Social Interactome Recommender Project

by Andrew Baehre, Colin Gray, and Trevor Kinaman
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061

Final Project Report CS 4624
May 4, 2016
Client - Prashant Chandrasekar

Table of Contents

Table of Tables

Table of Figures

1.0 Executive Summary

Our team is working with the Social Interactome team to assist in coding the recommender functionality for the Social Interactome network, supported by a website and system (based on Friendica) designed by the Social Interactome team to help recovering addicts. The team used Python to parse participant's answers to survey questions, and to apply an algorithm to that data to show each participant's most favorable friend matches. The team is working in concert with Prashant Chandrasekar, the Graduate Research Assistant (GRA). He provided us access to the participants' answers to survey questions. As more and more surveys are filled out by users the team will continue to refine their algorithm to accommodate that extra data. As a further step we will work towards a hybrid recommender, which will incorporate not only content, but also collaborative-based recommending.

2.0 User Manual

2.1 Objectives

Addiction recovery is a major issue in the United States and there is not an easy path to recovery. Friendica is an online recovery community created to investigate the effect of social networks or people recovering from addiction on the effectiveness of their recovery. The idea behind the Social Interactome data analysis for recovery friend recommendation is that within the social community of Friendica having beneficial friends is a crucial step to a successful recovery. The recovery friend recommender will gather data from multiple sources in order to provide users with recommended friends that the program deems beneficial to the user. The suggested friends will then hopefully help the user to get through their recovery process.

2.2 Business Process

Currently Friendica allows any users to become friends with anyone they want, but with the updated Social Interactome friend recommender users will have easier access to more beneficial friends. This will make the program more useful and thus encourage more people to use and become a part of the Friendica community. With this addition the customer will get a better experience from Friendica and hopefully become more successful in his/her recovery process.

Our client is the Social Interactome team that has been working on this project. Our point of contact with our client is Prashant Chandrasekar. They will be providing us any access that we need to their database as well as their sandbox environment for testing our code.

2.3 User Roles and Responsibilities

Users of this system are both people in recovery as well as professionals in the fields of recovery, human studies, and psychology. The people in recovery use this system directly by being a part of the Friendica community and having access to friend suggestions. They will then use the friend suggestions to meet and communicate with new people who are more likely to help them through their process of recovery. Other users of this system include professionals in fields

involving recovery and similar studies. The system will be able to provide information on how the system calculates how to recommend recovery buddies and from this an analysis on the effectiveness on different parameters for buddies can be made.

Users in Recovery - These users will be accessing the Friendica network to connect with their buddies for help each other's recovery process. While interacting with the webpage all of their actions will be recorded within a database. These actions will include reading recovery stories, commenting on other participant's posts, messaging other users, attending video meetings, and completing required and optional assessments. The friend recommender will use the stored information from the database to determine what other users would make good buddies by checking for similarities. The similarities will be weighted differently based on analysis from the first experiment and live data. Once the algorithm has determined who would be the best person to recommend this information will be provided to the participant. The user would then decide if they want to connect with their recommended buddy or not.

Professionals - These users won't be directly interacting with the friend recommender as they aren't in recovery. Professionals will be able to access all the information from the database and algorithm. Their analysis of a user's recovery will help to refine the recommender system. The professionals will be able to see why a buddy was recommended and if the connection was effective in either buddies recovery. Using this the system could be reevaluated to decide which factors should have a higher weight.

2.4 Interactions with Other Systems

The Social Interactome friend recommender builds off of the Social Interactome network. The Social Interactome is a social network to connect people going through addiction recovery. To gather effective data for friend recommendations participants in the Social Interactome network will first fill out a survey to collect initial information, then participants will go through the first 12 weeks of the study with a set of six given friends without the ability to add new ones. On the 12th week they will be able to add new friends and the recommender will give suggestions on who the system thinks will be most beneficial to the participants. Throughout the first 12 weeks all the data on what the participants do will be stored in a database and used to create the friend recommendations. The data

will be stored in a MySQL and the client will either give access to the database or provide the data in a CSV file.

The recommender will use an algorithm that is based on information from a previously run experiment to determine the best buddies possible. This information won't be used for the recommendations but will be a point of reference for the design.

