on his network of friends.

**3.11 SIMILARITY MEASURES**

**(1 and ½ page)**

Euclidean distance: This is basically the distance between two points that have any number of dimensions [10]. This gives a picture of how close the two points when represented by those features. The below table shows the results for each algorithm:

|  |  |  |  |
| --- | --- | --- | --- |
| Twitter rank | | K-means | |
| Node Index | Euclidean Distance | Node Index | Euclidean Distance |
| 6 | 30.659 | 676 | 7.141 |
| 13 | 31.686 | 938 | 5.568 |
| 18 | 29.597 | 998 | 6.164 |
| 37 | 26.153 | 767 | 5.196 |
| 8 | 30.806 | 659 | 7.937 |

**Table 1. Euclidean distance for each algorithm**

Jaccard coefficient: It is also known as the intersection over union and the Jaccard similarity coefficient, it is a statistic used for comparing the similarity and diversity of sample sets [11]. The below table shows the results for each algorithm:

|  |  |  |  |
| --- | --- | --- | --- |
| Twitter rank | | K-means | |
| Node Index | Jaccard Coefficient | Node Index | Jaccard Coefficient |
| 6 | 0.275 | 676 | 0.536 |
| 13 | 0.25 | 938 | 0.59 |
| 18 | 0.27 | 998 | 0.542 |
| 37 | 0.3 | 767 | 0.657 |
| 8 | 0.26 | 659 | 0.511 |

**Table 2. Jaccard coefficient for each algorithm**

The Sørensen–Dice index, also known by other names is a statistic used for comparing the similarity of two samples [12]. The below table shows the results for each algorithm:

|  |  |  |  |
| --- | --- | --- | --- |
| Twitter rank | | K-means | |
| Node Index | Dice Coefficient | Node Index | Dice Coefficient |
| 6 | 0.424 | 676 | 0.684 |
| 13 | 0.41 | 938 | 0.761 |
| 18 | 0.427 | 998 | 0.767 |
| 37 | 0.462 | 767 | 0.816 |
| 8 | 0.419 | 659 | 0.641 |

**Table 3. Dice coefficient for each algorithm**

**3.12 HIT/MISS RATIO**

**(1/2 page)**

Using the average results from each algorithm, we obtain the Jaccard Coefficient and Dice Coefficient to compute the acceptance ratio or the hit ratio. Based on 50 random reviewers taken, we calculated Hit accuracy as;

Jaccard Coefficient Hit Ratio: 32% in Twitter rank and 68% in K means page-rank algorithm

Dice Coefficient Hit Ratio: 36.77% in Twitter rank and 63.3% in K means page-rank algorithm

**3.11 SIMILARITY MEASURES**

**(1 and ½ page)**

**3.12 HIT/MISS RATIO**

**(1/2 page)**

**3.21 KMEANS PAGERANK**

**(1 and ½ page)**

**3.22 PROOF - Subbu (P.S.Akhil check if this is okay)**

**(1/2 page)**

The users are assigned the Page Rank scores according to the algorithm, a detailed explanation for which was already provided. This score signifies the influence of that particular user. The user who has a high Page Rank score has a higher influence compared to a user who’s Page Rank score is less.

Proof of correctness can be obtained by testing our model to achieve expected results. If a user with a low score is connected to a user with a high score, then their score improves. This can give us the correctness of our model if it can follow the same. For this, a random user who’s Page Rank score is low is chosen. Edge connections are formed between this user and other users whose score is high. The users are selected from the same cluster, such that they are similar. This can be made by changing the adjacency matrix to reflect the edge connection i.e. the values 0 are changed to 1 in the adjacency matrix to signify an edge between those users. After this change has been made, we now have a new adjacency matrix for that cluster. Next, we recompute the Page Rank scores for all the users with the updated adjacency matrix.

We can observe that there will be a change in the scores and the ranks of the users. The initially selected users with low scores will now have an increased score and rank. As we can observe, this can be attributed to their connection with the influential users. However, they are not the most influential users because of other factors like teleportation vector.

The results of this test show that a user with low score can have an increased influence in the underlying graph if there is an edge formed between them and an influential user. The increased influence would contribute to an increased dissemination of information through the users as a result. Our approach follows the similar methodology of identifying the most influential node, who is similar to our given user, to form an edge between the user and the influential similar user. This ensures that the initial propagator gains influence and that information is disseminated consequently. Relevant information dissemination is achieved, as the edges are formed with influential users in the same cluster

**4. EXPERIMENTATION AND VISUALIZATION**

**4.11 TESTING FOR A SINGLE USER**

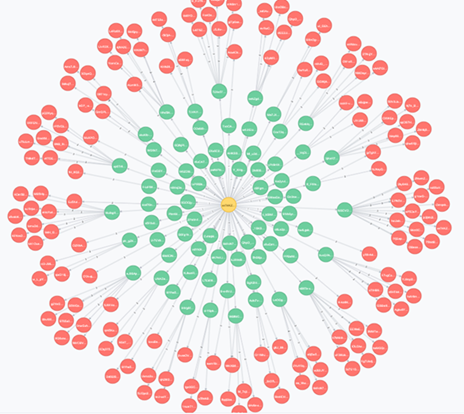
**(1 and 1/2 page with diagram)**

**4.12 TESTING FOR MULTIPLE USERS**

**(1 and ½ page with diagram)**

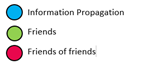
**4.13 INFORMATION DISSEMINATION**

After obtaining the results from each algorithm, we collected the reviews for each restaurant audited by the influential user and using the hit ratio we visualized the information dissemination based on the friends of the influential user posting their own reviews about the same place.

