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

Akhil Ravichandran
Arizona State University
Tempe, AZ 85281
aravic12@asu.edu

Gurumurthy Raghuraman
Arizona State University
Tempe, AZ 85281
graghura@asu.edu

Narendra Kumar
Arizona State University
Tempe, AZ 85281
Nsampat1@asu.edu

Subramanian
Arizona State University
Tempe, AZ 85281
svenka71@asu.edu

Ravikiran
Arizona State University
Tempe, AZ 85281
rtangir3@asu.edu

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.

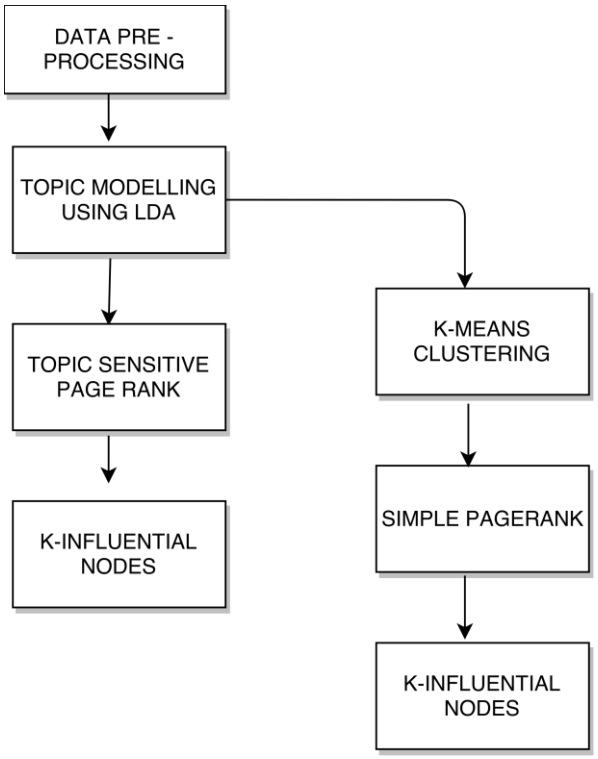


Figure.1.a Existing methodology and Our Approach

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.

yelp_academic_dataset_business.json

```
{
    "business_id":"encrypted business id",
    "name":"business name",
    "neighborhood":"hood name",
    "address":"full address",
    "city":"city",
    "state":"state -- if applicable --",
    "postal code":"postal code",
    "latitude":latitude,
    "longitude":longitude,
    "stars":star rating, rounded to half-stars,
    "review_count":number of reviews,
    "is_open":0/1 (closed/open),
    "attributes":["an array of strings: each array element is an attribute"],
    "categories":["an array of strings of business categories"],
    "hours":["an array of strings of business hours"],
    "type": "business"
}
```

yelp_academic_dataset_review.json

```
{
    "review_id":"encrypted review id",
    "user_id":"encrypted user id",
    "business_id":"encrypted business id",
    "stars":star rating, rounded to half-stars,
    "date":"date formatted like 2009-12-19",
    "text":"review text",
    "useful":number of useful votes received,
    "funny":number of funny votes received,
    "cool": number of cool review votes received,
    "type": "review"
}
```

yelp_academic_dataset_user.json

```
{
    "user_id":"encrypted user id",
    "name":"first name",
    "review_count":number of reviews,
    "yelping_since": date formatted like "2009-12-19",
    "friends":["an array of encrypted ids of friends"],
    "useful":"number of useful votes sent by the user",
    "funny":"number of funny votes sent by the user",
    "cool":"number of cool votes sent by the user",
    "fans":"number of fans the user has",
    "elite":["an array of years the user was elite"],
    "average_stars":floating point average like 4.31,
    "compliment_hot":number of hot compliments received by the user,
    "compliment_more":number of more compliments received by the user,
    "compliment_profile": number of profile compliments received by the user,
    "compliment_cute": number of cute compliments received by the user,
    "compliment_list": number of list compliments received by the user,
    "compliment_note": number of note compliments received by the user,
    "compliment_plain": number of plain compliments received by the user,
    "compliment_cool": number of cool compliments received by the user,
    "compliment_funny": number of funny compliments received by the user,
    "compliment_writer": number of writer compliments received by the user,
    "compliment_photos": number of photo compliments received by the user,
    "type":"user"
}
```

yelp_academic_dataset_tip.json

```
{
    "text":"text of the tip",
    "date":"date formatted like 2009-12-19",
    "likes":compliment count,
    "business_id":"encrypted business id",
    "user_id":"encrypted user id",
    "type":"tip"
}
```

yelp_academic_dataset_checkin.json

```
{
    "time":["an array of check ins with the format day-hour:number of check ins from hour to hour+1"],
    "business_id":"encrypted business id",
    "type":"checkin"
}
```

photos (from the photos auxiliary file)