2.5 Production Rollout Considerations

After the first 12 weeks are conducted the new recommender will use the gathered data of how the participants interacted with their given friends, what the participants did, and how successful they were in their recovery process to give friend suggestions. A major case, to see whether a friend suggestion is a success or not, is if the user actually friend-requests the suggested friend. On the other hand, if the user requests friends that are not high on our recommendation list then our algorithm has made a false negative error. If the participant sends more friend requests with the recommender system implemented and he or she has more interactions with their friends than without the system then it can be considered a success case. Based on whom the participants add as friends and how much they communicate with said friends will determine the overall successfulness of the recommender.

2.6 Terminology

Recovery Buddy - The only people that participants are able to interact with during their participation in the study. Anything posted, personal information, and activities are viewable by recovery buddies and vice versa.

Recommender - The system that provides participants with suggestions for new friends. The recommender will use an algorithm that is developed from the first experiment's data and data from the ongoing experiment.

Social Interactome - an online recovery community created with the purpose of investigating social networks as a means for therapy for people recovering from addiction

Friendica - The network that the recommender is being introduced into. The network allows users to interact in order to assist each other's recovery and prevent relapse. Friendica stores the information of every interaction a user has with the system within a database. The network will provide live data from its database to the recommender system in order to create buddy recommendations.

Similarity Matrix - A matrix with values that correspond to the likeliness that one person is similar to another person. These values are made using an algorithm implemented in Python.

2.7 Equipment Used

The application will be developed using personal laptops.

2.8 Statement of Functionality

The Social Interactome recommender will gather data from prior weeks of the study in order to provide suggestions for beneficial recovery buddies to participants. Variables that will be accounted for in the algorithm that we will design for selecting recovery buddies may include, but are not limited to:

- Readings
- Viewings
- Things liked
- Assessments
- Interactions with other peoples
- Ethnicity
- Gender
- Age
- Substance recovering from
- Job background
- Living conditions
- Income
- Location
- Occupation
- Urban versus Suburban lifestyle
- Hobbies

Then from these different variables the recommender will formulate a dynamic algorithm in order to recommend participants with recovery buddies. As the information get updated and the interactions between suggested recovery buddies is analyzed, the algorithm will change in order to better provide participants with recovery buddies.

2.9 Scope

During the initial 12 weeks of the study the data gathered from the interactions between participants and their recovery buddies will be used to formulate an algorithm for suggesting recovery buddies to participants. As more data is gathered during the 12 week process, the algorithm will be adjusted. As the 13th week begins participants will receive their recovery buddy suggestions. Continuing, more information on the interaction between the participant and their suggested recovery buddy will be used to adjust or validate parts of the algorithm.

2.10 Concurrency

The recovery buddy recommendations must stay concurrent with the data of the Social Interactome in order to not recommend people who are already buddies of the participant.

3.0 Developer Manual

3.1 Design Overview

To start our recommender project we have to make a design based on the entrance surveys that all participants must take. From that information we will make a recommender for the very start of the first experiment's second iteration. While this will not be used at that time it is a good place to start for our initial draft of a recommender. Our recommender will then be improved as more surveys are filled out by the participants. However, since a number of the participants may not fill out any surveys we need our recommender to work solely based off of the initial survey.

The initial data will be provided to us as a comma separated value (CSV) file. Using the computer programming language of Python we will parse through the CSV file, and fix any irregularities that are found in the data. Afterwards, we will apply the algorithm provided by the Social Interactome team to the data to find the similarity of each of the 128 people in the homophily section of the experiment. Based on that similarity we will fill out a matrix (Figure 3.1) with each person's similarity score. Using that similarity table when the Social Interactome team are ready to use our recommender they will have a table of every person's score, and from there will be able to decide how best to have people distributed as friends. Our code can be found at https://github.com/baehre/SIRecommender.

