Who Can Be My Next Friend?
(Friend Recommendation System)
CS555 : Distributed Systems

Harshil Shah
Computer Science
Colorado State University

December 9^{th}, 2015

Contents

1 Problem Statement & Importance

The need of the day in the 21st century on the social media platform has been analyzing user preferences. [11] Social networking is now-a-days "basic" need of every person. For getting update regarding activities going on around person's life is an important feed for that person. We, as users, usually rely on some external knowledge to make informed decisions about a particular artifact or action [7]. Every person is more inclined towards knowing what the people, he/she is following, are doing. Person will be provided such information by social media from his/her connection list. Social media is very well aware of user's behavior, interest and should provide suggestions of people to user. Any suggestions made by social media should be highly accurate and convincing. Now what if system has to give suggestion from abstract dataset? Furthermore, what if user has just entered social media - how system will predict, as system will not know preferences of users. What if items, user likes, are lacking of characteristic information? And it is not uncommon that large social media will surely have fake users - how to deal with this? It is highly required that any suggestions made to people should be highly accurate and genuine.

Sometimes, users do not know what they want (what kind of services they want, what kind of people to follow etcetera)? As far as social-media is concerned, last thing the users want is to have connection with important people(affection) without putting much effort. User may stop using any media if he/she has to put so much effort finding relevant stuff. Any type of powerful recommendation system directly contributes to success of social networking websites or any e-commercial websites. Recommendation system can not only be used from client perspective, but also for business perspective. Using correct implementation, companies can predict future products they should make, whether their newly launched product is going to be popular or not.

As far as this report is concerned, report represents the project implementation that underpins the key feature of suggesting friends with higher accuracy using collaborative filtering approach. FriendSter - social networking website's - dataset was used as far as project is concern.[2] Dataset contains details of nearly 125 Million users and up-to 2 Billion edges. Each edge represent directed following tendency from user to user. Dataset is too abstract - as no user's characteristic information is provided. Only edge link descriptive dataset was provided and developed model needs to use such dataset for recommendation.

2 Possible Solutions & Trade-Off Space For Solution

Basically, for any prediction system or recommendation system implementation, below shown models can be used to filter out results.[1]

Content Based Filtering

The system generates recommendations using - the features associated with items/services and the affection shown by a user to them. Content-based recommenders treat recommendation as a user-specific classification problem and learn a classifier for the user's likes and dislikes based on items'/services' features. For each individual user, any machine learning classifier will be applied and user to service/items/users matrix will be generated. Considering billions of users, matrix will be too huge & dense and space will be traded off. Furthermore, for each user, algorithm's weight will be different - so for training of model, computation time will also be too much. Another key issue with content-based filtering is whether the system is able to learn user preferences from user's actions regarding one content source and use them across other content types. Furthermore, as far as this project is concerned, content based cannot be applied as dataset is too abstract - it is lacking from users' features list. [4][12]

Geographical Based Filtering

A demographic recommender provides recommendations based on a demographic profile of the user. Due to lack of location based data, this approach is not applicable to this project either.

Collaborative Filtering

The system generates recommendations using information about affection behavior of different users. Collaborative filtering works on the assumption that if a set of users have agreed on a certain idea at some point of time, it is very likely that they would agree on another related idea in future too. A key advantage of the collaborative filtering approach is that it is capable of accurately recommending complex items without an "understanding" of the item itself. As far as this project is concerned, Collaborative approach can be used for FriendSter dataset because of this characteristic. Though this approach works pretty well using Pearson Correlation, the main requirement of this approach is large amount of user activities and preferences. Collaborative approach often suffers from cold start problem too. Furthermore, computation time can also be large if calculation of correlation matrix is defined in weird way. [3][6][11][13]

Hybrid Approach

To overcome the inaccuracies of above mentioned models, a hybrid approach can be used. In this approach combination of the approaches for recommender systems can be applied independently and then can be combined using specified weight matrix. [5]

Now, as far as this project is concerned, Collaborative filtering approach will be used. Collaborative can be applied in various ways. Creating user-user similarity matrix in which user A and user B will be having weight according to number of mutual friends between them. Or, Creating user to user matrix according to if person is friend with user A and also with user B. [17] For this project, user-user similarity based on mutual friends was created and used it for further recommendations. [10]

3 Methodology

For evaluation purpose, this project is following traditional testing technology only. For this project, whole dataset was randomly divided into training-testing datasets. Module was trained over training dataset & evaluate over testing dataset. As supervised learning was implemented, model was tested perfectly in traditional way only (accuracy - as evaluation parameter).