This file is formatted as a JSON list of objects.

```
[
    {
        "photo_id": (encrypted photo id),
        "business_id" : (encrypted business id),
        "caption" : (the photo caption, if any),
        "label" : (the category the photo belongs to, if any)
    },
    {...}
]
```

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

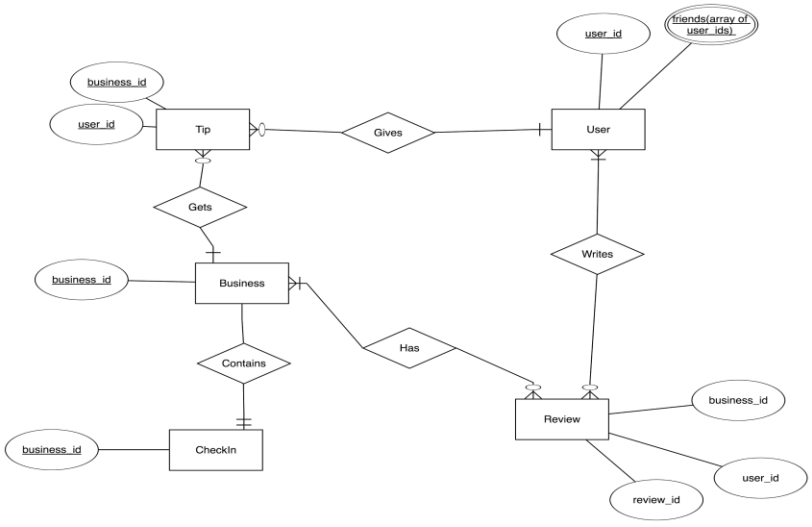


Figure 2.1.1.a Entity Relationship Diagram

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.

☑	☐	review_id	varchar	☐	
☐	☐	user_id	varchar	☑	NULL
☐	☐	business_id	varchar	☑	NULL
☐	☐	review_text	varchar	☑	NULL
☐	☐	business_name	varchar	☑	NULL
☐	☐	business_categories	varchar	☑	NULL
☐	☐	business_state	varchar	☑	NULL
☐	☐	business_city	varchar	☑	NULL
☐	☐	business_attributes	varchar	☑	NULL
☐	☐	business_review_count	integer	☑	NULL
☐	☐	business_stars	float	☑	NULL
☐	☐	user_name	varchar	☑	NULL
☐	☐	user_review_count	integer	☑	NULL
☐	☐	user_average_stars	float	☑	NULL
☐	☐	user_fans	integer	☑	NULL
☐	☐	review_stars	integer	☑	NULL
☐	☐	review_date	date	☑	NULL
☐	☐	review_useful	integer	☑	NULL
☐	☐	review_cool	integer	☑	NULL
☐	☐	review_funny	integer	☑	NULL
☐	☐	user_useful	integer	☑	NULL
☐	☐	user_yelping_since	date	☑	NULL
☐	☐	user_cool	integer	☑	NULL
☐	☐	user_funny	integer	☑	NULL
☐	☐	business_address	varchar	☑	NULL
☐	☐	business_postal_code	varchar	☑	NULL
☐	☐	review_types	varchar	☑	NULL
☐	☐	user_friends_count	integer	☑	NULL
☐	☐	user_friends	varchar	☑	NULL

Figure 2.1.1.b Database Schema

2.12 CONSTRAINTS IN DATA PREPROCESSING

With the constructed table, Reviews_Business_Users we had an additional step to prune the dataset with respect to the Users considered. We initially deemed friends_count to be a good estimate to determine strong nodes in our network to determine information dissemination. But social network signups prompt users to become friends with existing friends as a courtesy this gives us a skewed result in regard to strongest nodes in a network. Hence, we needed another parameter to determine the users to be filtered.

We decided to use attributes such as the user_average_star, user_fans and our pre-calculated user_review_count as we determined it to be the required information to deem if a user is a strong contributor of reviews in the Yelp network.