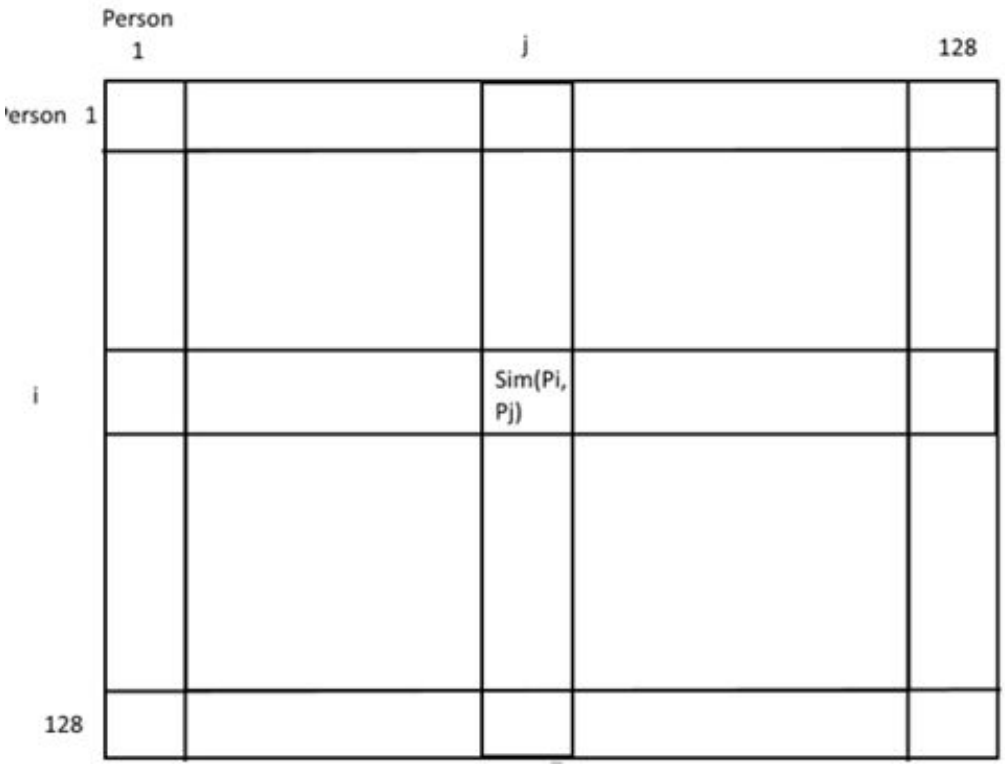

Figure 3.1 - Similarity matrix which will hold recommender output

3.2 Homophily

Our recommender has to be able to suggest people based solely off their initial survey. This way of recommending friends based on their similarities is referred to as homophily, and is one of the two ways that the second experiment will be designed (the second way being random similar to Figure 3.2). It is similar to the lattice structure (Figure 3.2) which is being used in the first experiment. What factors decide homophily will be an important part in our recommender. There can be many different categories that can go into associating participants. Socio-economic status, their addiction, and many other factors may go into the final algorithm for deciding these people's closest matches.