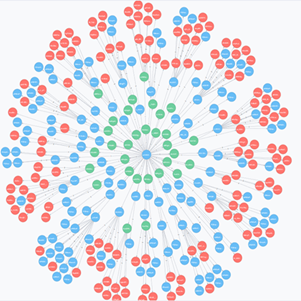




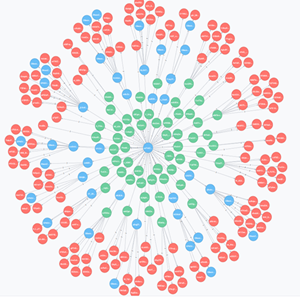
By choosing the highest influential user, we used Neo4j to visualize information dissemination to their friends. Due to higher similarity tastes in K-means algorithm there is more propagation observed compared to Twitter-rank algorithm. This is shown using the data from the K-means and Twitter-rank user data.



**K-means algorithm**



**Twitter-rank algorithm**



**5. EMPIRICAL EVALUATION**

**(tables – 2 page)**

**6 FUTURE WORK**

**(3/4 page)**

Doc2Vec model can be an approach that could be used to substitute LDA in the future. Doc2Vec is an unsupervised algorithm to generate vectors for sentence/paragraphs/documents. It is based on existing Word2Vec models. Simple analogy questions can be answered using Word2Vec models. For instance, “King” - “man” + “woman” = “Queen” (Mikolov et al., 2013d). The advantage of Doc2Vec is that it can work with unlabeled data.

In our proposed approach for using Doc2Vec, each user’s reviews are collated together as a single document. There are as many documents as the number of users. The documents are used to train the Doc2Vec model, along with the User IDs as labels. We utilized gensim library to build our Doc2Vec model. Using this model, we obtain the vector representation for each user.



The above figure clearly represents the outline for our proposed approach with Doc2Vec models. As we discussed, the model generates vectors for each user. In our case, we have the number of dimensions for the vectors set to 100.

The resulting user vectors are then used to cluster the users. The proposed approach uses k-means clustering to find similar users. The optimal number of clusters is identified using the elbow method wherein, the number of clusters showing a sharp reduce while increasing the cluster number from 1 to k followed by very less decrease in error is chosen.

The clustering results are then used to build the adjacency matrices for each cluster. Then, to propagate information from one user to another in the same cluster, we identify the most influential node in that cluster. This is obtained using Page Rank algorithm on the derived adjacency matrices. The information is then propagated through the selected influential user. The marginal error with Doc2Vec is lesser when compared to LDA, hence it would be a better alternative to use instead of Doc2Vec.

Other such proposed alternatives would be to use Gaussian Mixture Model (GMM) instead of using k-means, since it starts with a prior. Also, Probabilistic Latent Semantic Indexing could be used as an alternative to LDA. It is the maximum a posteriori equivalent of LDA that takes into consideration the co-occurrence of words.

**7. CONCLUSION**

The model is able to effectively disseminate information to the relevant user, which results in a greater reach. The model is able to achieve this relevant information dissemination since it uses a similarity score identify the right user to propagate information through. It is because of this reason, the model has an advantage in information dissemination compared to the regular approach of using Twitter Rank algorithm.

The model does have some downsides when it comes to complexity. The model utilizes k-means clustering which causes us this complexity overhead. The original proposed Twitter Rank algorithm is able to obviate this problem since it does not cluster the users based on similarity. Rather, it directly approaches the problem with Topic Sensitive Page Rank applied over topic modeling with LDA. The usage of k-means on the other hand is a costlier process while fetching us good results. This creates a tradeoff in our model between computing and effective information dissemination, which is another downside. The model has to compromise on either computing or effective information dissemination. When the computing resources have to be reduced, the model will not yield relevant information propagation as with other models. In case of compromise on computing, the model can yield the expected results.

**7. REFERENCE**

**(1 page)**

1. **WENG, Jianshu; LIM, Ee Peng; JIANG, Jing; and HE, Qi. Twitterrank: Finding Topic-Sensitive Influential Twitterers. (2010). Proceedings of the Third ACM International Conference on Web Search & Data Mining: February 3-6, 2010, New York. 261-270. Research Collection School Of Information Systems**
2. **L. Page, S. Brin, R. Motwani, and T. Winograd, “The pagerank citation ranking: Bringing order to the web.” Stanford InfoLab, Technical Report 1999-66, November 1999, previous number = SIDL-WP-1999-0120.**

1. **T. Haveliwala, “Topic-sensitive pagerank: a context-sensitive ranking algorithm for web search,” Knowledge and Data Engineering, IEEE Transactions on, vol. 15, no. 4, pp. 784 – 796, july-aug. 2003**

1. **David M Blei, “Probabilistic topic models,” Communications of the ACM, vol. 55, no. 4, pp. 77–84, 2012.**
2. **Thomas Hofmann, “Probabilistic latent semantic indexing,” in Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 1999, pp. 50–57.**

1. **David M Blei, Andrew Y Ng, and Michael I Jordan, “Latent dirichlet allocation,” the Journal of machine Learning research, vol. 3, pp. 993–1022, 2003.**

1. **K. Vorontsov, A. Potapenko, EM-algorithm modification for probabilistic topic modeling // Machine learning and data analysis, 2013, V. 1, № 6, pp. 657–686**
2. **"Yelp Dataset Challenge - Yelp". *Yelp*. N.p., 2017. Web. 27 Apr. 2017.**
3. **"Yelp/Dataset-Examples". *GitHub*. N.p., 2017. Web. 27 Apr. 2017.**