Additionally we put certain constraints on these attributes such as an user having an average star rating above 4.0, a user having a fan following of 50 and a user review count of greater than 5.We used these parameters to prune the users being considered and considered the reviews posted by such users only.

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.

For example: A word belongs to topic t1 and t3

$W \in t1, t3$

$I_{t_1} = i \ , I_{t_3} = j$

If $j > i$ then $W \in t3$, increment the count of t3

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
User1	15	25	13	19	22	16	12	23	25	26
User2	10	8	15	17	11	15	21	24	16	18
User3	12	9	14	15	8	16	23	21	11	12
User4	8	16	23	21	10	12	17	11	21	24
User5	6	8	14	12	11	14	17	19	22	25
User6	8	9	11	17	12	11	9	8	17	14
User7	11	8	15	14	18	15	21	21	17	19
User8	15	12	17	19	22	6	8	9	12	15
User9	19	13	16	14	12	11	22	8	7	4

Figure 2.2.2.a User-Topic Matrix

Based on this method, we form the User-Topic matrix. A sample is shown in figure 2.2.2.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 s_i to $s_j$is defined as

$$P_t(i,j) = \frac{|R_J|}{\sum_{a:\, s_i\ friends\ with\ s_a} |R_a|} * sim_t(i,j)$$

$$sim_t(i,j) = 1 - |DT'_{it} - DT'_{jt}|$$

$|R_J|$ - Number of Reviews posted by s_j

$\sum_{a:s_i\ follows\ s_a} |R_J|$ - Sums up the number of Reviews published by all s_i'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

We formulate 't' different graphs to create the topic specific connections between the users. We create 10 adjacency matrices for 10 topics by using the above formulas. These adjacency matrices represent the weight linkage of a user to his friends based on his topic relevance posts on Yelp.

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 rank the websites. It is a random walk model. Consider a node containing several edges, we take any one route and follow one of the edges randomly, then the probability of taking that path is given by 1/n where n is the number of outgoing edges.

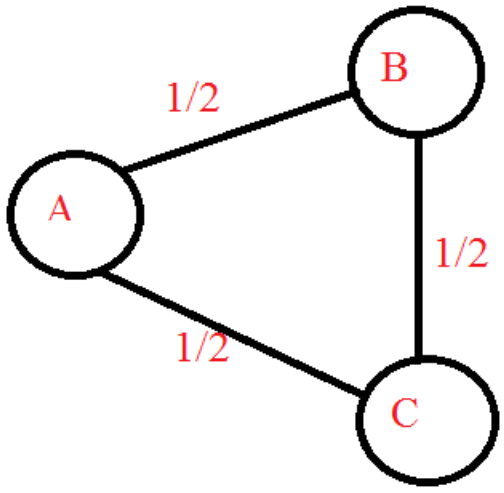


Fig 2.2.4.1.A PageRank Illustration

In Fig 2.2.4.1.A we see that the probability of moving from A to B is 1/2. Similarly, the probability of moving from A to C is also ½. 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 continues the random walk or the user again starts the random walk from different position. We have taken the damping factor as 0.85 which is most widely used. The probability of continuing the random walk is 0.85 and the probability of restarting the random walk is 0.15. The algorithm continues to run until it converges. We get the PageRank scores of all the nodes once it converges. 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. It is one of the most significant nodes in the entire graph.