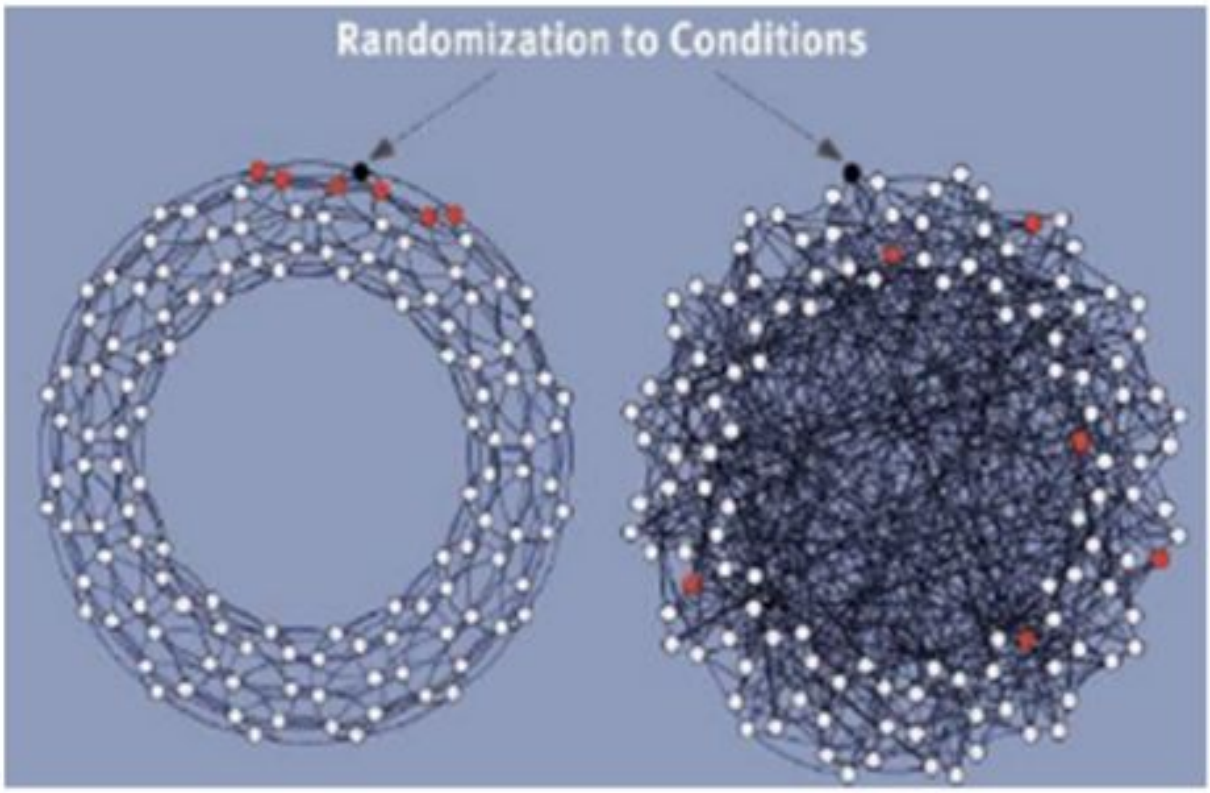

Figure 3.2 - Example of Lattice vs. Small world friend organization

3.3 Parsing

While some of the data that comes in from the survey may be usable immediately some of the data that is provided to us may need to be fixed to match the rest of the data. Any subjective questions asked, or anything that can be written in a text field may need to be standardized to match the rest of the data in that column. While going through the .csv file that will be our first task. After going through the data we then have to create a similarity score for every person with every other person. This will result in 8001 different similarity values. While not particularly efficient this should not cause any problems with our code.

3.4 Python

The choice of the Python programming language was simple. Python provides a number of modules that allow easy access to SQL databases and CSV files. The other option would be the statistics language R, but R is not a particularly well known language, and our team has a better handle on Python as an overall

language semantically and syntactically. R is also not particularly good at text processing as compared to Python. Usage of numpy will also allow us to make matrices to hold all the similarity values computed by our algorithm. This will be the main data structure that we use in our code. Numpy(3) also provides an easy way to do any calculations that may be necessary for our algorithm to work.

3.5 Recommender Versions

There are three basic types of recommenders: content based, collaborative and hybrid. We will be focused on content based recommenders since that will be the easiest way to get a working algorithm up and running without having to include any relationships between people in the experiment. An example of a content based recommender would be to make a recommender based on solely information such as what we will be getting from user surveys. A collaborative recommender would be using information about user's friends to make a recommender. It also allows us to easily add more specifications as more surveys are filled out, which provide us more content. While a collaboration based recommender would allow us to better understand relationships between people, and which friends of friends would be good choices to recommend to each other. The final option, hybrid, while quite possibly the best option, is not something that can be easily implemented in the time that we have to finish our project.

3.6 Extendability

While our recommender may not need to implement many different ways of recommending friends by collaborative or hybrid recommending methods, we do need to make our recommender scalable for any other changes that need to made for extra information coming in from surveys or text based information provided to us by Abigail Bartolome. This also allows for the Social Interactome team to use our code even after our part of the project is finished.

3.7 Implementation Strategy

The first step of the implementation of the SI Recommender will be to gather the starting data on all the participants. The source of our beginning data will be a questionnaire given to the users. Initially before participants join Friendica, they will be prompted with a survey to record initial data such as age, gender, location, addictions, etc. This information will be stored in a CSV which will be provided to

us. From the data we will extract specific characteristics for each participant such as their age, their area of living, their gender, what substances they are recovering from, etc. and create a similarity matrix. The similarity matrix will have participants in both the rows and columns, and their intersections will be the number of shared characteristics. From the values of the matrix we can then generate a friend suggestion list for each participants by sorting the numbers in the matrix from greatest to lowest, greatest meaning the participants has the most things in common with the specific person.

3.8 Tools and Frameworks

The SI Recommender will be implemented using the Python language. Inside our recommender we will be using NLTK (the Natural Language Toolkit)(4) for processing text. We will also be using the package Numpy(3) for a matrix to hold the output of our algorithm. We will also be using the sqlite3 API(6) to access the database holding the information of all the user's survey responses.

