GENERATING STOCHASTIC DATA TO SIMULATE A TWITTER USER

Jason Li
Department of Computer Science
Ryerson University
245 Church Street, ENG-234
Toronto, ON, Canada
jcli@ryerson.ca

Abdolreza Abhari
Department of Computer Science
Ryerson University
245 Church Street, ENG-234
Toronto, ON, Canada
aabhari@scs.ryerson.ca

ABSTRACT

Twitter is a popular social network that carries information in short messages. A user's tweets can contain information that is similar to another user's tweets. In this research, we aim to provide stochastic tweets that can be used for testing recommender systems with large data. For this reason, we used term frequency and inverse document frequency (tf-idf) to analyze users' aggregated tweets. The empirical results show Weibull distribution fits the model of tf-idf of the words in users' tweets. Then Weibull distribution is used to generate stochastic data for users' tweets. A simulation of a recommender system was also conducted to test classification of users based on stochastic tweets. The recommender system uses collaborative filtering to find similarity between users. The simulation used k-means clustering to verify the similarity of the stochastic data versus real data.

Keywords: stochastic data, Twitter, k-means clustering, recommender system.

1 INTRODUCTION

Twitter is a continually growing social network that propagates information through its networks to its users. The messages known as tweets can contain information that are relevant to other users. Similar users would have tweets about similar topics. Datasets are often collected from real users when researching Twitter. The simulation of a Twitter network would contain real tweets. However, gathering a large dataset would require a time-consuming process. When the aim of research is the same as our previous research which is to find similarities between users who follow the same followee, collection of all such users may not be possible (Ahmed and Abhari 2014, Almeshary and Abhari 2013, Alrashoud, AlMeshary, and Abhari 2015). Also by collecting followers of different users there is also no guarantee that users that are following the same users are always similar to one another. Therefore, collection of real data may not be useful. Since finding the similarity between tweets is identified by clustering the users, in particular with the presence of big data, the interest of designers of recommender systems is to test the efficiency of their algorithms (Ahmed and Abhari 2015). Therefore, in this work we focus on generating stochastic tweets to test the algorithms of recommender systems for processing massive data sets.

The recommender system employs a collaborative filtering to produce recommendations (Nilashi et al. 2013). These algorithms require information from other users to obtain a recommendation for a user. Generating stochastic data will help test these algorithms by providing similar users. In the previous works of our research lab http://dsmp.ryerson.ca (Ahmed and Abhari 2014, Almeshary and Abhari 2013), we simulated simple tasks of a recommender system to provide recommendations (by clustering similar users)

SpringSim-CNS 2017, April 23-26, Virginia Beach, VA, USA

to the simulated Twitter user. In the next section this paper will go through related works, methodology, discussion, and conclusion. The creation of stochastic data for a user's tweets is proposed in this paper.

2 RELATED WORK

In this work, we created a simulation of a recommender system and conducted a simulation first by collecting real data from Twitter users. Then to test the simulation with artificial data, we created stochastic data for users' tweets which can simply scale up to a larger amount of data. To the best of our knowledge there is no similar work that tries to create stochastic data for Twitter messages. There are other works that aggregate tweets from a user into one document (Alvarez-Melis and Saveski 2016, Hong and Davison 2010). However, the aggregated tweets are not explored to generate stochastic data.

A multi-agent system has been created based on a Twitter financial community (Yang, Liu, and Mo 2014). Yang created a multi-agent system that tests the propagation of tweets based on different groups of users in the community. Their results showed that their simulation results were close to the real data.

A multi-agent system was used to create a distributed recommender system (Ahmed thesis, 2016). In his work, his focus is to increase the scalability of a distributed recommender system. Similar to common recommender systems, the recommender algorithms used both cosine similarity and k-mean clustering to verify the similarity between users.

Another multi-agent system that is created to simulate a recommender system is composed of agents representing users (Saga et al. 2011). There are user agents, a recommender, a controller, a collection of items, and a recorder. Each user is represented by a user agent and it will handle the preferences of a user. For example, the user agents will vote on the rating of the recommended items provided by the recommender. The goal of the recommender is to calculate similar users to determine a recommendation for a user agent. The controller oversees the whole process flow of the simulation. The recorder gathers the recommendation results to be evaluated. This multi-agent system is used to simulate a recommender system.

3 METHODOLOGY

3.1 Twitter Dataset

The data collected for the simulation is composed of 1000 real tweets. The 1000 real tweets in total were collected from 10 users. The tweets are a subset of a collection of tweets that were gathered over a 5-month period. The data was collected from the Twitter API using Twitter4J (Twitter4J 2016). Data was only collected from accounts that were set to public. We conducted this experiment with a smaller subset to gather empirical results. Table 1 contains an example of data gathered from Twitter.