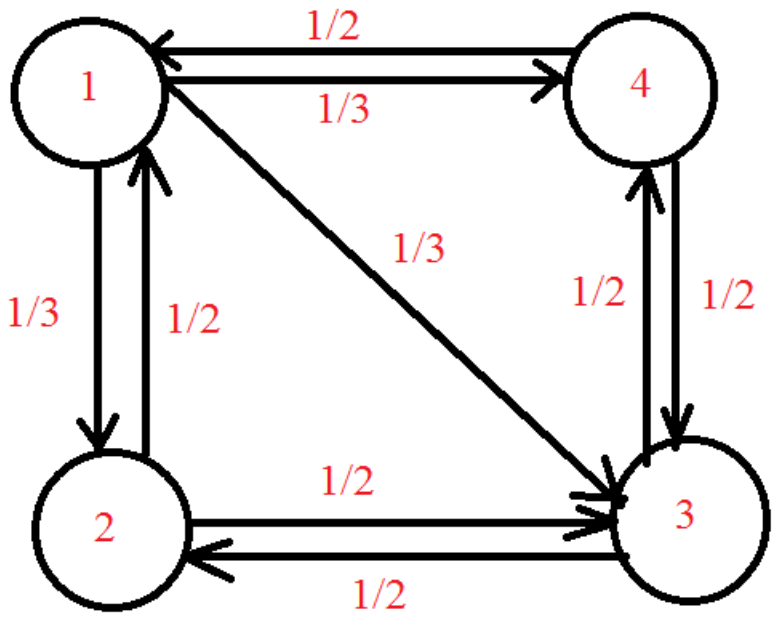


Fig 2.2.4.1.B PageRank Example.

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

Nodes	1	2	3	4
1	0	½	0	1/2
2	1/3	0	½	0
3	1/3	½	0	1/2
4	1/3	0	½	0

Fig 2.2.4.1.C Transition Table

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 algorithm is explained in detail

Algorithm:

1. Let the transition matrix be A.
2. Let the Teleportation vector be B. Initially all the nodes have equal teleportation probability. Thus B is

B=

0.25
0.25
0.25
0.25

3. Initialize oldB to a matrix of size B all values initialized to 0.

4. *while (B !=oldB)*
   OldB = B;
   *B = A*B*
5. *Output B*

One of the key problems in the above matrix is that for an unconnected graph some of the nodes will never be reached. In order to solve this problem, we introduce the damping factor. The equation for PageRank can be re written as

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

P represents the Damping factor

T is the Teleportation vector

2.2.4.2 TWITTERRANK

TwitterRank is like PageRank but there is a change in the teleportation vector. The equation for TwitterRank is as follows:

$$TR_t = (alpha) * P_t + (1 - alpha) * E_t$$

E_t is the teleportation vector.

$E_t = DT'$, where DT' is a column normalized matrix of DT.

If a user has a large contribution in a specific topic, there are higher probability of being teleported to that user in the random walk.

2.2.4.3 AGGREGATION OF TOPIC SPECIFIC PAGERANK

We get the topic specific PageRank scores from the above method. In our case, we have ten different PageRank scores. For a given test user (a reviewer), we find the contribution of his interest for the list of topics. We do so by taking the bag of words of that user, then we find the contribution of the user to topic t i.e. number of intersecting words of the user with the LDA applied words, obtained from the overall corpus, for that topic t.

$$C_t = (User\ contribution\ to\ topic\ t)/(topic\ number\ of\ words\ in\ t)$$

The obtained C_t is multiplied to the topic specific PageRank score.

$$TR = \sum C_t\ TR$$

After getting the Topic specific PageRank Score, we add together all the PageRank scores. The user with the maximum score is the most influential user.

2.3 K-MEANS PAGE RANK

We propose this new methodology K-means PageRank as the influential users in the previous

approach mainly depends on the number of connections that he/she has. The topic distribution of the influential user is not similar to the topic distribution of the reviewer. In other words, he is not a likeminded influential user. Also, the top influential user remained the same for most of the random reviewers that we experimented. It was due to the edge connections he/she has in the graph. In order to give a considerable weightage to reviewer's topic interest, we propose this clustering based approach.

In our approach, we apply k means clustering after forming the User-Topic 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 *quite atmosphere* then such users will be in the same cluster. Users who may have multiple similar interest will be clustered together. One of the key issues of k means clustering is that we must 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 No of clusters 'K' vs the objective function where the objective function is defined as

$$Objective\ Function = \sum i = 1\ to\ N\ |u_j - x_i|^2$$

N = Number of Feature vectors of the user

u_j – Cluster centroid

we plot a graph of k vs the objective function.

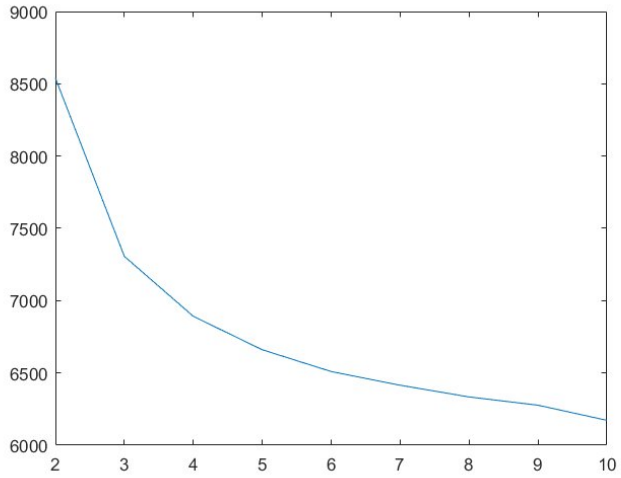


Fig 2.3.A K value vs Objective Function

The optimal value of k is chosen as 5 as it tapers down drastically till K = 5.

