CS 484 Project Report

Recommendations for GitHub

# 

# RecGit

# Alejandra Vigil & Benjamin Waters

[www.github.com/the-ben-waters/gitrecd](http://www.github.com/the-ben-waters/gitrecd)

# 05/10/14

# Problem Description

GitHub is a web-based hosting service for friends, classmates, co-workers, and even complete strangers that is used to collaborate and share code in order to build various software development projects. The service provides a variety of collaborative features and serve over five million people with over 11.7 million repositories, making it the largest code host in the world since it’s initial launch in 2008.

One of the reasons that allows GitHub to be successful is their focus on providing social networking functionality which include feeds, followers, and wikis. GitHub allows users to discover and contribute to other repositories, yet with it’s vastly growing rate it has become difficult for users to find specific repositories that appeal to a user’s interest.

Currently GitHub only recommends the top starred repositories from the entire website and displays this information consistently to all their users. Since users are already able to “star” their favorite existing repositories on GitHub, my partner and I came up with a solution to make GitHub more personal with a mutual-follow recommendation system.

The emphasis on a personalized recommendation system is due to the impact of social networking. Social networking is a powerful feature since it not only opens doors for consumers but also increases marketing opportunities and a growth within the business.

# Related Work:

There are many websites that utilize recommendation systems, such as Amazon or Yelp, in order to create a more personalized profile for their users. It is a great marketing strategy and actually benefits the user so they are more pleased with the service.

The difference between most recommendation systems and the one we were able to implement is based off the followers feature. With the usage of mutual friends (i.e. “friends”) we are able to gather more information as opposed to restricting ourselves to only what a user has visited or favorited beforehand. Websites such as YouTube utilize a recommendation system that is similar to the one we are focusing one.

On the other hand of the spectrum, GitHub already uses a recommendation system however it is much more generic and not personalized to a single user. The purpose of our experiment was to extend the recommendation tool from GitHub and provide each user with personalized results as opposed to a generic one.

# Solution:

Our solution to GitHub’s lack of personalized recommendations is to create a recommendation system based off of a user’s mutual followers.

GitHub currently has two features, followers and favorites, that allows users to benefit from a personalized experience from the web-hosting service. With the use of followers, users already begin to create a web of connections between people that they may share similar interests with; and the favorite feature allows users to keep track of repositories they are highly interested in.

By utilizing both features and by implementing the K-Means algorithm and the Jaccard Coefficient, my partner and I were able to create a recommendation system based off a user’s list of friends (i.e. “mutual followers” - both users must be following each other).

We first gather a user’s information, primarily a user’s starred repositories, and compare them to their list of friends and their own list of starred repositories. Since GitHub indicates a starred repository as a binary value of 1 if it’s a starred repository, or a 0 if it is not, we are able to implement the the Jaccard Coefficient to create a binary vector to find the similarities between a user’s starred list and a user’s friend’s starred list. Afterwards, we use the K-means algorithm to cluster the results based off of the returned similarities value of the Jaccard Coefficient.

We later accrue a much better result of specified repositories that can be recommended to each GitHub user from this data.