Dataset contains users' following list. Every user's outgoing link (directed) was shown in dataset with having no further information at all. In dataset, either users contain some following friends list (may contain zero friend) or users may be private (not publicly accessible list of friends) or users may have left social media. User ID in dataset was numeric only and dataset looked like as shown below.

```
102 : notfound
109 : private
205 : 102, 209, 215, 231, ..
3015 :
```

Whole computation was divided into small phases. Description regarding each phase is given below.
Phase One

For constructing user-user matrix, only candidate users were chosen who actually can be related. To reduce memory space, we need to comprise user-user matrix. Although finding relation, between user and not available or zero friend user, is waste of computation time and waste of space.

Using inclination of content based approach, Dataset was parsed and users with more than zero friends were considered only. Unwanted users were also removed from the following list of any other users. Not found users were meant to be removed. Users with zero friends won't have any implications on prediction of friends recommendation. And to keep dataset consistence, private users were also meant to be removed. Comprised users list not only had reduced space, but also further recommendation calculation too.

Phase Two

For evaluation purpose, random partition of dataset was needed. Dataset was partitioned based on following list of each user. Following list of each users from parsed dataset of *Phase One*, is randomly partitioned into 70:30 ratio.

Generation Of User-User Similarity Matrix

As discussed earlier, similarity between users were generated on basis of mutual friends. Two approaches are possible. Compare each user with remaining users with the both user's following list. This approach can be done by writing only one MapReduce job. But with the use of commodity machines, this approach is not feasible as it is too compute intensive & data transfer intensive. Instead writing three jobs, 3 MapReduce jobs were written to compute similarity matrix. Description of each phase won't be sufficient to have good idea regarding phases, so detailed jobs description is shown in below sections.

Phase Three

In this phase, users' list was created for every common friend.

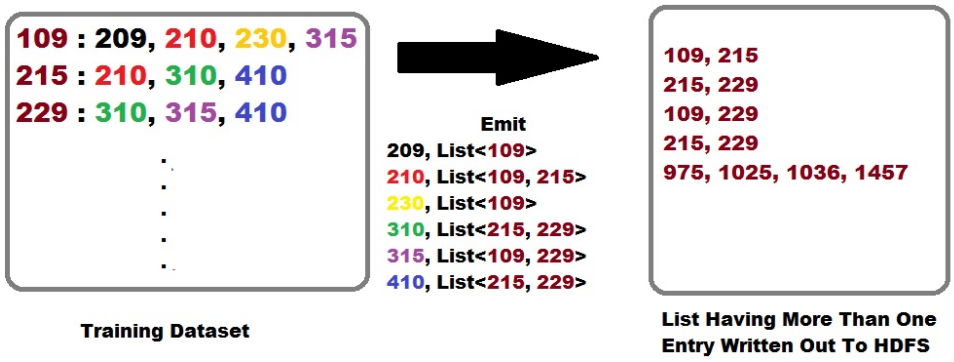

Figure 1: Phase 3

Phase Four

For every list from output of phase three, every possible combination of users were emitted with having weight of 1 & sum of total weight of same combinations were written to HDFS. Instead of calculating every combination in n * (n-1) manner, only n * (n-1) / 2 combinations will be useds - as (109, 232) is same as (232, 109).

Phase Five

As previous phase results are just half done, for final calculation and generation of user-user similarity matrix, for every user-user combinations from previous phase, for both users in combination, relative user and weight will be emitted (mapper) and will be combined in reducer.

Combined results will be sorted according to weight of each users for every key(user) and top 75 users will be written in one row of matrix associated with each key(user).
So after distributed execution of each phase, similarity matrix will be generated with lowest possible computation in Mappers-Reducers & with low possible data transfer.

Recommendation Phase

Recommendation phase is using User-User similarity matrix, users' actual following list from training dataset & users' testing list for evaluation purpose.