2.3.1 GRAPH FORMATION FOR K-MEANS

We have already clustered the users with similar interest. Now for a test user or reviewer, we find the closest cluster to that user based on his/her feature vector and the centroid of each cluster. We used Euclidean distance to compute the distance. Once we find out the right cluster for a reviewer. We form a graph with the users who belong to that cluster and the edges of the graph are only within the 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.

2.3.2 FINDING MOST INFLUENTIAL USER

After obtaining the Adjacency matrix with the users belonging to the cluster where the test user falls in. We apply simple PageRank algorithm with weights as 1 if there exist an edge and 0 if there isn't. The PageRank converges with the scores for every user. The top K users are the most influential and likeminded users for the given test user.

3. RESULTS

From the K-means page-rank and TwitterRank algorithms, we get the topic vectors using the reviews of the users in k-means and using the number of incoming edges in twitter-rank. These are used to obtain the most influential and most active users respectively.

Below 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.

Topic	TR Influential Users	Active Users
pizza, bread, vegan, soup, potato	nXMMoDzXRByQiWH9cf-FAQ, Wc5L6iuvSNF5WGBlqIO8nw, BA9SDRZGyRcUMxTdUTVhmw, PcPminF0blUsKjUh9e4DMQ, 8DGFWco9VeBAxjqsuh1aSw	vxR_YV0atFxlxfOnF9uHjQ, H5d_nFqzwrREE-YduK2ABg, eAJ3Ip0vh-zuk_BmcmchYA, el3TmKFEFzZOcNbCw2FNlQ, Bsbumyr3Of8BauN2URK4bg
chicken cheese salad fresh lunch	nXMMoDzXRByQiWH9cf-FAQ, BA9SDRZGyRcUMxTdUTVhmw, Wc5L6iuvSNF5WGBlqIO8nw, PcPminF0blUsKjUh9e4DMQ, 8DGFWco9VeBAxjqsuh1aSw	GABWrw5Et9jubriKwMUDbw, Azkt7v-m9iBLjMZtCMxorg, s5F0vildrelfeDtcMM65PQ, 8v3aEuuHEJU_8h0OgmsfDA, W83aZrDrYKwycCHB_eOYYg
sweet dessert night delicaci chocolate	nXMMoDzXRByQiWH9cf-FAQ, Wc5L6iuvSNF5WGBlqIO8nw, BA9SDRZGyRcUMxTdUTVhmw, PcPminF0blUsKjUh9e4DMQ, 8DGFWco9VeBAxjqsuh1aSw	H5d_nFqzwrREE-YduK2ABg, el3TmKFEFzZOcNbCw2FNlQ, HmlLc-IgVKZS7mWNdYCSJw, 9v83UksZrkThOiE584GRBQ, Bsbumyr3Of8BauN2URK4bg
day tabl side,dish top night happi meal	nXMMoDzXRByQiWH9cf-FAQ, BA9SDRZGyRcUMxTdUTVhmw, Wc5L6iuvSNF5WGBlqIO8nw, PcPminF0blUsKjUh9e4DMQ, 8DGFWco9VeBAxjqsuh1aSw	T5KBc5QbwZ-Oj9ApE4vZJA, s5F0vildrelfeDtcMM65PQ, coKnuxLznH0FHb34m4ZLCw, vxR_YV0atFxlxfOnF9uHjQ, HVJgTH5qu0goywOHNpOjPA
drink bar restaurant friday friend	nXMMoDzXRByQiWH9cf-FAQ, BA9SDRZGyRcUMxTdUTVhmw, Wc5L6iuvSNF5WGBlqIO8nw, PcPminF0blUsKjUh9e4DMQ, 8DGFWco9VeBAxjqsuh1aSw	fZY97UjlP-iv35beOl_OfQ, nffEoRhzQYVIL3Y6bFhcGA, mjOKUSdlqxFPSYMB-MmS9g, tCbEkK11lnqeaNVkhlurrQ, TwjiWvK0ZP15lzGC049oIQ