# Experiment

**Data:**

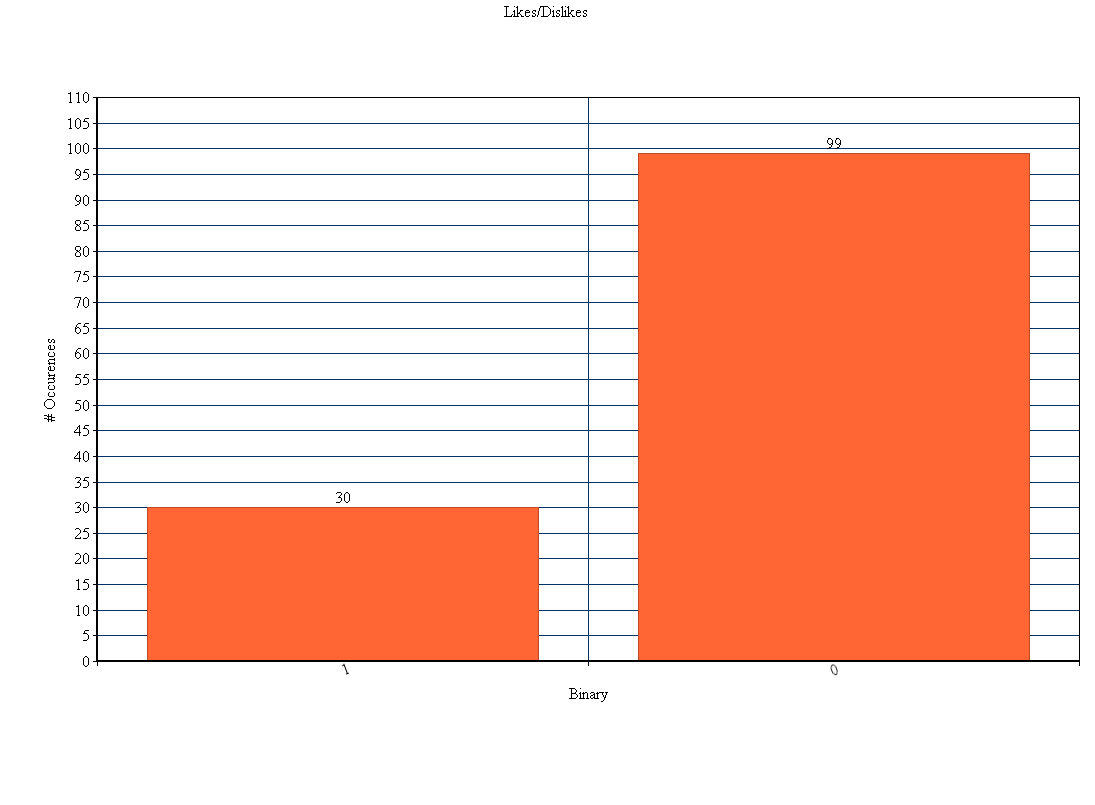
A user's GitHub “friends”, also known as users the use follows, starred repositories. We collected this data and the starred repositories of the user to fill our Redis server with binary keys. The Redis server would hold the key Like:Repository\_Name and Dislike:Repository\_Name. Recommendation Raccoon (RR) would then convert these keys into a binary array. Then RR would take the binary array and calculate the similarity between the user and their friends. Then would give each user a Jaccard score and return the top repository for that user.