Recommendation Phase One

Recommendation of testing users will be done on basis of similar users of testing user's following list (available from training dataset). Each user from following list of similar user will be assigned weight equivalent to similar user's weight to testing user(available from user-user similarity matrix). Results will be combined for every testing user - weighted sum of possible recommended users. Sorted list of possible recommended users according to weights will be recommended to testing user. Now as model is following supervised learning technique, testing user's actual following list will already be known. Accuracy can be calculated on basis of matching recommendation of prediction list and actual list. Lexical description of this phase is bit confusing but it is easy to visualize. (see figure : 2)

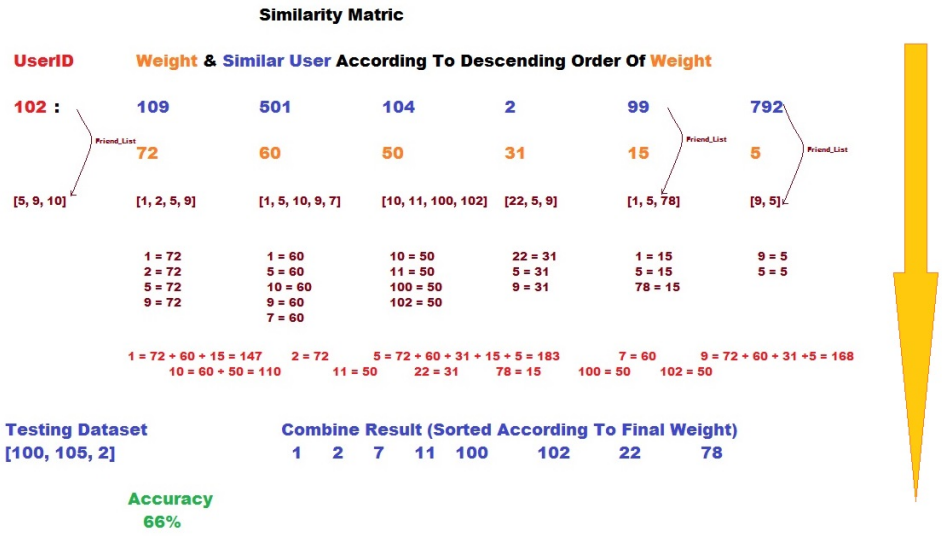

Figure 2: Recommend Phase 1

For some users, recommendation was not done. Such users are suffering from cold start issues. Possible explanations regarding such users are - (i) they have low number of following list so no similar users have been found, or (ii) they have so big following list such that because of overlapping no further recommendation can be done.

Recommendation Phase Two

For cold start users, collaborative does not work fine. Union of user's following friends' friends will be recommended to users. If user does not have any friend initially then above approach wont work either, then such users will be recommended system's most popular users.

For evaluation of system, standard tradition approach to test module was followed and accuracy observed was nearly 50%. So for any recommendation model, such accuracy can be considered higher. So full system's recommendation can also be done in similar way. Instead of training model on training dataset only, module will be trained on whole dataset and recommendation will be done. (see figures : 3 & 4)