Fig 3.A Active Users vs Influential User

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

Test Users (5 different Users)	TR PageRank Most Influential User	Our Approach (K-means PR)
dessert, star,visual, salad, quiet, atmosphere	sandwich, tea, coffee, salad, bread, fruit, dessert	atmosphere, star,gaze, view,perfect, milk,make
pizza,beef, pie, italian	sandwich, tea, coffee, salad, bread, fruit, dessert	sauc, burger, fast, chicken, time
coffee, price, area, locat	sandwich, tea, coffee, salad, bread, fruit, dessert	sober,trust,scarf,best, definit
salad, delici, food	sandwich, tea, coffee, salad, bread, fruit, dessert	restaurant, yummi, soup , hot
burger,cake,food,sweet, good,buffet	sandwich, tea, coffee, salad, bread, fruit, dessert	pizza, caramel, smoothie

Fig 3.B 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.

Below, we compare the bag of words obtained from approach 1 and approach 2 for a random test user

Test user's Topic words:

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

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

We compare the results of Topic sensitive PR and K-means PR. we see that the test user's bag of words and the influential user's bag of words match on food tastes like dessert, salad (marked in red) whereas using our approach (K-means PR) the bag of words match on finer details like the atmosphere, ambience, food items. The blue words signifies the words that matched with a specific topic in the overall corpus for test user and the most influential user. The words green signifies the exact matches.

3.11 SIMILARITY MEASURES

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. We use this measure because it helps to find the distance between a random user in a cluster and the (centroid) influential user. 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]. This helps to identify the intersections between the words in a topic of a random user and the influential user. 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

Dice coefficient, also known by other names is a statistic used for comparing the similarity of two samples [12]. This helps to compare the words in a topic of a random user and the influential user. 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

What is a 'Hit'? When a reviewer posts a review, and the information is spread to the most influential user. If the Influential person accepts the information (say, he likes the post, comments it, or even visits the same restaurant) it is a Hit. If the user ignores the post, it is a Miss. 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

3.22 PROOF AND ANALYSIS

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 user. The user who has a high Page Rank score has a higher influence compared to a user whose 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 whose 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 recomputed 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.

User	Original Rank	Rank Score	Updated Rank	Updated Rank Score
1029	1042	0.000027	193	0.001012
959	1052	0.000003	93	0.002421
1018	1045	0.000025	70	0.00322
1013	1048	0.000016	147	0.001495
1007	1047	0.000023	78	0.002922
98	1049	0.000014	15	0.009701
982	1043	0.00025	63	0.00377

Fig. 3.2.2.A PageRank score increases when connected to likeminded influential user

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

We simulated the user behavior based on the similarity scores obtained. We visualized the most influential user based on a test user.

4.1 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.