**Experimental setup**

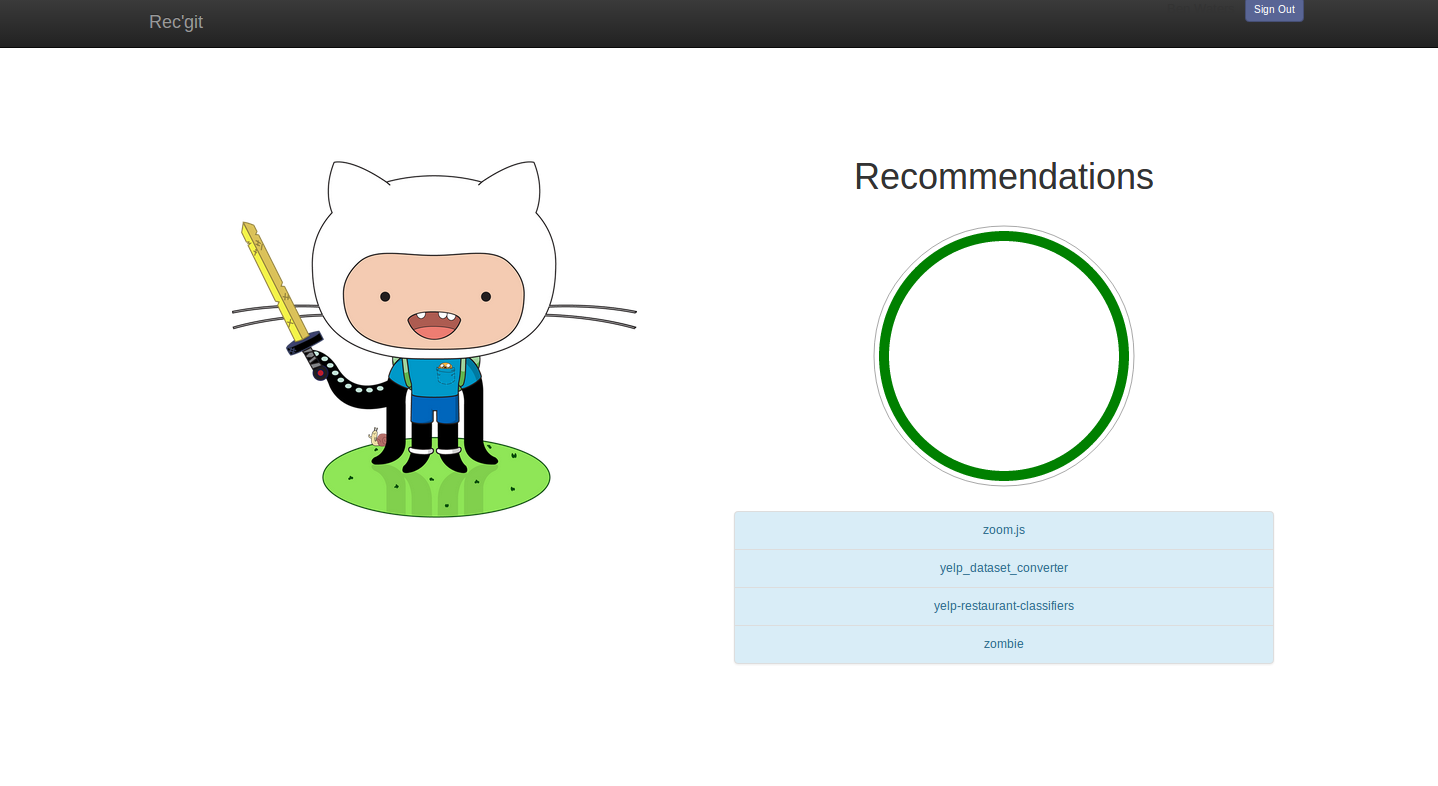
We configured Reccomendation Raccoon to k=5 for k-means. And we asked to return four recommendations. We took the user the-ben-waters (github.com/the-ben-waters) because he had a large number of friends and starred repositories. We then configured out MeteorJS webserver to run Recommendation Raccoon with the above users information and then presented the recommended repositories in the form of a website.

**Experimental results**

User the-ben-waters's likes and dislikes



What the-ben-waters sees



Recommendation Raccoon found that the four most recommended repositories for the user the-ben-waters was: zoom.js, yelp\_dataset\_converter, yelp\_restaurant\_classifiers, and zombie.

# Conclusion

After a month of turning in various project proposals, we were then finally able to use the last month of the semester to create a successful recommendation system for GitHub users. My partner and I used the K-Means algorithm and the Jaccard Coefficient method that were taught in class to implement a tool that was widely based on collecting and organizing data in such a fashion that could later be used to return successful results. In the future we would like to add a voting system so that we could use additional statistics based on feedback to determine more specific recommendations. We would also like to create our own recommendation engine of a modified version of Recommendation Raccoon to allow us to use different techniques such as SVM or J48 Decision trees to show what we learned from class and maybe see better resuls.

# Contributions of Each Member

The two members of the Team were Alejandra (Ale) Vigil and Benjamin (Ben) Waters. Each student was involved in the process of coming up with ideas for selecting a topic and submitting a proposal. There were multiple submissions of the proposal however due to obstacles that dealt with not having the resources to handle such large data sets, which led Ben to come up with the idea of utilizing GitHub and creating a recommendation tool for the large web-hosting service.

Ben led the experiment due to his higher knowledge of the GitHub service, however both team members participated in the entire execution of the experiment. Alejandra was able to contribute during the experiment by providing input in regards to which features to use during the experiment and what the results should look like in comparison to the current, generic recommendation system on GitHub. Ben developed the website and Alejandra tested it daily.

The presentation and the project report was led by Alejandra, however both team members contributed information that dealt with the experiment, experimental results, and the overall solution to creating a recommendation system for GitHub.