Table 1: Example of data from Twitter.

Followee	TweetID	Date of Tweet	UserID	UserName	Tweet Message
RyersonU	71063617	2016-03-17 21:15	12345678	follower	If you don't watch the office I strongly recommend you do

Not all the information is required and the data must be preprocessed and cleaned before it is usable. Stop-words, hash-tag symbols, and any symbols not recognized in the English language are removed from the tweet message. We are interested in the username of the one tweeting and the tweet message in Table 1. These two parts are what the experiment will be working with.

3.2 Document Vector Space

We represent our text message from Twitter in a document vector as a bag of words model (Huang 2008, Manning, Raghayan, and Schütze 2009). A document vector consists of the weights of each word or term (Manning, Raghayan, and Schütze 2009). The weights of the terms in the documents are determined by the term frequency. The term frequency (tf) is the count of the number of times the term appears in the document (Manning, Raghayan, and Schütze 2009). Term frequency alone is not enough to represent the weights of the terms as it assumes all terms are weighed equally. Inverse document frequency (idf) of a term is used as well to lower the weights of the terms that are common (Manning, Raghayan, and Schütze 2009). Document frequency (df) is the number of documents the term appears in. The idf is $\mathbf{log}(\frac{N}{df})$ where N is the number of documents in the corpus. Thus, a product of term frequency and inverse document frequency (tf-idf) is used as weights for the terms in the document vector. In our experiment, a user is represented as a document containing an aggregation of all the user's tweets (Hong and Davison 2010).

3.3 Cosine Similarity

We determine the similarity between document vectors by using cosine similarity.

$$similarity = \cos(\theta) = \frac{A \bullet B}{\|A\|\|B\|} \tag{1}$$

Cosine similarity measures the cosine of the angle between vectors based on the formula in (1). If two document vectors are equal in all the weights of the terms, they would have an angle of zero between them. Since we are dealing with tf-idf, we are working with the positive space. Therefore, the value of cosine similarity falls in the range of 0 to 1. We can score the similarity between document vectors based on the cosine similarity (Manning, Raghayan, and Schütze 2009, Islam 2014, AlMeshary thesis 2015). A value closer to 1 for cosine similarity means the document vectors are more similar.

3.4 K-means Clustering

The clustering algorithm used in our experiment is k-means. K-means requires the number of clusters to be determined before starting the algorithm. In our experiment, we used k=3 and the same initial seeds are used for each test. The initial seeds are also the initial centroids of the clusters. Data points are assigned to clusters based on the distance between the data point and the centroids. The algorithm will iteratively assign data points to the centroids with the lowest distance. The centroids will also be updated as data points are assigned to the clusters. The algorithm is considered complete when the data points are assigned to the same clusters as the previous iteration. The data points in our experiment are the document vectors representing a user. Normally k-means uses the Euclidean distance in Euclidean space. The distance measure we use is cosine similarity from (1) for the document vector space (Singh, Tiwari, and Garg 2011). To retain the distance property, we modify the distance to be 1-cosine similarity (Huang 2008). Since the most similar vectors are equal to 1, 1-cosine similarity will equal to 0 which is the minimum Euclidean distance. The centroids are the document vectors representing all the terms in its clusters. The weight of each term in the centroid is the average of the terms from all document vectors in its cluster (Han and Karypis 2000). Clustering is the method for collaborative filtering in the recommender system simulator.

3.5 Generation of Stochastic Data

The first step to generating the stochastic data is to prepare the real dataset by processing the tweet messages. Afterwards, generate the tf-idf matrix consisting of all the users and terms in the corpus. The tf-idf matrix is a sparse matrix. Table 2 contains a submatrix of the tf-idf matrix.

Table 2: A submatrix of the tf-idf matrix.