3.9 Implementation Methodology

To develop the SI Recommender algorithm we will first analyze the surveys that new Friendica users take. This survey includes things such as demographic information, their primary addiction, secondary addiction and other information. The different sections of the survey will be analyzed to determine which are likely to be most important for recommending that users connect. Five factors(age, gender, primary addiction, education level, and household income) will be used by our algorithm to generate recommendations. These five factors are being used as they were recommended by the GRA.

3.10 Survey Result Parsing:

Before users join the Friendica network, they are required to take a survey to get background information on each of the users. All the answers are then stored in a database in accordance to the user's ID number. The SI Recommender will then take the data in the form of a CSV file. The CSV file contains a line of headers for the question from the survey along with the survey results for each participant.

Each participant's results are one line of the file starting with their ID number and followed by each of their question responses.

	A	L	M	N	X	Y	
1	id	Month of	Year of bi	Gender:	Education: []	Household Incom... []	What was your p... []
2	4179	March	1964	Female	Some college	$50,000 - $69,999	Alcohol (liquor, beer, wine)
3	5353	July	1983	Female	Bachelors degree	$30,000 - $49,999	Opioids (heroin, opium, mo

Table 3.3: Sample of participant survey results

Figure 3.3 above shows the format of the CSV file containing the results from the survey given to the participants. The first process of the SI Recommender is to parse the data from the given CSV file and then create a simpler, easier to read CSV file. **Figure 3.4** below shows the output CSV file after parsing.

	A	B	C	D	E
1	id	Age	Male	Female	Other
2	4179	52	0	1	0
3	5353	32	0	1	0

Table 3.4: Output CSV file from data parsing

The parsed data will either be in the form of a numerical value or a bit value of ‘1’ or ‘0’ for true or false. Numerical values will be used for information such as ID number and age. For the ID number, it will be directly translated from one CSV file to the next. For the age, the parser will take the input of two strings, the value for “Month of Birth” and the value for “Year of Birth,”compare those values to today’s date, and return one numerical value for the user’s age in years.

Bit values will be used for both yes or no questions or categorical questions. Categorical questions are questions that provide a drop box, containing the answer possibilities, with the question. For these questions the parsed data will provide one column per answer choice initially assigned ‘0’ and the column with the answer selected will be a ‘1’. For example, if the answer options were alcohol, drugs, and shopping and the user choose shopping the output file would display [0, 0, 1] for this question. For initial testing we are only using a few of the questions to recommend friends. These include gender, addiction, educational level, and income. The code uses a list of the different addictions as well as a HashMap of the addictions to their indices in the list. A HashMap is used to speed up the lookup step in the algorithm.

This code style and data structures are used for all the other categorical questions. Some of the questions had an ‘other’ option as well as a second survey question to fill out their answer to the question. These answers are uncommon outliers and are grouped together in an ‘Other’ category in order to simplify the coding and friend recommendation processes.

3.11 Similarity Matrix Creation:

The similarity matrix will have participants in both the row and column, and their intersection will have the number of shared characteristics. From the values of the matrix we can then generate a friend suggestion list for each participant by sorting the numbers in the matrix from greatest to lowest, greatest meaning the participant has the most things in common with the specific person.

Once the data is parsed it is interpreted to create a Similarity Matrix storing the similarity scores for participants. The matrix is output as a CSV file to be used by Friendica’s recommender functionality. The matrix creator uses the output from the parser seen in **Figure 3.4** as its input. The input is then parsed to generate a two dimensional array storing the user answers for age, gender, primary addiction, education level, and primary addiction.

The array used to easily search from specific user information by index instead of navigating through the CSV file using function calls. This vastly reduces the function calls required as each user’s information is compared to the other 127 users. The comparison information is stored within another two dimensional array that is used for generating the output matrix. The prototype currently uses a nested for loop to generate the similarity scores and store them within the output array. These values are rounded to two decimal point to reduce visual clutter in the output CSV.

The nested loops use a function defined in the script to compare the two users and return a similarity score for them. The function has two parameters which are arrays representing each user's information. The values in the array will be age, gender, primary addiction, education level, and household income. The values for each user are compared to determine how similar they are. The comparisons check if the users have the same value for gender, addiction, education and household income and check if their values for age are within five years. The results of these comparisons are summed to determine a similarity score for the

two users. The score ranges from zero to one. A score of zero indicates that the users are completely different and a score of one indicates that all their values were the same.