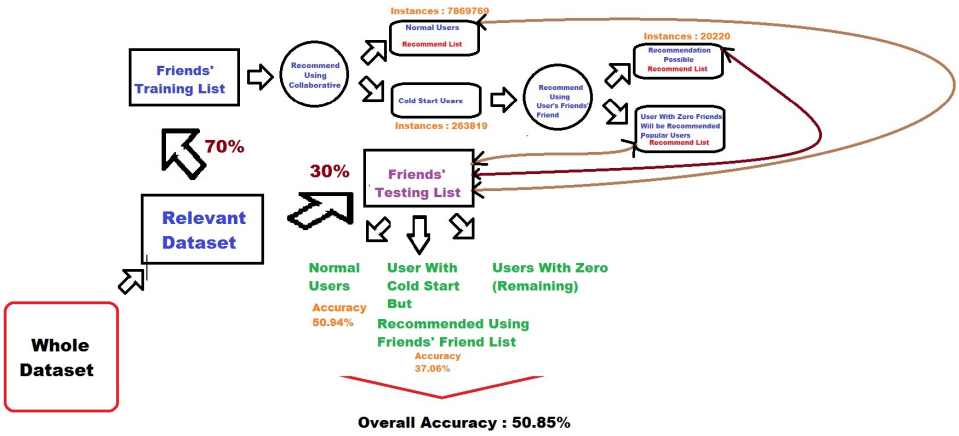

Figure 3: Evaluation Model

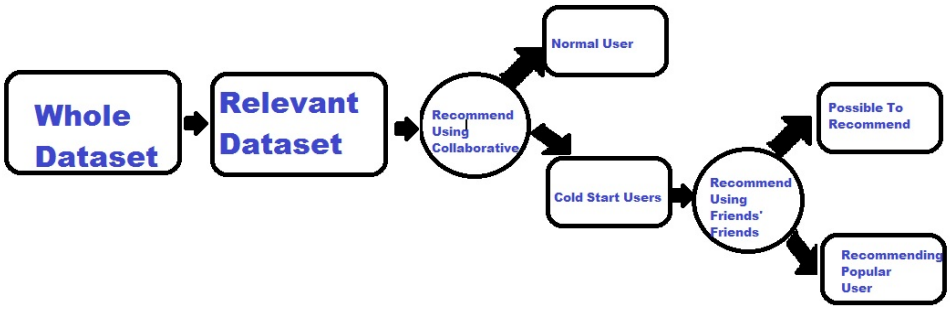

Figure 4: Recommendation Model

How To Store Huge Data To Make Program Client Interactive? [*Future Work]

Recommendation files containing each user's recommendation lists is pretty huge (nearly 350GB). For every users, new MapReduce job cannot be implemented. Recommended data needs to be dispersed over different commodity machines and client's request needs to be handled. Chord implementation can be used to store such huge data with having userID as key. But more powerful Apache Casandra or Dynamo like file system structure (key, value pair) can also be used to back-end system. Future work will be to store data over any file system structure and make client's API interactive with back-end to get recommendations.

4 Results And Analysis

In figure shown below, collaborative filtering approach covered nearly 97% of whole dataset. Only 3% of users are suffering from cold start issue. Regarding that, main issue of using collaborative filtering is solved as only few instances are suffering from cold start. Accuracy of normal users is nearly 51% and for cold start users is 38%. Combine accuracy is nearly 50.85%. Having accuracy of nearly 50% suggests good system implementation. Furthermore, for cold start suffering users, having accuracy of more than 37% is not bad either.

Testing User	**8133588**
Recommended Users	**7869769 (96.76%)**
Cold Start Users	**263819 (3.24%)**

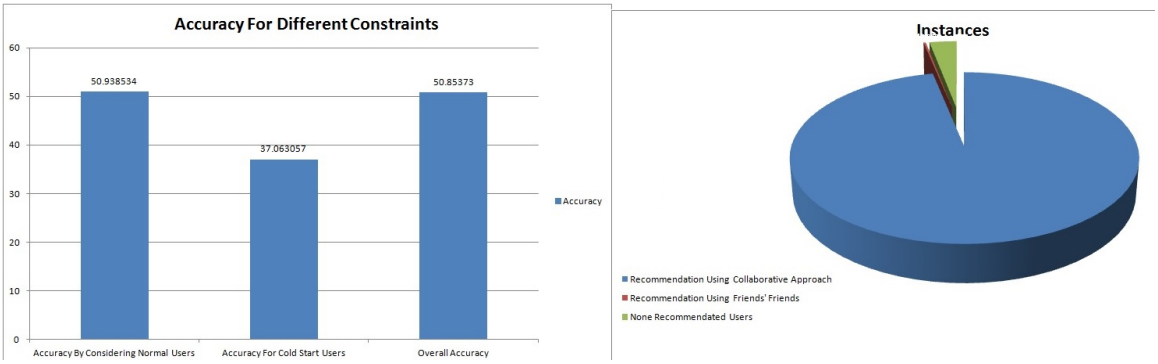

Figure 5: Accuracy Of Different Constraints

Let's consider diagram 6. Diagram suggest accuracy distribution across number of users. Graph looks like normally distributed range. Number of instances having highest and lowest accuracy are very higher. Meaning, either algorithm is giving, in most of the cases, very nice solution or very bad solution. Reason behind such weird behaviour can be cold start suffering users and number of fake users' impact. Collaborative filtering does not consider user's behaviour towards recommendation - so it fails to encounter fake users. According to data of 2014, facebook stated that nearly 11.2% users were fake. Even twitter has also confirmed that out of 10 users, 1 is fake user. [14][15][16] Future work regarding expansion of recommendation system, will be to integrate content-based filtering to figure fake user using abstract dataset only.