	User E	User F	User G
abroad	0	0.017384	0
abt	0	0	0
abuzz	0	0.017384	0
academics	0	0.017384	0
accept	0	0	0.070835

To generate stochastic data for a user, we take the tf-idf values for all the terms in the corpus and arrange them in ascending order. The terms are kept in the order of the respective tf-idf value. We create bins equal to the number of terms. The values that will be placed in the bins start from the minimum tf-idf value for the user to the maximum tf-idf value. The range of each bin is $\frac{max-\min}{total\ terms}$. The bins are the indices of each term. From our experiment the Weibull distribution fit the users' tf-idf values which is visually verified in the next section. We generate a random value from a Weibull distribution and ensure it is at least the minimum value. The shape and scale parameters for Weibull distribution are the average of the shape and scale parameters found for the users' models in this dataset. The bin is selected by the closest value to the randomly generated number. We get the index corresponding to the bin and grab from it, so we can generate words randomly. The number of terms for a message is from 3 to 7. The minimum number of words from a tweet to be considered useful after processing is 3. The average number of words from a tweet in our dataset is 7. The process is repeated for each user until the desired number of tweets have been generated. This process generates stochastic data for Twitter messages.

4 DISCUSSION

The tf-idf matrix is generated by the program we developed to simulate a recommender system. From its data, we fit the tf-idf values with various distributions and found Weibull distribution to be the ideal distribution. A summary of tf-idf statistics for each user is shown in Table 3.

Table 3: The summary of tf-idf statistics for all the users' tweets used in the dataset.

	User A	User B	User C	User D	User E	User F	User G	User H	User I	User J
Mean	0.00254	0.00332	0.00375	0.00416	0.00697	0.01349	0.00494	0.00615	0.00734	0.01156
Median	1.00E-08	1.00E-08	1.00E-08	1.00E-08	1.00E-08	1.00E-08	1.00E-08	1.00E-08	1.00E-08	1.00E-08
COV	11.56583	8.82644	7.82062	7.02934	4.1174	1.94634	5.89324	4.69287	3.89905	2.35072
Model	Weibull	Weibull	Weibull	Weibull	Weibull	Weibull	Weibull	Weibull	Weibull	Weibull
Shape	0.22221	0.20421	0.19494	0.18881	0.1574	0.16149	0.16291	0.16234	0.14956	0.14264
Scale	3.94E-08	5.39E-08	6.69E-08	7.85E-08	3.06E-07	4.78E-04	2.62E-07	2.32E-07	6.46E-07	7.02E-05

In the plots from Figure 1 to Figure 4, we can see that Weibull distribution fit for most of the users. The empirical distribution is compared with the Weibull and Gamma distribution. Gamma distribution was also one of the candidates for modeling. By visual comparison of all the graphs it is clear Weibull and Gamma distribution are both comparable but Weibull is one of the most common distribution models that we observed in all the users' data. Since it is one the top three best models, we used it for modeling the tweets and chosen as the tf-idf random number generator for the stochastic tweets dataset. Weibull distribution was implemented to generate the random terms for each user in the program. Although the tests in the following Figures 1 to 4 show Weibull is a good model for the data that we collected, more data should be collected and more statistical tests should be done to confirm this visual observation. In the next section,

we show the clustering results of generated data is comparable with real data so we believe using Weibull distribution is an effective method to generate stochastic tweets.