After the similarity scores are computed and stored the data is written to the file ‘matrix.csv’. **Table 3.5** shows the format for the similarity matrix. The user IDs are stored within the first column and row. The similarity scores are stored within the cells corresponding to the two users that were compared.

id	user 1 ID	user 2 ID	user 3 ID
user 1 ID	similarity score	similarity score	similarity score
user 2 ID	similarity score	similarity score	similarity score
user 3 ID	similarity score	similarity score	similarity score

Table 3.5: Format for the similarity matrix

3.12 Max Flow:

After the similarity matrix is created, there is one main issue that needed to be addressed. By merely using the similarity matrix scores to recommend friends, it is possible for someone to be a ‘recommended friend’ to any number of people while someone else could only be a ‘recommended friend’ for one person. In order to prevent this we decided that we would put an exact number on how many people a person should be recommended to, in this case we put the number at six. In order to enforce this cap and still recommend friends with high similarity scores we created a max flow algorithm. Max flow is a part of optimization theory where you find the maximum flow from a single source to a single sink. To apply this theorem to our problem we created a single source node that branches out to the number of participants, in this case 128. These participant nodes then link to a node for every other participant, then those participant nodes link to the final sink node.

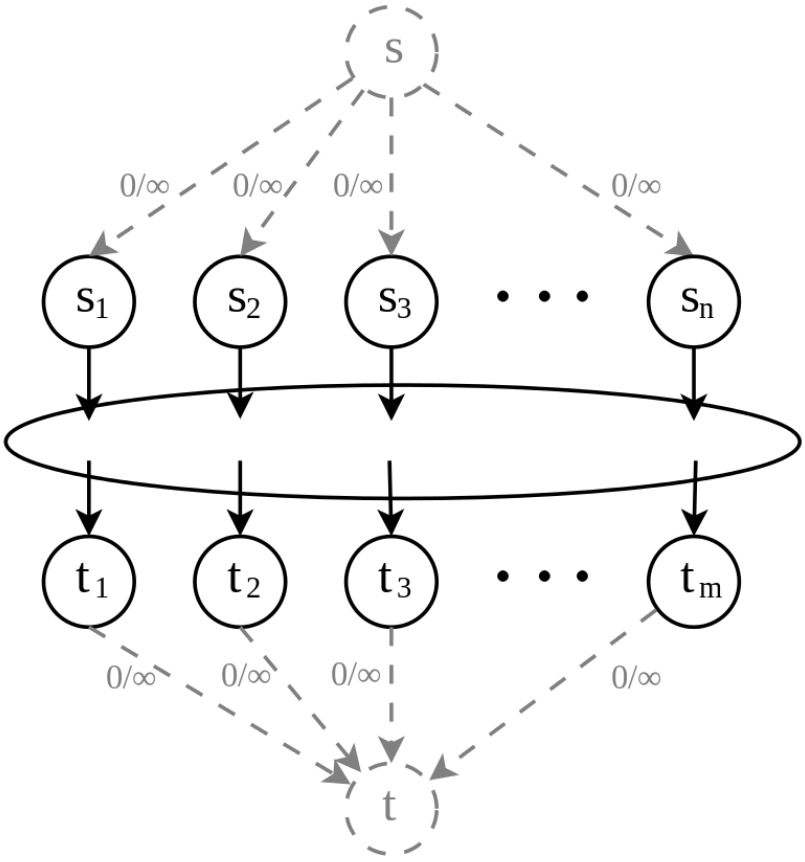

Figure 3.6: Max flow graph of participants to participants, adapted from [ref-3]

Figure 3.6 shows the graph for the max flow algorithm, ‘s’ is the source that links to the participants (s_1 to s_n) and those participants link to other participants (t_1 to t_m) which then link to the sink ‘t’. The links from the source to the participants and participants to the sink have a fixed value while the links from participants to participants use the value from the similarity scores for their edge. The algorithm then uses maximum bipartite matching, allowing only six edges from each participant to other participants, to calculate the max possible value for the entire graph. The max flow value is virtually the sum of the six similarity scores for each of the 128 participants and their recommended friends. The algorithm then updates the list of recommended friends for each participant that gives the max flow.

3.13 Testing:

The goal of our project was to provide a recommender to the Social Interactome team. To test that what code we provided was correct we will test it with two different methods. The first will be to use our data on Steam, an online gaming

network, data to see if the code we have written is versatile. The other is to check especially similar scores between participants by looking at the values that were used to calculate that score.