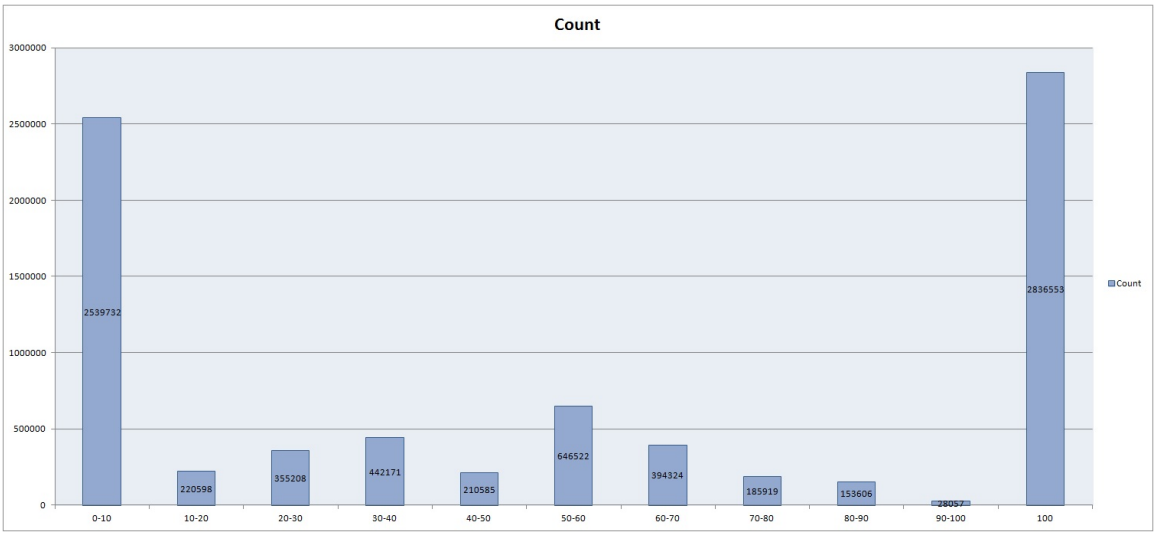

Figure 6: Accuracy Distribution

5 Performance Benchmarks

For prediction and evaluation purpose, randomly nearly 32M users' dataset were chosen. Intermediate output regarding such dataset was 3.25TB. If whole dataset was used then nearly 10-12TB of intermediate data may have been generated and it may have taken 10-12 Hours to complete execution. As commodity machines used for this project has limited storage system, project was implemented on randomly partitioned, but enough for conclusion, dataset. Below table suggest partition results. Randomly partition was performed to make module unbiased. Nearly 25 commodity Linux machines were used for concluding this project.

-	Actual Dataset	Used Dataset	%
Nodes	65608366	32000000	48.77%
Edges	1806067135	651051111	36.05%
Total Memory Used Considering Replication	10TB (Anticipated)	3.23 TB	-
Time Taken	10-12 Hours (Anticipated)	2 Hours	-

Collaborative filtering needs dense matrix - dense dataset. If user's friend list is null then that user will eventually become useless in further computation. In dataset provided by FriendSter, there are so many private and deactivated users. Friend list of private dataset is not accessible as far as this project goes. Furthermore, many users in dataset were having zero following list. So instead of automatic removal of this kind of users by algorithm, to speed up process and to save memory, using content based approach, by using characteristic of users, such users were removed. As below graph depicts, computation to create similarity matrix was done only on 25% whole dataset only(user with more than zero friend).