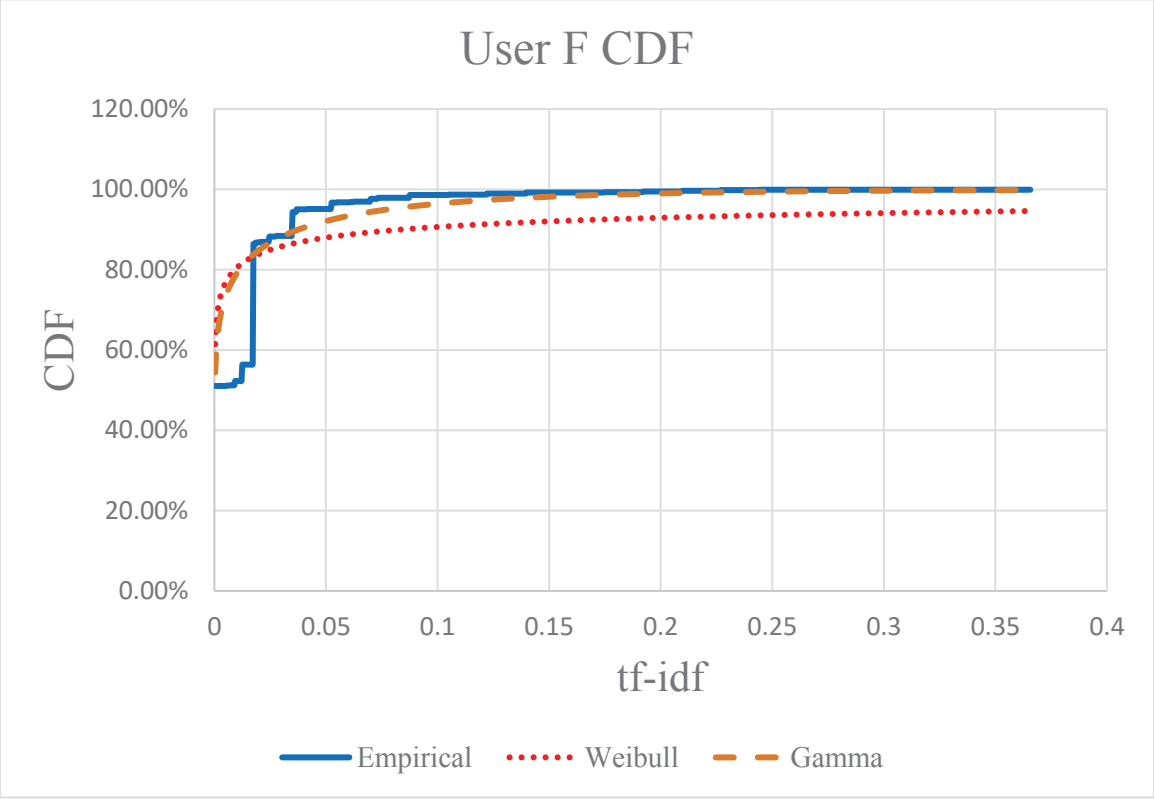

Figure 1: The cumulative distribution function for User F.

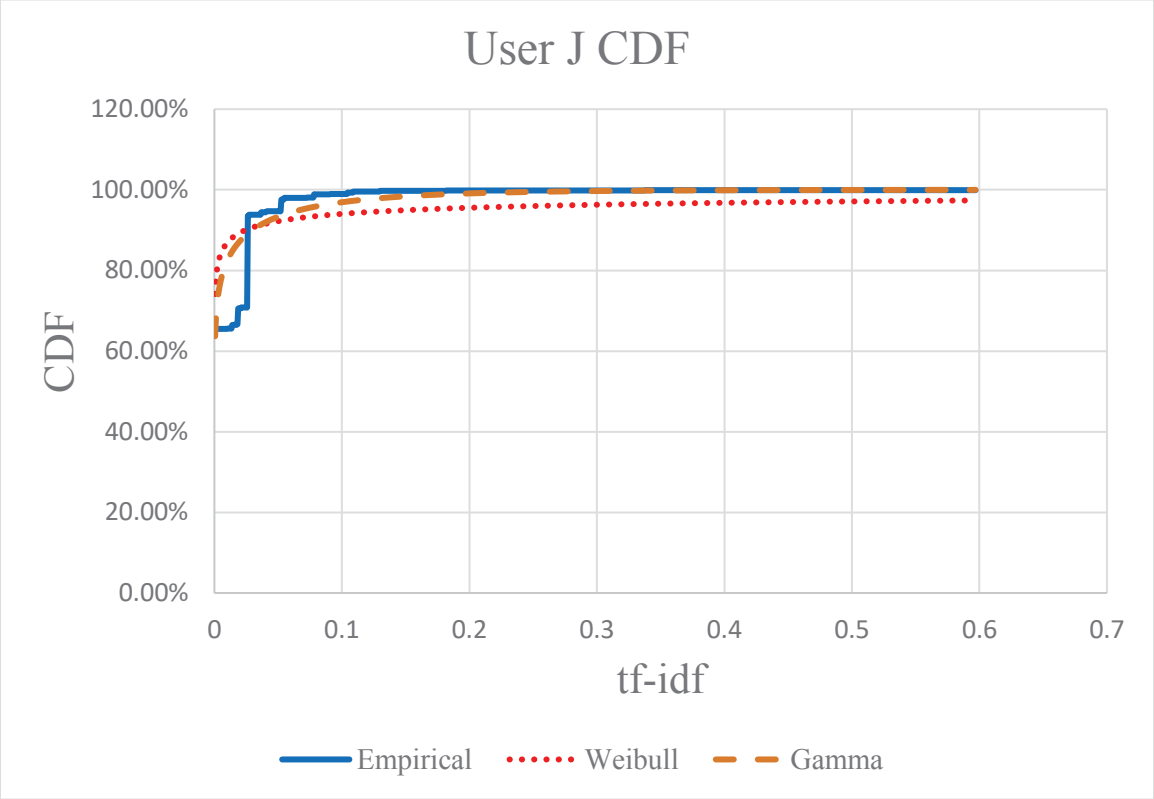

Figure 2: The cumulative distribution function for User J.

