# **DS2 ProjectB: Implementing A Recommender System for Fika**

December 8, 2018

Diana Gao 216301285

Ramnath Kumar. 216060907

Vivek Kumar Singh 216301707

Wanwen Huang 216258402

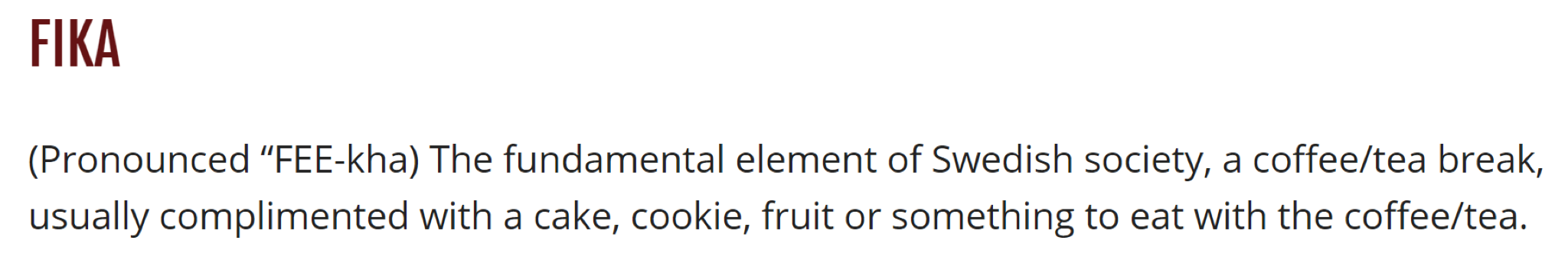
Robert Feldman 216696700

# **Client background: Fika**

We are the Dataxon Consultant Group and we have been retained by Fika to advise them on their growing social network.

Fika is a social network where users follow one another and also declare their interests. If a user follows a second user, it does not mean that the second user follows the user back.

Fika is named for the Swedish custom of having a chat over coffee and a snack, similar to the English ‘tea time’. The name Fika (pronounced ‘FEE-ka’) was chosen to express friendliness and community.



# Objectives: Profit by pushing interconnectivity

The Fika team hopes to expand the social network and create interconnectivity through ‘Follow Recommendations’. Because the spirit of Fika is about shared interests, Follow Recommendations should not be made when users do not have interests in common.

Further, the recommendation system should run smoothly even with high volumes of data.

To increase interconnectivity by making intelligent tailored recommendations that make the user happy and engaged with the network, both to established users and to new users.

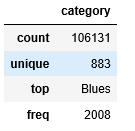
Following implementation, the next steps will be to evaluate the system in the context of Fika’s business model.

# **Part A**

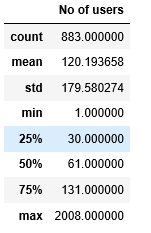
#### Overall explore of the dataset

There are 3,042 users currently using Fika. They have declared 883 different interests. There is a wide disparity between the most and least popular interests, and the most and least popular users. The distribution of the entire population of individuals is right-skewed. The top interest is blue and there are 2008 users following it. But the mean of the interest be followed is only 120. In total 883 different categories of interest only 239 interests are above the average number of users number.

Using pandas to rank and list aspects of the data, we can give more insights about the shape of the network and the popularity of various interests. See the ProjectB PartA.ipynb Jupyter Notebook for the corresponding code.

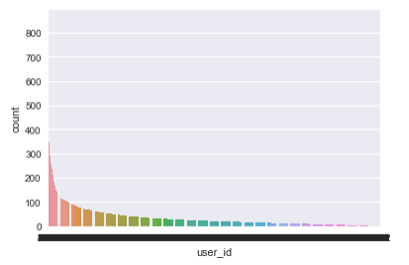






(Here let’s list the top five most-most-followed interests, BLUES, GUADELOUPE, etc.