Our code currently takes into account five factors from each person in the Social Interactome. These five factors are age, gender, addiction, education, and household income. These factors are used as the GRA recommended using them to determine the similarity between two users. Based on these factors we wrote our algorithms.

The shortest distance algorithm as mentioned above gets the difference of each value (with the exception of age) and then giving each value an equal weight and returning a value between zero and one for each person. A value of one indicates that the users being compared are either the same user or have the same values for all five factors. A smaller return value indicates a larger difference between the users being compared. The other algorithm compares each person and returns the euclidean distance between each person. The euclidean distance is calculated by for each value taking the difference squaring it and adding it to all the other squared differences and then taking the square root of that sum. For our purposes at this point that gives us a 256x256 matrix with the top left to the bottom right diagonal of zero. The rest of the values generally range between one and three. A distance of zero indicates that the users being compared are either the same user or have the same values for each factor. Larger values indicate a greater distance meaning the people are more different the larger the distance becomes. The age was an outlier in that if it was included without any change it would skew the data. It instead was changed so that if the age was within five years of each other then it was considered a value of one and otherwise was given a value of zero.

The algorithms are accurate if they produce recommendations that result in people with the most similarities being recommended. We want to recommend people that are most similar as the social interactome experiment is testing to see if organizing individuals in recovery based on homophily will result in continued recovery. Homophily is “The tendency of individuals to interact with others that are characteristically similar”(Welcome to the Social Interactome). So our algorithm should correctly recommend people who are most similar.

Steam

Our code will also be evaluated on its extensibility. We will check our algorithm using data from another source to show that friends are recommended properly. We will create a Python script to crawl the data from a Steam friends list and check the results of the algorithm. Since Steam users don't have any addictions in their profile we will instead use the list of games that they own as a replacement for the addiction comparison within our algorithm. After adjusting the algorithm to account for owned games instead of addiction we will inspect the results to determine if users were recommended with similar owned games.

We tested the Shortest Distance Algorithm using data gathered from 50 Steam, a gaming software distribution platform, profiles to show that it is extensible and will give recommendations based on similarities. For this test instead of using addiction as a comparison factor it was replaced with games owned. There were eight games(Enter the Gungeon, Counter Strike, Rocket League, Super Meat Boy, Garry's Mod, Hotline Miami, Team Fortress 2, and Dota 2) that were compared. The comparison was based on if the user owned a game or not. These games were chosen as they were some of the most recently played games by the users. Steam profiles don't require including a user's age, gender, education, and salary so those factors were irrelevant to this test.

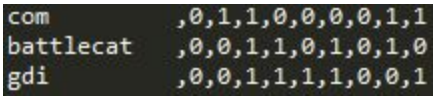

Figure 3.7: Sample input for Steam data

The test results confirm that users with more differences are correctly given a higher score, showing a greater distance between the two, than users with less differences. **Figure 3.7** shows a sample of three users and their values for each of the eight games as well as a zero for the age field. The differences amongst the three users should result in the Shortest Distance Algorithm giving the greatest distance between users one and three and the smallest distance between users two and three. After running the Shortest Distance Algorithm the distances were 2 for users one and two, 2.24 for users one and three, and 1.73 for users two and three. So our assumption was correct and the Shortest Distance Algorithm is giving greater distances to users with more differences. The results also show that the algorithm can be used for different forms of input to show similarities.

At the suggestion of Dr. Franck we tested the Euclidean distance algorithm by taking people with low scores, and checking their values to ensure that the low distance outputted made sense. If a large number of the values matched then the Euclidean distance computed made sense. We tested 10 random low scores to ensure that our algorithm was sound.

Results

Our testing has shown us that the algorithms that we wrote are indeed sound. They will be able to be implemented easily into the PHP code by the developers of the project, and that the code is extendable enough so that any changes that may need to be made in the future are possible.