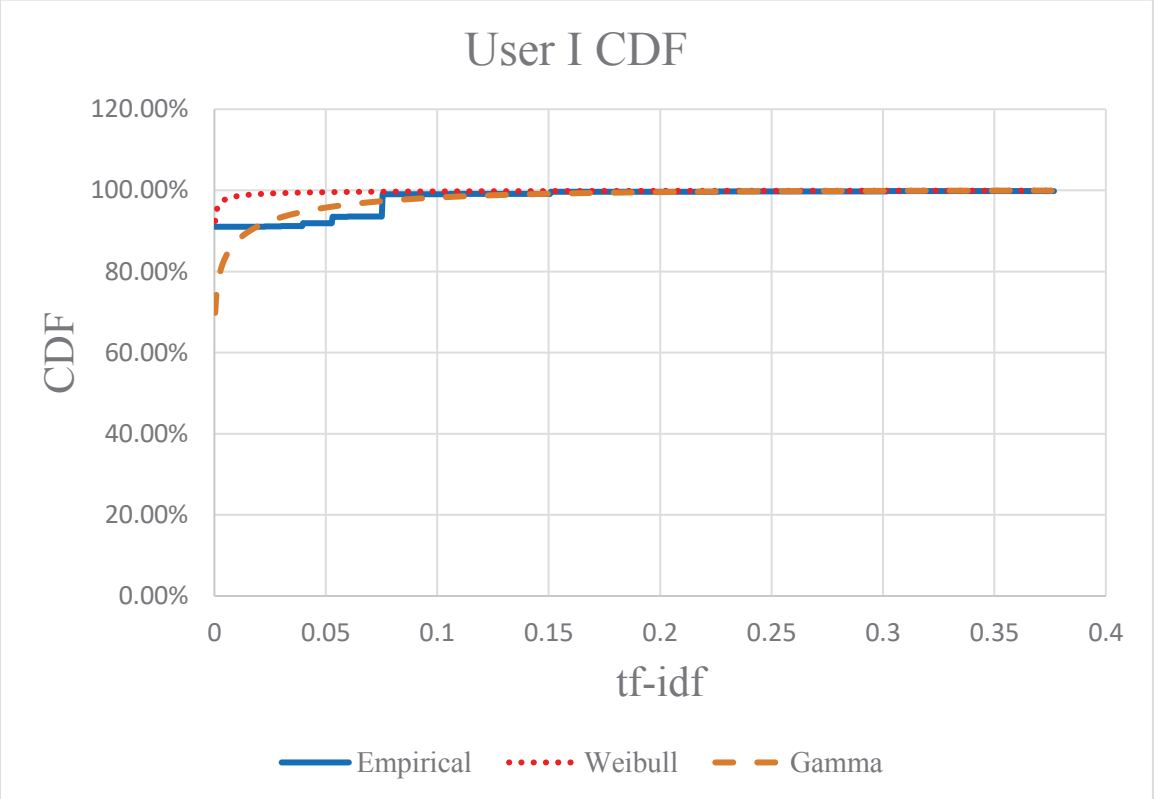

Figure 3: The cumulative distribution function for User I.