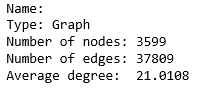
The histogram of declared interests do individual users have looks like below:



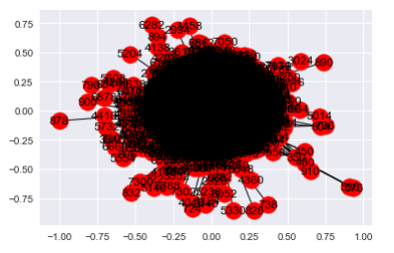
It is in the order of the frequency count of the number of interest.

#### Connection degree distribution

The information of our nodes features.

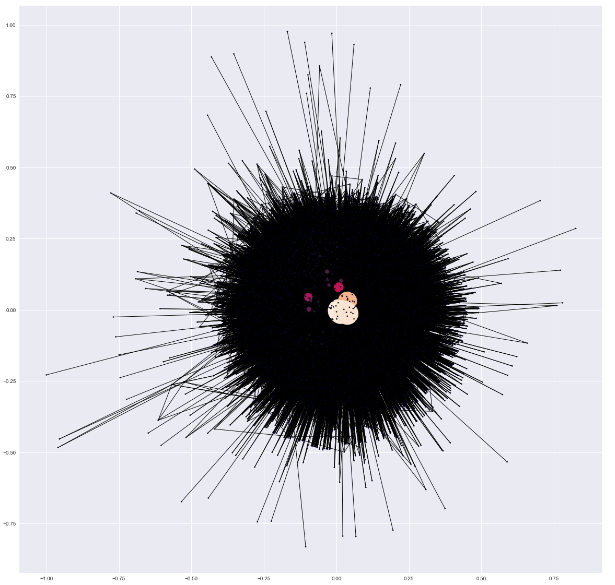


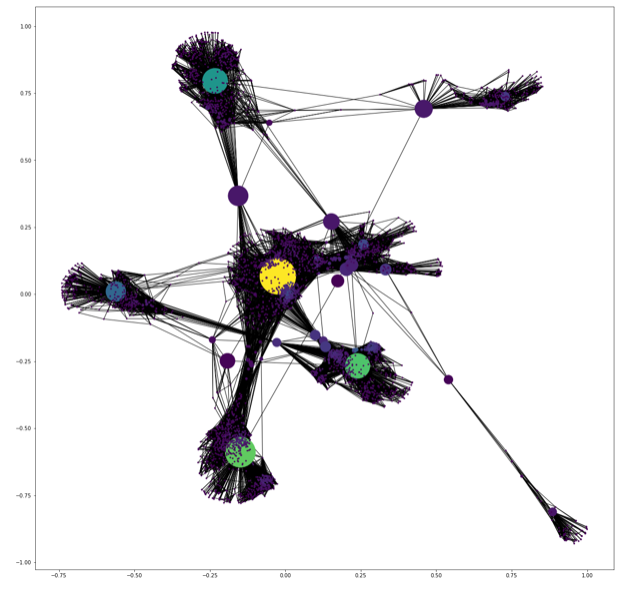
The visualization of our network.



Compare to studied social networks such as Facebook. Our network is way too centralized, as shown the first graph below, which is the basic plot of the network

. There are some popular people in the community and all the other users are all trying to follow them. The second graph below is the basic plot of the Facebook network, it looks more decentralized. The question be followed is how to create a healthy network and avoid over centralized community. We did put the weight to solve this problem in our recommender system.





## Comparison with Twitter

Our client’s social network is more like Twitter than Facebook. The fundamental difference between these two social networks is the type of the relationship between users. The Facebook friend relationship takes place between two personal profiles after mutual consent while Twitter follows is a one-sided relationship. Because Twitter relationships can be one-sided, there are ‘followers’, ‘followees’, and ‘mutual followings’ where two users follow each other. In contrast, the ‘friend’ relationship on Facebook is back-and-forth.

Fika is like Twitter because users follow users who do not follow them back. The ‘follows.csv’ dataset demonstrates this.

