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

**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 user particularly describes about how many users are a fan of the reviewer. This helps in identifying whether the user is a fan of the reviewer’s reviews or are socially circled friends from other applications like Facebook, twitter etc. We consider only for the foods and restaurant reviews to minimize the complexity. 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 Influential people. Currently, most of the research papers are focusing on the influence of the user based on the number of friends that 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 timeline. In that case, it’s a direct miss. We propose a new methodology where the Information will spread to the likeminded Influential user. This methodology has the multifarious benefits, it potentially 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 an Information, it is likely that his friend’s circle would be of similar taste to the user, which then increases the hit ratio in the consecutive propagation. Thirdly, it is also likely that the most influential user for a particular topic say, “Indian food” would be socially connected to the Influential user who reviews about every food cuisine, so the Information is given to a proper set of people 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.

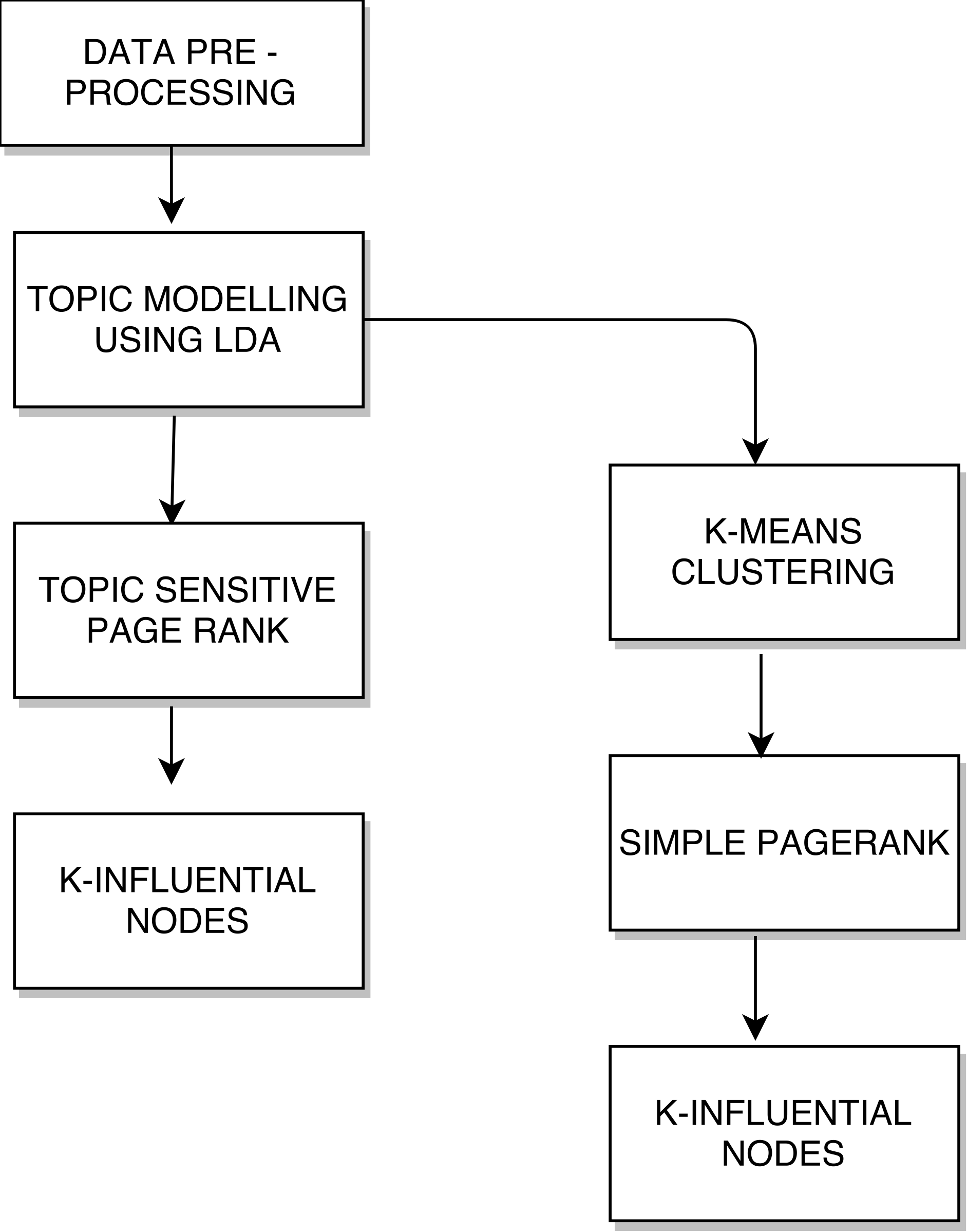


Figure.1 Existing methodology and Our Approach

We obtain the dataset from Yelp dataset challenge. We apply certain constraint and preprocess the data. We took the entire collection of reviews written by all the users in the dataset and form a huge corpus and obtained the set of topics (10 Hidden topics) using Latent Dirichlet Allocation and a bag of 200 words for every topic. The process ran for 500 iteration 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 form a matrix with