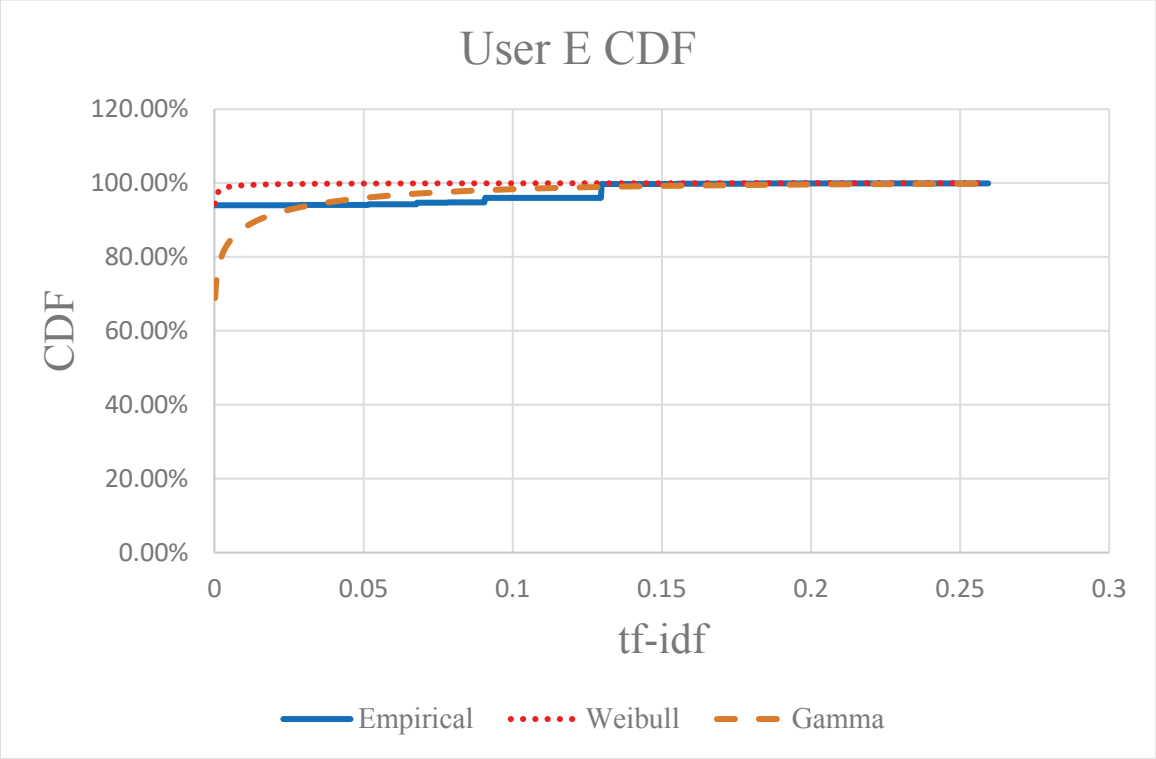

Figure 4: The cumulative distribution function for User E.

4.1 Recommender System

The simulator for the recommender system was created to test clustering of real and artificial data. The simulator program creates the tf-idf matrix and performs clustering on the document vectors with tf-idf weights. The following used pseudo data to provide an example of the process of the simulator program. Table 4 shows example documents after text processing with the Twitter user's name on the left.

Table 4: Example documents after text processing with the Twitter user's name on the left.

Steve	network york times baseball
David	network york post
Alice	los angeles rain times
Bob	dog food price dog pet
Fred	sky blue dog

Clustering is performed after the terms in each document vector are assigned a tf-idf. The number of clusters chosen is 3 for k-means because by visual inspection for this example, we can tell there will be 3 clusters of similar documents. We also used visual inspection to verify the results of the simulator. K-means will randomly assign 3 of the documents as the initial seed. For this example, Steve, Alice, and Fred will be chosen as the seeds for the centroids. The centroids for this iteration in this example are the document vectors of Steve, Alice, and Fred. David and Bob are the remaining document vectors that need to be assigned to a cluster. The document vectors of David and Bob are assigned to their clusters based on the distance measure of cosine similarity. The centroid document vectors of Steve, Alice, and Fred will conduct cosine similarity with David and similarly for Bob. David and Bob are assigned to the respective

clusters that have the lowest distance from the centroid vectors. In this example, David is assigned to the cluster with Steve while Bob is assigned to Fred. We update the centroid tf-idf weights with the average of the terms in the cluster. Since all document vectors are assigned to a cluster, we iterate this again with all document vectors being compared to the current centroid vector weights. We repeat the process if needed to but for this example, there are no new assignments after the first iteration. The clustering results of this example are shown in Table 5. The cluster number is the index of the cluster. The centroids are the terms from all the documents that are contained in its respective cluster. The cluster members refer to the number of documents in the cluster. The terms within the documents and usernames of the documents are also included in the cluster results. By visual inspection, the resulting document vectors are similar to each other in their respective clusters.

Table 5: Clustering results of example documents from Table 4 given by the simulator.

[Cluster: 0]	[Cluster: 1]	[Cluster: 2]
[Centroid: angeles los rain times]	[Centroid: blue dog food pet price sky]	[Centroid: baseball network post times york]
[Cluster members: 1	[Cluster members: 2	[Cluster Members: 2
Alice, los angeles rain times	Fred, sky blue dog	Steve, network york times baseball
]	Bob, dog food price pet	David, network york post
	]	]

The simulator will complete the recommendation after giving a score based on cosine similarity between the user given the recommendation and the other users. The document vectors within the same cluster as the recipient of the recommendations will have a higher score. If we chose Fred to be given recommendations, the resulting recommendations will be Bob, Alice, then David. In Figure 5 the results are shown from the simulator. The top 3 recommendations for the user Fred that is selected on the left are provided under the heading of "Recommendations for User".