# **Part A Appendix A**

What kind of hardware did you run the analysis on?

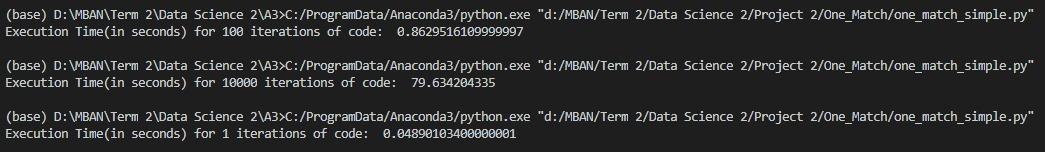
i7-8750H processor. 6 Cores. 16GB ram.

How long did it take?

Execution Time(in seconds) to run 1 iterations of code: 0.049467756999999946

Execution Time(in seconds) to run 100 iterations of code: 0.8629516109999997

Execution Time(in seconds) to run 10000 iterations of code: 79.634204335



The python script timeit\_check.py is included in the file. You'll have to change the file address to execute this.

What are your recommendations regarding the volume of data you could manage for further analysis? The ipython timeit magic command may become handy here.

- To manage a huge volume of data, we can do the following steps to increase the efficiency of our code:

a) Object Oriented Programming: It will increase our codes modularity, extensibility, and reusability. Faster development and reusability even in future projects.

b) Reduce memory footprint

# **Part B**

## Declaring a User score and defining a threshold for who is recommended

The output of our recommendation system will be a weighted score given to a potential connection between two users. The score represents both the similarity of two users’ interests and also the interconnectivity among the users they follow.

A single user will receive up to five ‘Follow Recommendations’ differentiated by the score. If the scores are below 0.166666. then the system will not make the recommendation. Instead it will prompt the user to declare more interests and follow more users. Then, the next time the system runs, that users’s score might cross the threshold.

The highest-scoring connection will be shown to the user as a ‘MegaBestFriend’ which indicates the high potential for shared interests. This way, users are encouraged to accept Follow Recommendations and the users have clear expectations for how compatible they will be with a recommended user.

|  |  |
| --- | --- |
| **Rank of Recommendation Score** | **Type of Follow Recommendation** |
| 1 | MegaBestFriend |
| 2 | BFF |
| 3 | Someone Exciting to Meet |
| 4 | Interesting New Person |
| 5 | Potential Connection |

## Making allowances for new users

A new user to Fika will begin with no followers, no followees, and no declared interests, and thus the system cannot make accurate Follow Recommendations.

The Connection Degree Distribution Histogram showed that of the 3042 users, fewer than one hundred had one or two degrees of connection. The vast majority had between 4 and 9 degrees. The average number of declared interests was 35. These numbers allow the system to make intelligent recommendations.

Therefore, we advise Fika to prompt new users to declare their top ten interests and to choose five users to follow. Using that initial data, the system can create Follow Recommendations for new users.

## **Model Algorithm**

Basically, the algorithm is that we can calculate the scores of **interest’s similarity** and scores of **social network similarity**, and then sum up both of them to get the final scores of each potential candidate. Lastly, we **rank the final scores** and select the top 5 users to recommend.

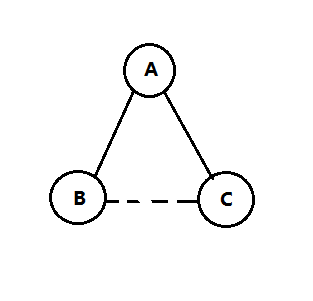
## **User Similarity**

### Second-degree users are the potential pool of Follow Recommendations

### Theory Background

Triadic closure is a concept in social network theory, first suggested by German sociologist Georg Simmel in the early 1900s. Triadic closure is the property among three nodes A, B, and C, such that if a strong tie exists between A-B and A-C, there is a weak or strong tie between B-C.