users on the rows and topics as the columns, with number of user’s words fall into the topic bag of words. This lists down the feature vector for every user. We apply K-means clustering to the user’s feature vector to obtain the cluster of users who are likeminded. Then we apply Page Rank algorithm to find out the most influential likeminded user. We also various similarity measures between the reviewer and the influential user obtained from the Twitter rank algorithm and our K-means PR (Page rank) algorithm and prove that our algorithm provides the increased hit ratio which in turn produces increased information dissipation in a large graph database.

**1.1 RELATED WORKS**

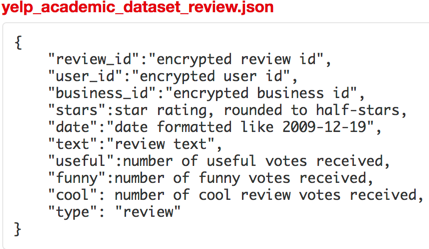
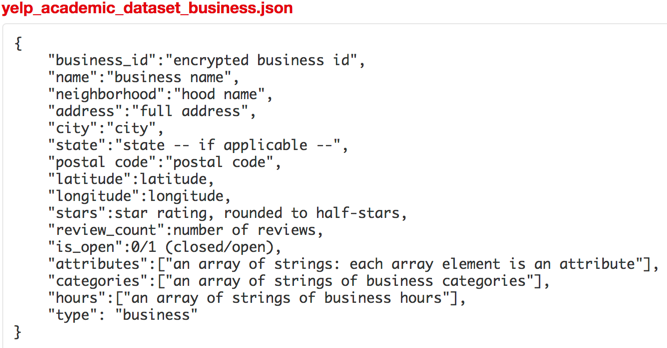
PageRank is the core concept of modern search engine. In the PageRank [3], the rank of a page is equivalent to the probability that a random surfer reaches this page. The random surfer choses a page with a probability P, then at each step, it choses another page to visit with probability Por uniformly 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.

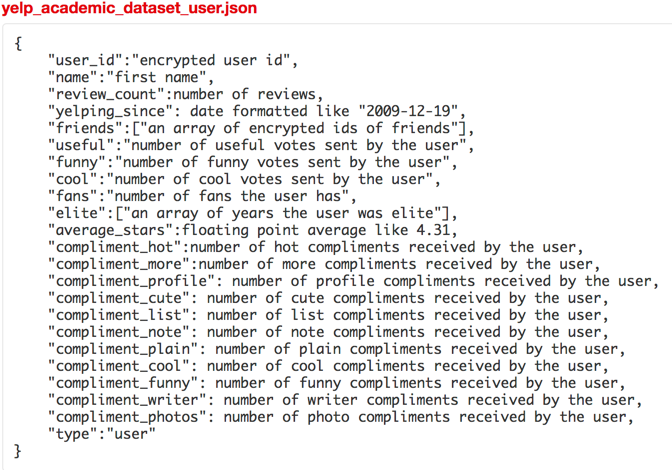
Latent Dirichlet Allocation (LDA) is a generative statistical model that allows set of observed topics to be explained by unobserved topics. A document usually contains series of topics which can be described by specific frequency or probability distribution. Several hidden topics is assumed to be present 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.

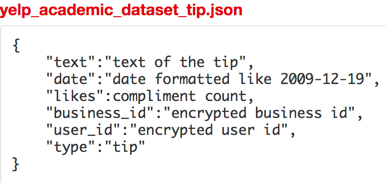
Jianshu et all. came up with a new algorithm called TwitterRank [9] which focuses on the problem of identifying the influential users of micro blogging services like Twitter. The twitter currently uses PageRank and Topic sensitive PageRank. The TwitterRank algorithm, an extension of PageRank algorithm 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.