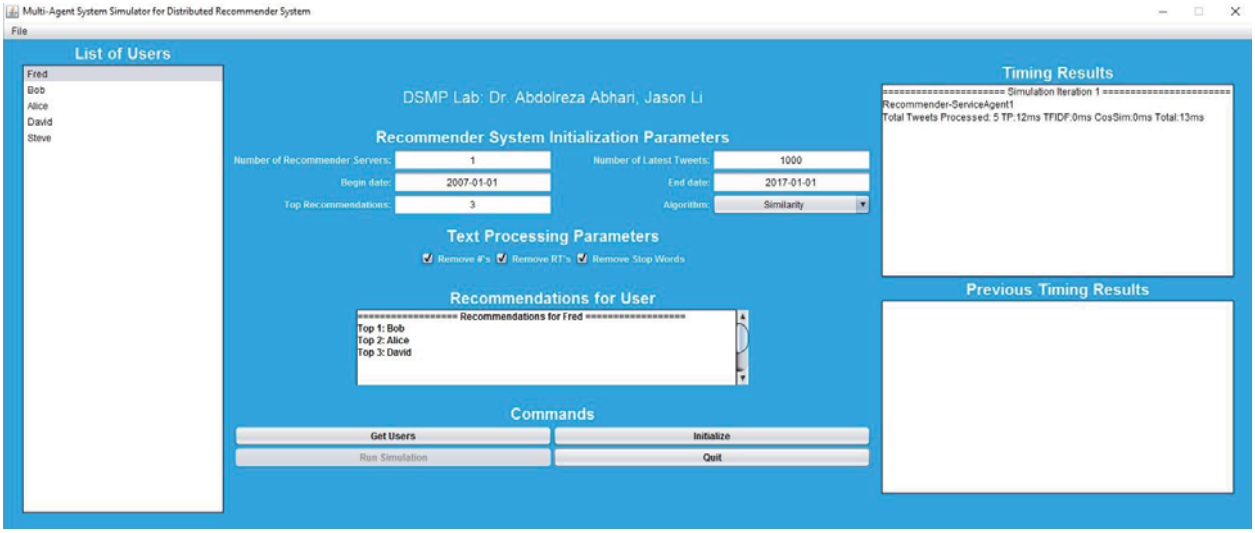

Figure 5: The result of recommendations for Fred after clustering in the simulator.

Figure 5 shows the user interface of the simulator of the recommender system. It provides recommendation by conducting clustering on the document vectors of the users. Our experiment clustered the real data and the generated data separately. The number of clusters was chosen to be 3 again for consistency. The initial seeds were Users B, D, and E. The generated users (stochastic data) follow the naming convention of "_g" after the username. Users B_g, D_g, and E_g were chosen respectively for the clustering of the generated data. The generated data contained the same number of tweets as the real data. The clustering results are shown in Table 6.

Table 6: Clustering results of the real and generated data.

Real Data			**Generated Data**		
Cluster 1	**Cluster 2**	**Cluster 3**	**Cluster 1**	**Cluster 2**	**Cluster 3**
User A	User G	User D	User C_g	User F_g	User A_g
User B		User F	User J_g		User E_g
User C		User E			User H_g
User J		User H			User G_g
		User I			User D_g
					User I_g
					User B_g

As it is shown by Table 6, the clustering result of stochastic data is visually comparable with the real result. The results are promising and generated stochastic data can be easily scaled up to millions of tweets to test the efficiency of recommender system algorithms.

5 CONCLUSION

The recommender system designer need to test the efficiency of their algorithms against large amounts of data. The problem is such data collection is very time consuming for Twitter users because it needs to be done in the specific time intervals with the limitation of available APIs. This paper proposes a method to generate stochastic tweets. The proposed methodology provides stochastic data for users' tweets. The empirical data and visual observation of the distribution fittings on a real dataset shows the tf-idf of each user follows a Weibull distribution. The Weibull distribution is used to generate tf-idf and from that, stochastic words are generated to mimic users' tweets. The simulation of a recommender system verifies the generated stochastic data and they can be clustered similar to real data. Clustering of stochastic data ensures the classification of users (the main task of recommenders) can be done. Therefore, the efficiency of their employed algorithms can be tested. The model proposed by this paper can be used for creating a massive amount of stochastic user tweets that can be used by recommender system designers.

REFERENCES

Ahmed, L., and A. Abhari. 2014. "Agent-Based Simulation of Twitter for Building Effective Recommender System". In *Proceedings of the 17th Communications & Networking Simulation Symposium (CNS 2014)*, pp. 30–36. San Diego, California, The Society for Modeling and Simulation International (SCS).