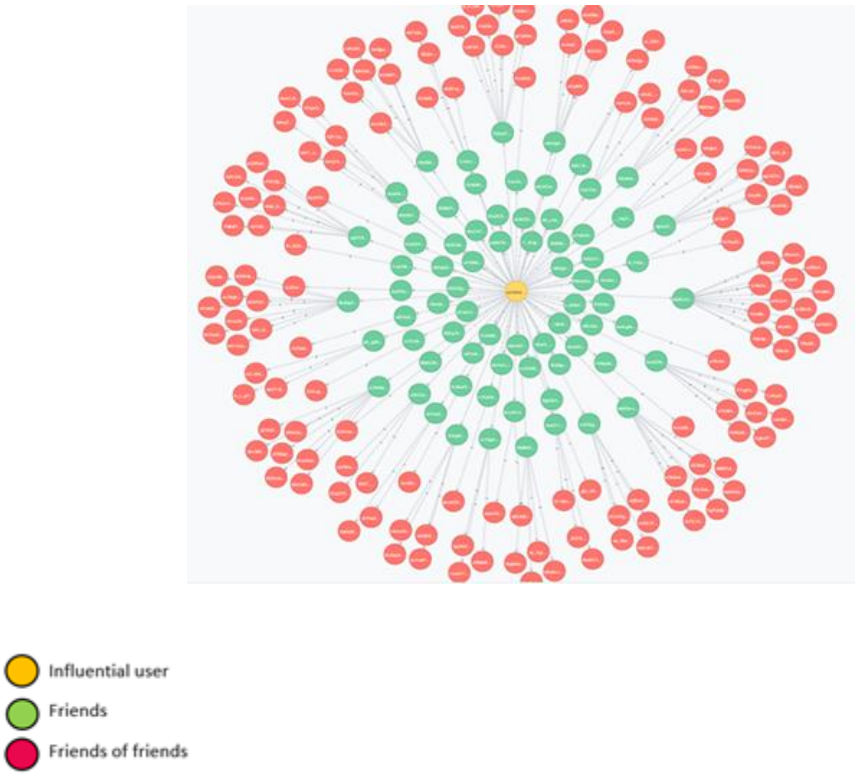


Fig 4.1.A Visualizing the most influential user and his network

K-means algorithm

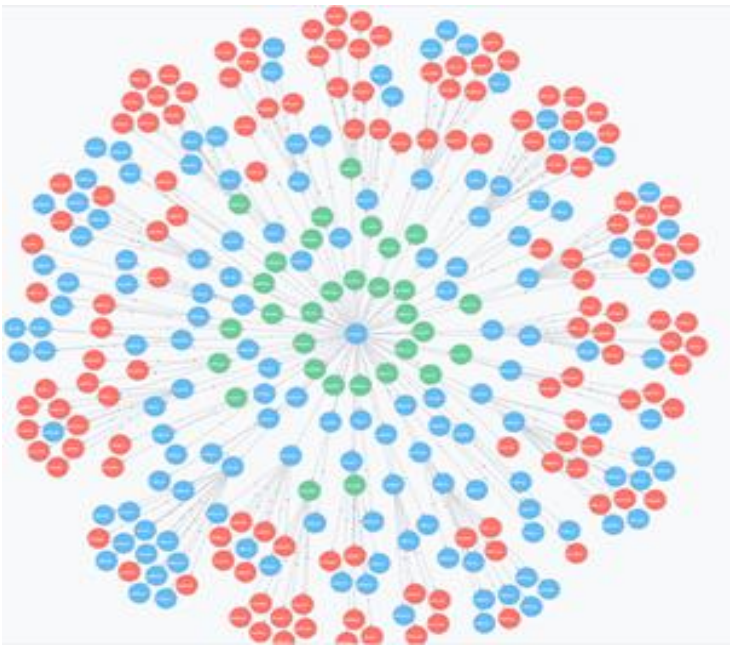


Fig. 4.1.B Information propagation for most influential in K-means

Twitter-rank algorithm

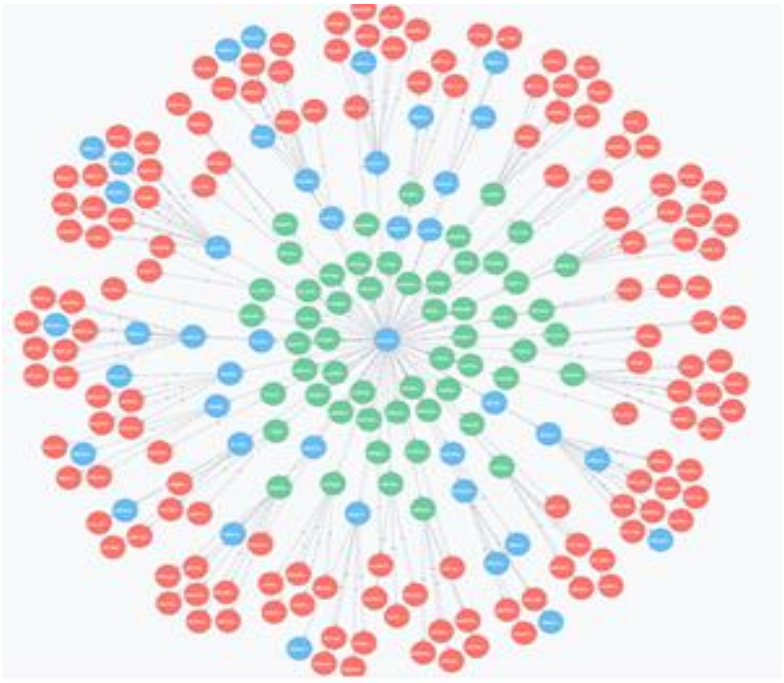
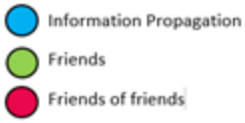