4.0 Lessons Learned

Responsibilities

Andrew Baehre - Write algorithm and test algorithm to provide multiple options for recommending friends, general writing for the write up of each section

Trevor Kinaman - Parse CSV file to provide information to Colin Gray, general testing, general writing for the write up of each section

Colin Gray - Algorithm analysis, programmatically create similarity tables to provide to Andrew Baehre, general writing for the write up of each section

Prashant Chandrasekar - Provide access to Sandbox development server, access to user data, and general guidance in the project

Time Table - Planned Dates

2/14-2/20
Andrew Baehre - Work on Requirements and Design report

Trevor Kinaman - Work on Requirements and Design report

Colin Gray - Work on Requirements and Design report

2/21-2/27

Andrew Baehre - Start working on algorithm analysis

Trevor Kinaman - Work on parsing the CSV file

Colin Gray - Start working on algorithm analysis

2/28-3/5

Andrew Baehre - Start writing algorithm, work on implementation report

Trevor Kinaman - Standardize all data for output CSV file

Colin Gray - Start writing algorithm

3/6-3/12

Andrew Baehre - discuss which factors would be best to be used in the algorithms

Trevor Kinaman - discuss which factors would be best to be used in the algorithms

Colin Gray - discuss which factors would be best to be used in the algorithms

3/13-3/19

Andrew Baehre - take extra data and make sure algorithm still works with it

Trevor Kinaman - provide extra data to Colin Gray and Andrew Baehre

Colin Gray - update algorithm to accommodate new data

3/20-3/26

Andrew Baehre - Improve algorithm

Trevor Kinaman - Work on prototype and refinement report

Colin Gray - Improve algorithm

3/27 - 4/2 -

Andrew Baehre - Write prototype and refinement report, work on other algorithm

Trevor Kinaman - Properly extract data from CSV file provided by Prashant Chandrasekar, and provide to Colin Gray and Andrew Baehre

Colin Gray - Take extracted data from Trevor, and apply an algorithm to make a similarity table

4/3 - 4/9 -
Andrew Baehre - apply improved algorithm to make a similarity table

Trevor Kinaman - Provide more data from the CSV to Colin to further improve algorithm used

Colin Gray - Take extracted data from Trevor, and apply an improved algorithm to make a similarity table

4/10 - 4/16 -
Andrew Baehre - work on write up, compare similarity tables

Trevor Kinaman - Work on write up making any corrections necessary

Colin Gray - Help Andrew Baehre with comparison of similarity tables

4/17 - 4/23 -
Andrew Baehre - Finish algorithm, write up tutorial on how to make code

Trevor Kinaman - Finish parsing Python code, work on final write up of the project

Colin Gray - Provide final similarity chart, finish write up with Trevor Kinaman, create script for testing using Steam data

4/24 - 4/26 -
Andrew Baehre - Work on final touches for write up and code

Trevor Kinaman - Work on final touches for write up and code

Colin Gray - Work on final touches for write up and code

Problems

At the beginning of our project we assumed that an algorithm for comparing users would be provided to us by the Social Interactome team. We also believed that we would have access to the PHP sandbox environment for implementation of our similarity matrix.

Solutions

To work around our problems we coded two different algorithms to come up with similarity scores for each member of the study. We provided a CSV and python code to Prashant to provide to the coder in charge of the implementation of the recommender.

Future Work

In the future we would like to improve our algorithm, as well as take into account more variables. We would like to update the parsing of the data to better handle the ‘other’ categories, work on the max flow algorithm for assigning the friends, and some more testing to ensure the quality of our results. We would also like to work with Abigail Bartolome, an undergraduate research assistant working on parsing the communication between participants. We would like to include her data in our algorithm.

5.0 Conclusion

We will be providing a scalable recommender based on initial survey results in the next few weeks. Afterwards we will work on making a table of all the results of our algorithms between each person. As more information comes in we will update the table, and algorithm to give a better picture of which people would be best as recovery friends. When week 12 of the experiment approaches we will have the most up to date table for use of the Social Interactome team in recommending friends to people once the participants are allowed to choose their friends freely.

6.0 Acknowledgements

We would like to thank our client, Prashant Chandrasekar, for his guidance and help in completing our work. We would also like to thank Dr. Fox for his help in editing our report. Finally, we would like to thank Dr. Franck for his help in testing our code.

7.0 References

- 1. ChengXiang Zhai and Sean Massung. 2016.
  Text Information Systems: An Introduction to Information Retrieval and Text Mining. ACM Books and Morgan & Claypool Publishers.
- 2. Edward A. Fox. 2016.
  The Social Interactome of Recovery: Social Media as Therapy Development. https://www.cs.vt.edu/node/7439. Accessed: 4 April 2016
- 3. Digparty. 2016. Maximum flow problem.
  http://www.digplanet.com/wiki/Maximum_flow_problem. Accessed: 4 April 2016