In the example below, B and C follow the user A, regardless of whether user A follows back, user B and user C are the **second**-**degree** **friends** of user A.

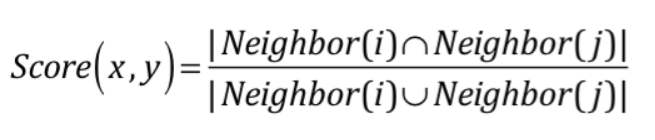


We want to use the recommender system to recommend user A to follow the other users, hopefully create more two-sided relationships like friend and increase connectivity among the users in the platform to bring more value to our client.

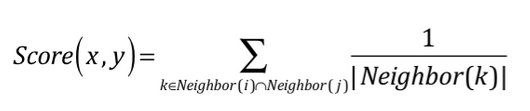
### Normalization: balances the users who have few interests and followees and those who have very many

The denominator of the equation is the sum of the following number of both user(i) and user(j). Put the total number of following into the consideration, which means the score of the user would punishment the users following too many users. The reason behind the logic is the recommendation would be less valuable if the recommend user already follow way a lot of friends.

The user similarity score is calculated by dividing the number of “common friends” that both user i and j follow by the number of friends that both users follow.



Another way of normalization is dividing the number of common friends that both users follow by the number of people who is following the “common friends” users.

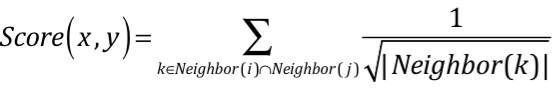


**two possible way of normalization**

If the common friend (user A) shared by user B and user C is followed by a lot of friends, it means that this friend (user A) is not "important" in establishing a potential connection between user B and user C. Put differently, probably user A is a celebrity. Following a celebrity can’t become a powerful predictor that two users have similar social network.

Hence, I add a **"weight" (normalization)** to the score above, which is the inverse of the number of friends that user A follows, so that the Score (x, y) can better represent the probability that two users (user B and user C) can become new friends.

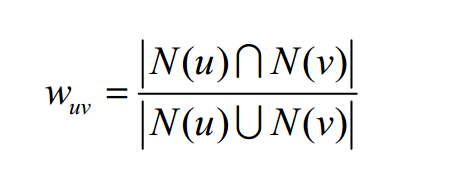
If the number of friends of common friend (user A) for two users (user B and user C) vary greatly from one common friend to another, then we can use square root in the normalization.



## **Interest Similarity**

### Jaccard Coefficient

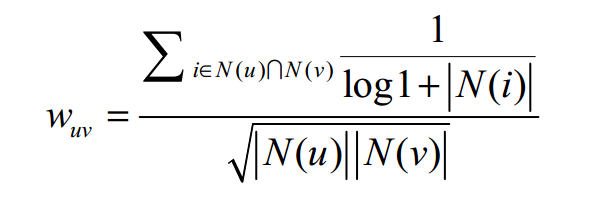
Similar idea to the normalization of the friend score. To concern the effect of the total number of interest to the interest score, the denominator of the equation is the sum of the interest’s number of both user(i) and user(j). The score of the interest would punish the users following too many interests. The reason behind the logic is the recommendation would be less valuable if the recommend user already follow way a lot of interests.



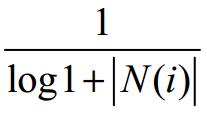
### Cosine Similarity

Consider the influence from the user following the top interest. Using the logistic to normalize the algorithm. N(i) represent the popular hot interest between the user(u) and user(v). The score punishes the user who follow the hot interest, because sometimes the hot interest does not represent the user’s real interest. We choose this formula to calculate the interest score.

This formula is first proposed by John S. Breese in *Empirical Analysis of Predictive Algorithms for Collaborative Filtering1*.

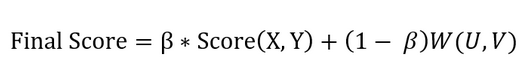


This formula has a punishment factor below to punish the impact of popular hot interest categories in the scores of interest’s similarity between two users.