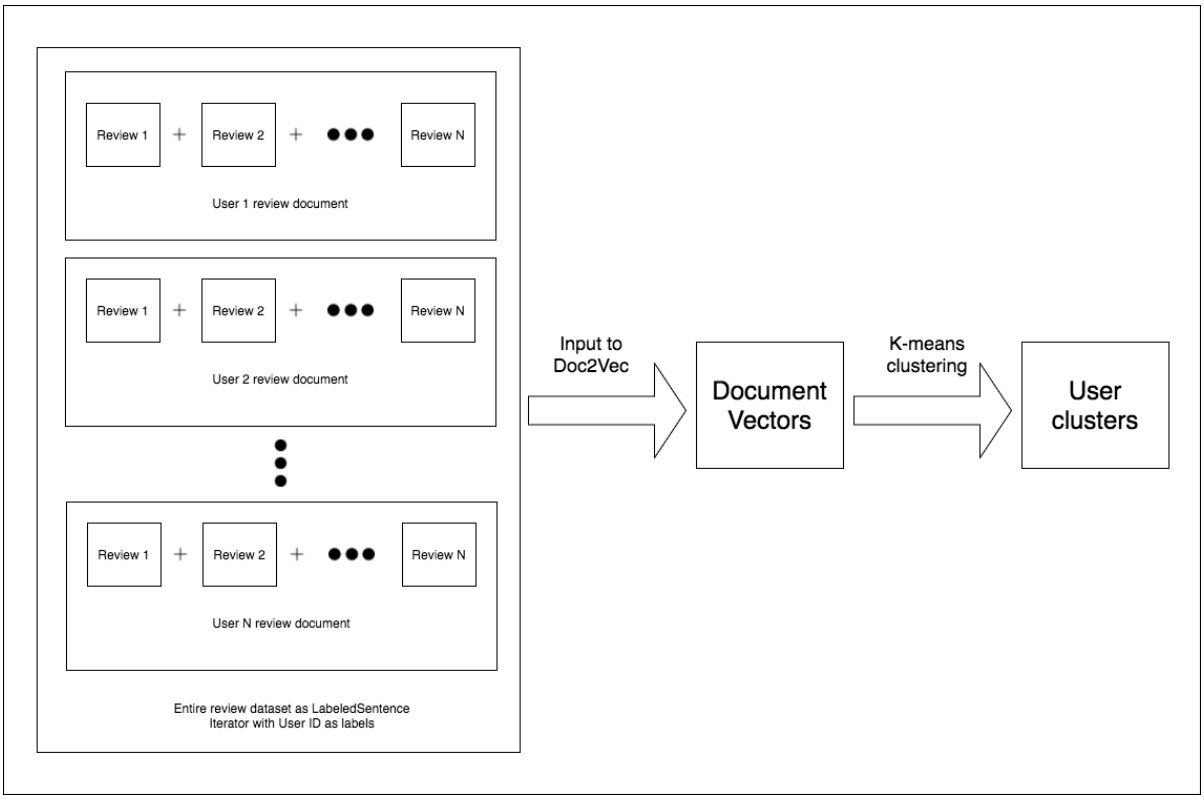
Fig. 4.1.C Information propagation for most influential in TwitterRank



5. FUTURE WORK

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) [13]. 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 likeminded influential user, which results in a greater reach. The users are clustered based on their interest. 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. **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.**

3. **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**

4. **David M Blei, "Probabilistic topic models," Communications of the ACM, vol. 55, no. 4, pp. 77–84, 2012.**

5. **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.**

6. **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.**

7. **K. Vorontsov, A. Potapenko, EM-algorithm modification for probabilistic topic modeling // Machine learning and data analysis, 2013, V. 1, № 6, pp. 657–686**

8. **"Yelp Dataset Challenge - Yelp". *Yelp*. N.p., 2017. Web. 27 Apr. 2017.**

9. **"Yelp/Dataset-Examples". *GitHub*. N.p., 2017. Web. 27 Apr. 2017.**
10. **https://en.wikipedia.org/wiki/Euclidean_distance**

11. **https://en.wikipedia.org/wiki/Jaccard_index**

12. **https://en.wikipedia.org/wiki/S%C3%B8rensen%E2%80%93Dice_coefficient**

13. **Quoc Le and Tomas Mikolov. Distributed Representations of Sentences and Documents. http://arxiv.org/pdf/1405.4053v2.pdf**