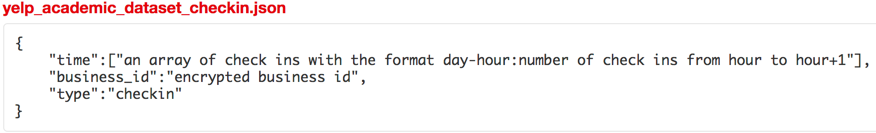
**2.1 DATASET - Akhil**

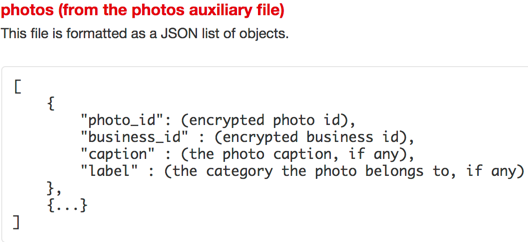
We have taken the masked dataset from Yelp. Yelp provides their academic dataset for the Yelp DataSet Challenge[1].The data contains several tables with UserId, User Name,User Location,BusinessId,Business Name,Business Reviews and other required data.The sensitive data like UserId , Username is masked 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 in regards to Users and Business .The Tables and the columns in the data set are listed below in json format[1][2].We have considered some of the tables from the provided dataset, they include : Users,Business,Review and Tips.