Ahmed, L., and A. Abhari. 2015. "Distributed recommender system for online processing of big social data". In *Proceedings of the Poster Session and Student Colloquium Symposium (Posters '15)*, pp. 15–16. San Diego, California, The Society for Modeling and Simulation International (SCS).

Ahmed, Lubaid. "Distributed Recommender System Using Multi-Agent For Social Networks." PhD diss., Ryerson University, 2016.

Almeshary, M, and A. Abhari. 2013. "A recommendation system for Twitter users in the same neighborhood". In *Proceedings of the 16th Communications & Networking Symposium*, pp. 1-5. San Diego, California, The Society for Modeling and Simulation International (SCS).

AlMeshary, Meshary Abdulrahman. "Follow your neighbors and engage to the new culture." Master's thesis, Ryerson University, 2015.

Alrashoud, M., M. AlMeshary, and A. Abhari. 2015. "Automatic validation for multi criteria decision making models in simulation environments". In *Proceedings of the 18th Symposium on Communications & Networking (CNS 2015)*, pp. 44–47b. San Diego, California, Society for Computer Simulation International (SCS).

Alvarez-Melis, D., and M. Saveski. 2016. "Topic Modeling in Twitter: Aggregating Tweets by Conversations". In *Proceedings of the Tenth International AAAI Conference on Web and Social Media (ICWSM 2016)*, pp. 519–522. Palo Alto, California, Association for the Advancement of Artificial Intelligence.

Han, E. H. S., G. Karypis. 2000. "Centroid-Based Document Classification: Analysis & Experimental Result". In *Proceedings of the 4th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, pp. 424–431. Lyon, France, European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases.

Hong, L., and B. D. Davison. 2010. "Empirical Study of Topic Modeling in Twitter. In Workshop on Social Media Analytics". In *Proceedings of the First Workshop on Social Media Analytics*, pp. 80–88. Washington D.C., District of Columbia, Knowledge Discovery and Data Mining.

Huang, A. 2008. "Similarity Measures for Text Document Clustering". In *Proceedings of the New Zealand Computer Science Research Student Conference 2008*, pp. 49–56. Christchurch, New Zealand, New Zealand Computer Science Research Student Conference.

Islam, Masudul. "Personalized Recommender System on Whom to Follow in Twitter." Master's thesis, Ryerson University, 2014.

Manning, C.D., P. Raghavan, and H. Schütze. 2009. "Scoring, term weighting and the vector space model". In *An Introduction to Information Retrieval*, pp. 109–133. Cambridge, Cambridge University Press.

Nilashi, M., K. Bagherifard, O. Ibrahim, H. Alizadeh, L. A. Nojeem, & N. Roozegar. (2013). "Collaborative filtering recommender systems". *Research Journal of Applied Sciences* vol 5, pp. 4168–4182.

Saga, R., K. Okamoto, H. Tsuji, and K. Matsumoto. 2011. "Proposal of a recommender system simulator based on a small-world model". *Artificial Life and Robotics* vol 16, pp. 426–429.

Singh, V. K., N. Tiwari, and S. Garg. 2011. "Document Clustering using K-means, Heuristic K-means and Fuzzy C-means". In *Proceedings of the 2011 International Conference on Computational Intelligence and Communication Systems*, pp. 297–301. Piscataway, New Jersey, Institute of Electrical and Electronics Engineers, Inc.

Twitter4J. "Twitter4J – A Java library for the Twitter API." http://twitter4j.org/en/. Accessed: Mar. 22, 2016.

Yang, S. Y., A. Liu, and S. Y. K. Mo. 2014. "Twitter Financial Community Modeling using Agent Based Simulation". In *Proceedings of the 2014 IEEE Conference on Computational Intelligence for Financial Engineering & Economics (CIFEr)*, pp. 55–62. Piscataway, New Jersey, Institute of Electrical and Electronics Engineers, Inc.

AUTHOR BIOGRAPHIES

JASON LI is a Master's student of Computer Science at Ryerson University. He is completing his Master's thesis. His research interests lie in multi-agent systems, simulation, and data science. His email address is jcli@ryerson.ca.

ABDOLREZA ABHARI is a Professor in the Department of Computer Science at Ryerson University and director of DSMP lab (http://dsmp.ryerson.ca). He holds a Ph.D. in Computer Science from Carleton University. His research interests include data science, data mining, Web 2.0 social networking, modeling and simulation, network and database systems, sensor networks and distributed systems. His email address is aabhari@ryerson.ca