## Final Score

The **final score** is the sum of scores of interest similarity score and social network similarity score.



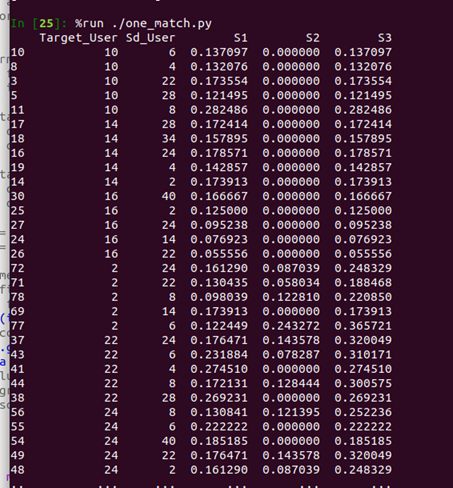
where represents the weight of social network score and represents the weight of interest score

To find out the optimal , we should try different values. Use the evaluation method below to identify the best . Or we can try different and get the result for our client to compare.

To simplify, we sum up the social network score and interests scores to calculate the final total scores. The client can adjust the weight by changing the . For example, if the client wants the recommendation system to be more inclined to social-network-based, he should increase the value of .

Lastly, I will rank the final score of each potential candidate and select the top 5 users with the highest scores to recommend.

The outcome on a sample test set is as below. In the output table, S1, S2 and S3 are social network scores, interest scores and final scores respectively.



**Outcome on Sample Dataset**

## Evaluation of the model find the best weights.

The model we are implementing has flexibility: it can give more weight to similarity based on shared interests and similarity based on overlapping followers and followees.

We recommend that Fika conduct A/B testing using different weights of the model to see which produces more interconnectivity and which creates positive user feedback and increased time on the network.

Because the brand identity of Fika is based around the Swedish social custom, it is likely that Fika management will prefer to give more weight to the cosine similarity score based on shared interests. Whatever management chooses, they have a flexible model that can be easily adjusted.

## Unix command line to run in the Ubuntu

C:\Users\Wanwen\AppData\Local\Temp\1544322174(1).png

The validation is run on **a small dataset (attached in the attachment),** because it takes some time to validate the algorithm on a large dataset. The average precision rate is about 41.2307%. The precision rate is very high in recommendation engine. Our recommendation algorithm has good performance.

# **Part C**

## Implementing a ranking system to increase model accuracy

In a recommendation system based on scores, a user indicates how much they like something. In the Fika user data, the users like each of their interest equally, and they like each of the users they follow equally.

However, in real life the users have preferences about their interests that are not captured in the data.

If Fika captured more data about user behaviour, e.g. how often and how long they interacted with certain users, they could establish a ‘ranking system’ more like that used to rank films.

The interests in the interests.csv file include many geographical locations. Some are related, e.g. Xinjiang and Shanghai are both related to China, but the data does not capture these relations. If one user likes Shanghai and another likes China, there could be a secondary score that captures the fact that they have related interests.

Recommendations on Data Dumps

The **follows** and **interests** datasets will only get larger as Fika expands beyond ~3000 users. Fika’s data will grow in **volume** as well as **velocity** because more user ‘nodes’ create several times more links between the nodes. Because the success of Fika will mean geometric data volume growth, we recommend splitting user data into ‘shards’ or segments and reiterating the recommendation periodically.

User data is constantly changing. Users add new interests. They follow new users and they gain new followers. Users also stop following users, and they stop caring about old interests.

The Follow Recommendations our system gives one day might be different one year later when a user’s interests change.

The %timeit command will help the recommendation system automatically adjust to changes in the network.