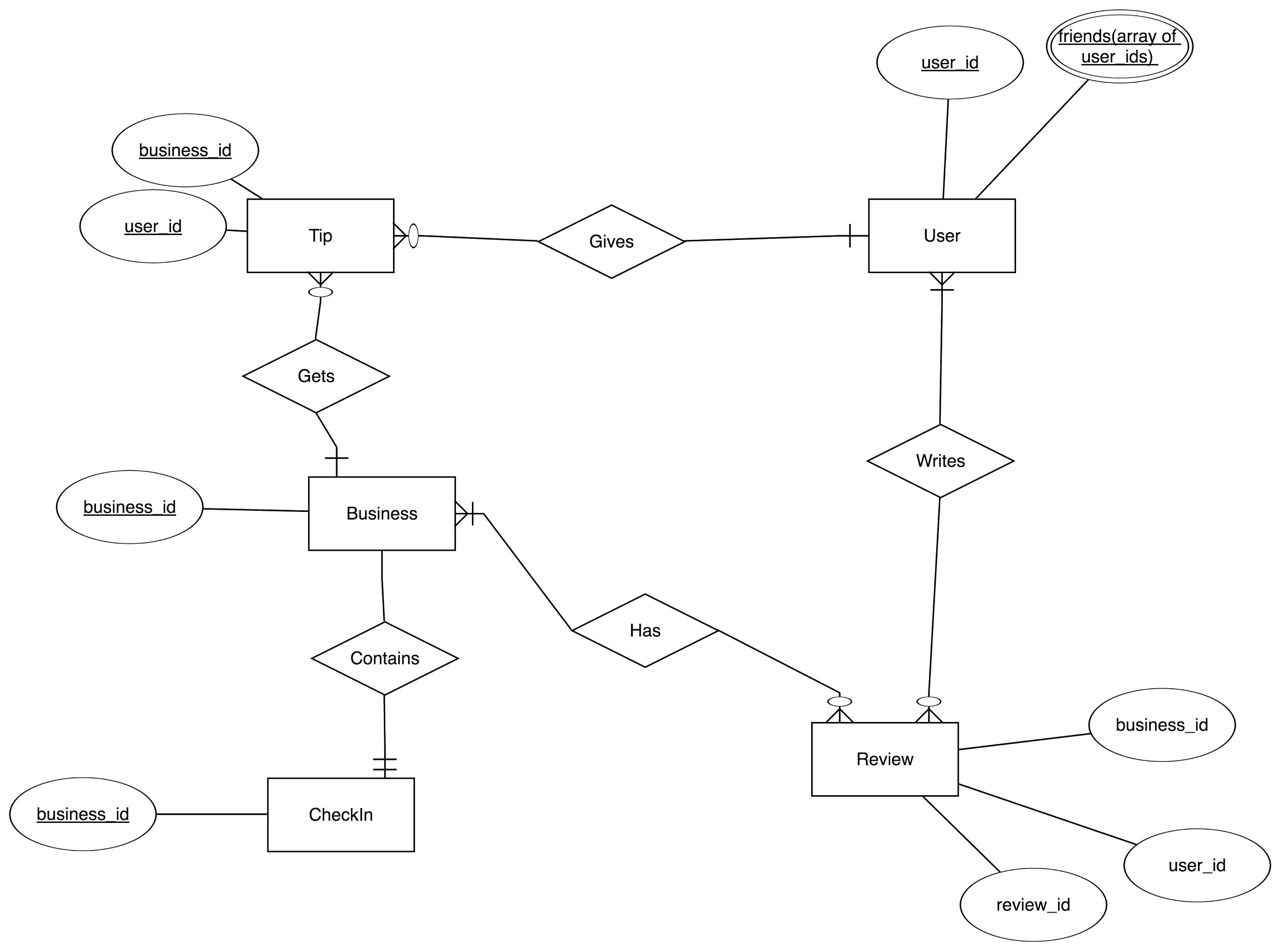








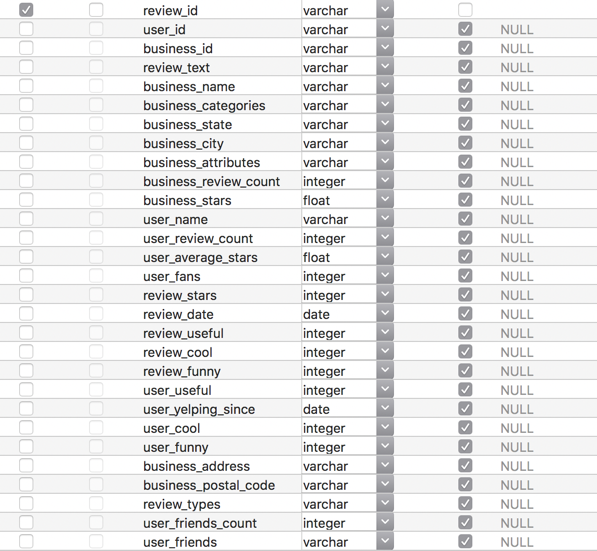
**2.11 Dataset preprocessing - Akhil**



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[2].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 this is the key requirement for our dataset.As 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 regards to its location , business type,attributes,categories and other parameters to be used as label data to test our clustering techniques in regards 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) below from our database.



**2.12 Constraint - Akhil**

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

**2.2 TWITTER RANK**

Intuitively, the influence of twitterer 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 twitter rank algorithm using Yelp dataset.

**2.21 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 associating 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.

**2.21 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 fall 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 2.21.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

**2.22 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 (NxN) 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 topic 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.

**2.22 TOPIC SENSITIVE PAGE RANK**

**2.3 K-MEANS PAGE RANK**

**(2 page)**

**3. PROOF AND ANALYSIS**

**(1/2 page intro)**

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

**3.21 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 are shown below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 |

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**

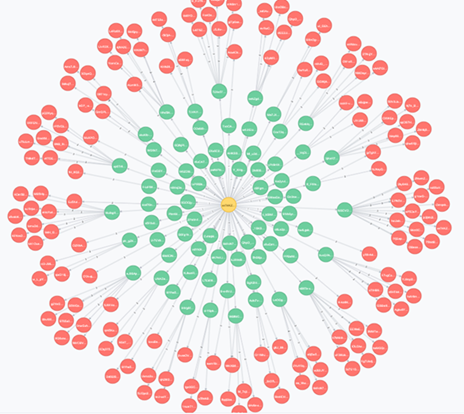
**(1 and 1/2 page with diagram)**

**4.12 TESTING FOR MULTIPLE USERS**

**(1 and ½ page with diagram)**

**4.13 INFORMATION DISSEMINATION**

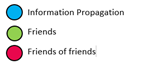
After obtaining the results from each algorithm, we collected the reviews for each restaurant audited by the influential user and using the hit ratio from each algorithm we visualized the information dissemination based on the friends of the influential user posting their own reviews about the same place.

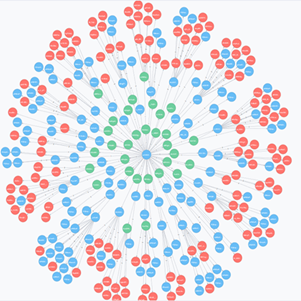




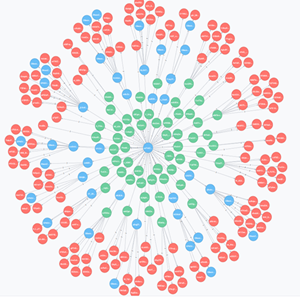
By choosing the highest influential user, we used Neo4j to visualize information dissemination to their friends. Due to higher similarity tastes in K-means algorithm there is more propagation observed compared to Twitter-rank algorithm. This is shown using the data from the K-means and Twitter-rank user 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 - "Yelp Dataset Challenge - Yelp". *Yelp*. N.p., 2017. Web. 27 Apr. 2017.**

**2- "Yelp/Dataset-Examples". *GitHub*. N.p., 2017. Web. 27 Apr. 2017.**

**[3] 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.**

**[4] 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**

**[5] David M Blei, “Probabilistic topic models,” Communications of the ACM, vol. 55, no. 4, pp. 77–84, 2012.**

**[6] 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.**

**[7] 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.**

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

**[9] 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**

**[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**