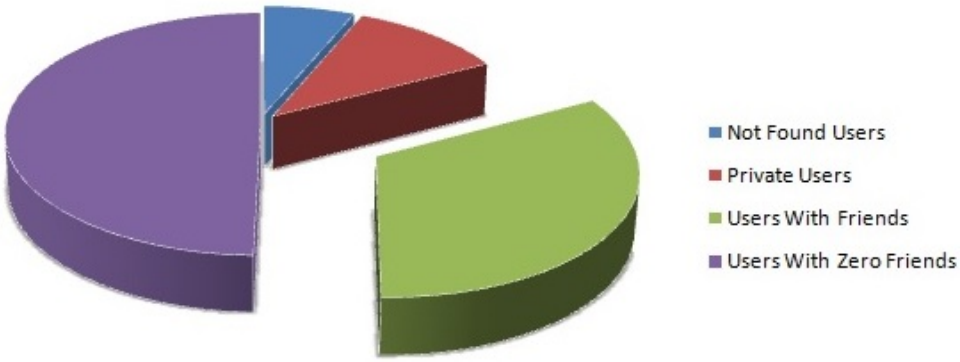

Figure 7: Relevant Users

Using collaborative two types of matrix can be generated - (i) User-User Similarity Matrix & (ii) User To User Matrix suggesting if client follows user A then chances of client to follow user B too. Both matrices work well in recommendation, but computation time and space used by both differs them from each other. Considering project scenario, each users have average 20 edges and there are 8M users, so nearly O(m) * O(n) scanning and execution time may require. But for creating User-User similarity matrix based on mutual friends, only O(m) scanning and execution time will be required. Both matrix will require almost same space allocation as far as this project goes. So to save time only, similarity matrix based on mutual friends is generated.

Below shown diagram suggests that, for user having less than 7 friends, more than 50% of such users will likely to be suffered from cold start problem. For future work, if all such users are considered as cold start users beforehand, and recommend them friend based on their friends' friends list. Accuracy regarding such recommendation is not so good and not so bad either (37%). But computation time to recommend such user will drastically reduce as no further computation regarding them will be needed for applying collaborative approach. Figure 9 suggests that user will having less than 10 friends span nearly 50% of total number of users. So, proposed idea suggests that computation and memory space will directly reduce by 50%.

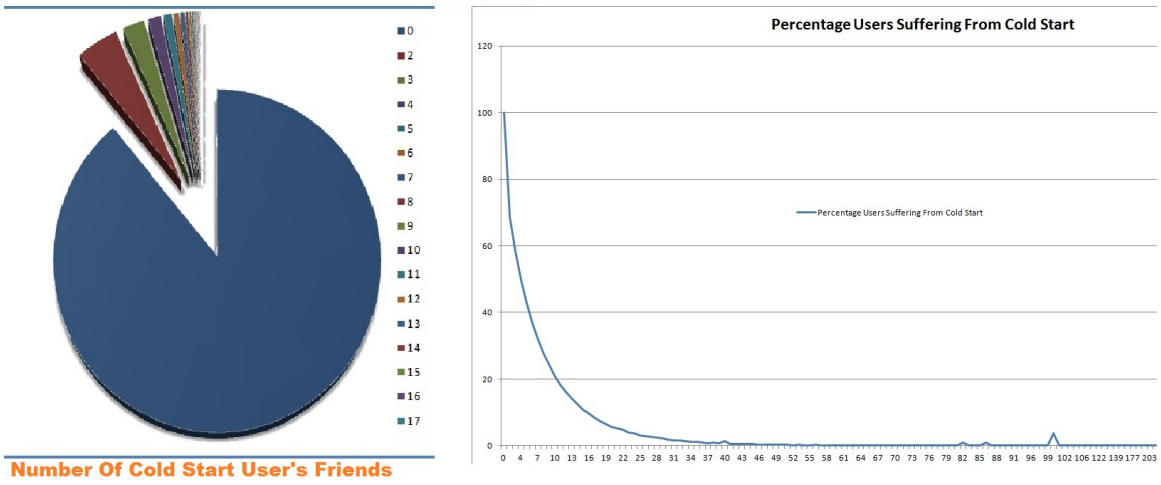

Figure 8: User's Behaviour Suffering From Cold Start

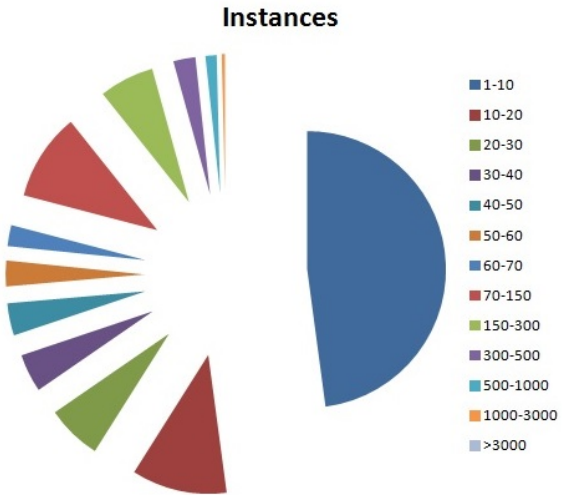

Figure 9: Number Of Friends Distribution

6 Key Innovations

For evaluation purpose, to speed up process and with the use of low memory usage, random partition of dataset seems good new approach. Prediction of whole dataset will remain same as small subset of dataset, if small subset does not contain any bias entries. Removal of users which are going to removed, anyway, by algorithm beforehand, speeds up computation and uses low possible memory. Selection of good approach for collaborative is totally data-dependant. For user - user dataset, calculation of user-user similarity matrix will be beneficial. But if user-item dataset is provided, then according to behaviour of dataset, approach should be choosen.