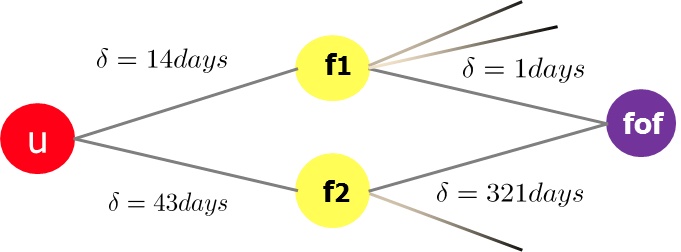
Timeit accepts the **timer** and **number** arguments, which indicate how much time elapses between repeated calls to the code as well as how many times the code is called.

If the code runs once every 24 hours, and it runs 365 times, then we can generate time-series data that shows the changing links in the social network, and the changing recommendations that are made.

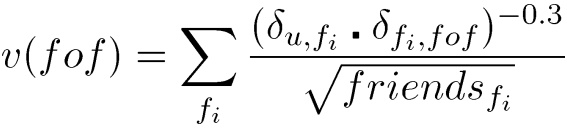
Based on the following reasons, the recommendation is that the data dumps can add new dimensions of time, interaction and the key words of users’ content in the social network.

Time Dimension

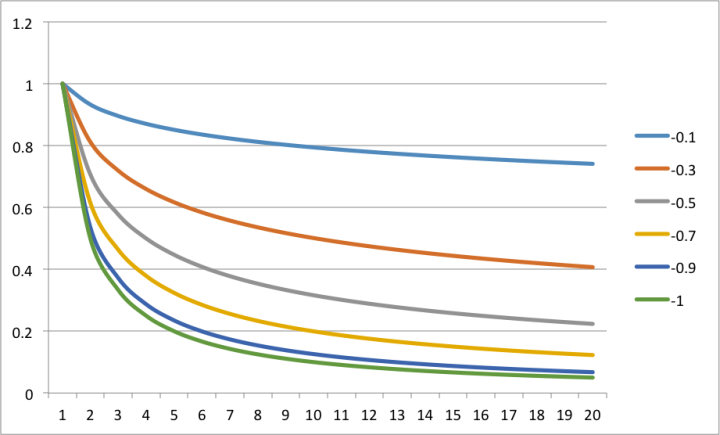
Adding a time dimension is based on the assumption that users in social network are more interested in new friends.



For example, both F1 and F2 are friends of user u. Compared to F2, F1 is new friend for user u. Hence, I assume that user u is more interested in F1.



Time dimension is added as a punishment factor in the social network score formula. δ u,*fi* is the length of time for which user u has added user *fi* as friends. δ *fi,fof* is the length of time for which user *fi* has added user *fof* as friends.



https://pic4.zhimg.com/80/dc8a04b9438e8d2a38ce5f096552812f_hd.png

Interaction Dimension

Social network tends to increase the interaction between different users. In social network, no matter how many friends a person follows, he can’t always interact with all of them. If two people have few interaction, they can’t be considered to be “friends”, even if they follow each other.

The interaction in the social network system can include messages, comments, likes, forward, retweets, etc. These interaction actions can, to some degree, reveal the “attraction” or “importance” of a user to another user.

The formula to calculate the interaction scores is as below. Interact (u, v) represents the number of interactions user u has on user v, and interact(u) represents the number of total interaction of user u to all the users in the social network.

Interact uv =

Key Words Dimension: Users with insufficient interests for the model

In a social network, users will not often label the interests they like, probably because the user is a newcomer, or because the user is not very active. One possible solution is that we can label interests for users by finding the key words from the user’s content (using TF-IDF). The user’s content can include the messages that he broadcasts or sends in the social network system. These key words can later work as interests’ vectors in calculating the interest similarity.

Evaluation Method

The evaluation methods to evaluate the top-k algorithm are Precision, Recall and F1-measure.

Recall =

Precision =

F1-measure =

In this project, we choose precision method to assess the performance of the recommendation algorithm on a sample dataset. We recommend that our client could use the F1-measure to evaluate the classifier’s operation in the context of the client’s business.