For future work, new idea regarding intelligent way to recommend friends to user (mentioned in section 5) seems more convincing and scalable with having too large dataset. User with lower friends will be recommended faster and user with large friend list will be recommended best possible recommendation.

Eventually, user with lower friend list will also be recommended best possible recommendation. Furthermore, for extension of project, recommended data will be stored on Chord like structure or Apache Casandra file system structure. Client's API will be made to communicate with file system & get recommendation service. As collaborative approach gives good predictions and very bad predictions as well - because of probability of fake user existence, in future work, algorithm will be chosen to remove impact of such users on result.

Proposed collaborative recommendation model is giving good accuracy. Model can be used on any dataset, where content is too abstract. So for such dataset, for which content based is nearly impossible to apply, proposed model can be used. Model is both scalable and uses memory and computation intelligently. Furthermore, model can be used to find out popularity trend, reach ability of any product, etc. Model can be used in companies, to predict whether the newly launched product is going to be popular or not.

To sum up, as far as this project is concerned, back-end infrastructure regarding friend recommendation system was implemented using collaborative filtering. Even though, dataset was too abstract, accuracy of nearly 50% was noted.

References

[1] Recommendation System - Wikipedia
en.wikipedia.org/wiki/Recommender_system

[2] Recommendation System - Wikipedia
https://archive.org/details/friendster-dataset-201107

[3] Collaborative Filtering - Wikipedia
https://en.wikipedia.org/wiki/Collaborative_filtering

[4] Content Based Filtering - Wikipedia
https://en.wikipedia.org/wiki/Recommender_system#Content-based_filtering

[5] Hybrid Recommendation - Wikipedia
https://en.wikipedia.org/wiki/Recommender_system#Hybrid_Recommender_Systems

[6] Cold Start - Wikipedia
https://en.wikipedia.org/wiki/Cold_start

[7] Basui, Chumki, Haym Hirsh, and William Cohen. "Recommendation as Classification: Using Social and Content-Based Information in Recommendation." (1998).

[8] Hsu, William H., et al. "Collaborative and Structural Recommendation of Friends using Weblog-based Social Network Analysis." AAAI Spring Symposium: Computational Approaches to Analyzing Weblogs. 2006.

[9] Wilson, Nathan R., Emily A. Hueske, and Thomas C. Copeman. "Systems and methods for providing recommendations based on collaborative and/or content-based nodal interrelationships." U.S. Patent No. 8,756,187. 17 Jun. 2014.

[10] Fan, Jiaqi, Weimin Pan, and Lisi Jiang. "An improved collaborative filtering algorithm combining content-based algorithm and user activity." Big Data and Smart Computing (BIGCOMP), 2014 International Conference on. IEEE, 2014.

[11] Sanghavi, Bhavya, Rishabh Rathod, and Dharmeshkumar Mistry. "Recommender Systems-Comparison of Content-based Filtering and Collaborative Filtering." (2014).

[12] Daniyalzade, Eytan, and Tim Lipus. "Facebook Friend Suggestion."

[13] Recommendation System - InfoLab Stanford
http://infolab.stanford.edu/ ullman/mmds/ch9.pdf

[14] Fake Users On FB - TheNextWeb News - 2014
http://thenextweb.com/facebook/2014/02/03/facebook-estimates-5-5-11-2-accounts-fake/

[15] Fake Users On FB - CNET News - 2012
http://www.cnet.com/news/facebook-8-7-percent-are-fake-users/

[16] Fake Users On Twitter - NBC News - 2012
http://www.nbcnews.com/tech/internet/1-10-twitter-accounts-fake-say-researchers-f2D11655362

[17] Amazon Recommendation System
http://www.cs.umd.edu/ samir/498/Amazon-Recommendations.pdf