Building a Recommender System for Simtk

CS224W Final Project Report, Group 51

Rahul Pandey
Stanford University
rkpandey@stanford.edu

Narek Tovmasyan
Stanford University
ntarmen1@stanford.edu

December 10, 2013

Abstract

Simtk.org is a website used by thousands of biomedical researchers around the world to host source code, download projects, and share resources. As a medium-sized social network, Simtk does not leverage its social structure to create engagement, nor does it utilize its network to improve the project navigation. Upon agreement with the site maintainers and developers, we analyzed this social network, and implemented a item-item recommender system for projects on Simtk. In this paper, we analyze the user behavior on the site, implement a project recommender system, and analyze the results and qualities of our system. We found that a small group of projects commanded the majority of user pageviews, but we still built an effective recommender system using log data to identify similar projects and optimize project navigation.

1 Introduction

Simtk.org is a site used by biomedical researchers who want to model physics-based simulation of biological structures [1]. The site was developed in the early 2000s by Stanford after a grant from the National Institute of Health (NIH). The site has more than 29000 registered members who use the site to disseminate their software, datasets, or tutorials to a like-minded community [2]. Most of the functionality of the site is tied to a certain project; users have the ability to set up downloads, list site maintainers, or update the wiki associated with a project.

One of the dominant uses of Simtk is to identify and download software that can help solve the users problem (generally related to physics simulations software). The user comes to the site with a well-defined problem but is unsure what solutions, if any, exist already. Given that there are 716 projects hosted on the site, it is infeasible to know the utility and use case of each one [2]. Simtk can be analyzed from the perspective where projects and users are two distinct node types in the social network. First we attempt to answer various questions about the health of the site regarding interaction between these nodes: What is the distribution of traffic and downloads on the various projects? How do new users find projects suitable for their work?

Since the site was built more than a decade ago, the majority of the site consists of static content. There are no project suggestions based on user behavior, and navigation of projects

occurs through a crude hierarchy created from project keywords. Therefore, the goal of this paper is to build a recommender system for Simtk which allows effective browsing of similar projects, for some reasonable interpretation of similarity. Project similarity means showing a project in the same topic area, with similar keywords, or which uses a similar dataset. User similarity is a form of collaborative filtering: showing projects which other users also viewed at that point (similar to how Amazon shows *Users who bought this product also bought:*). Our goal is to provide high quality recommendations for projects on the site.

2 Prior Work

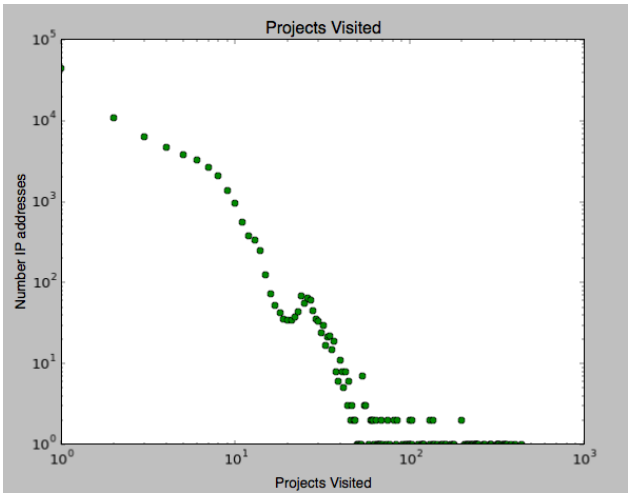
The task of building a recommender system has been well studied in recent decades following the rise of recommendations for e-commerce stores. In the canonical recommender system, the task is to recommend various items to users based on an utility matrix which should capture some notion of user preference. In the context of Simtk, the items are each of the biomedical research projects. Recommender systems allow individuals in the community to more effectively identify interesting content from a set of choices that would be too large to sort through manually [3].

Leskovec et. al. analyze item recommendations using information cascade subgraphs [3]. An information cascade is a phenomenon wherein a new action is performed due to social influence. The paper examines an e-commerce store where the purchaser has the ability to recommend the product to friends with the potential of getting a discount. The conclusion that cascade sizes roughly follow the heavy-tailed distribution follows from the degree distribution of the network; the size of a cascade is intimately associated with the out-degree of the node. We can imagine a similar heavy-tailed distribution when looking at downloads across projects.

Herlocker et. al. have an oft-cited paper that discusses the techniques and challenges involved with evaluating recommender systems based on the collaborative filtering [4]. The paper looks at a wide range of metrics on various datasets, and finds that there are 3 clusters of relevant accuracy metrics: accuracy metrics, classification accuracy, and rank accuracy. The most relevant metric for us is classification accuracy, which measures the frequency with which a recommender system makes correct or incorrect decisions about whether an item is good. The use of each evaluation metric depends on what the user cares about, but these numbers help us identify how our algorithms perform under various use cases.

3 Data

There were two main sources of data in our investigation. The primary data source is a dump of the past 18 months of log data. One line in the log file contains the IP address, page visited, project umbrella, and a timestamp of every request to the Simtk server (among other fields). The results presented here operate on the last 6 months of data, representing 1.4 GB worth of data, 14.2 million pageviews, and 142,800 unique IP addresses. We would like to view IP address as a proxy for a user, although it is possible for a single user to access the site from multiple IP addresses. Since session information is not included in the log data, there is no way to reconstruct the browsing pattern of an actual user using only log data. Therefore, we use IP addresses as a rough approximation for a single user in tracking



# projects visited	# people
15	123
16	69
17	54
18	46
19	46
20	39
21	60
22	79
23	85
24	94

Figure 1: Plot showing the distribution of users that visit a certain number of projects. Recommendations are designed for the users such as those in the table at right.

the navigational pattern through the site. One advantage to this approach is that it may be a more faithful representation of the projects viewed in one sitting; a user will likely not switch devices or attain a different IP address in the middle of a browsing session.

Deploying and tuning the recommendations depends heavily on the application and the definition of a utility matrix. In our context, the utility matrix is the number of times a project is visited by a particular IP address. We note two attributes about this definition of utility matrix: first, it captures implicit feedback, in that it contains the number of times a particular IP address viewed a project. We assume that a pageview inherently indicates an interest in the project. Second, as is often the case, this utility matrix is sparse. This follows from the fact that most users have not interacted with most projects on the site, so our utility matrix consists mostly of 0s.

The second data source is the publicly available numbers regarding project download information and forum activity. The number of downloads is an explicit feedback metric for the utility and quality of the project.

4 Methodology

First we present an analysis of the Simtk social network in relation to how users browse projects. As per the discussion above, we collected data over the number of IP addresses that visited each project; this is directly correlated with the number of users. The log-log plot in Figure 1 shows that, as expected, the vast majority of IP addresses simply navigate to one or two projects, with an average of 2.92 projects visited. However, there is a heavy tail of IP addresses which browse many projects. The graph shows a distinct increase in number of IP addresses that visit 13-17 projects. Moreover, we found that 1.23% of IP addresses visit 15 projects or more; the recommendations are designed for these power users.

Analysis of Figure 1 indicates many regular Simtk users; those who have used multiple projects or are perhaps interested in learning more about other projects. Figure 2 is further evidence of this characteristic user, for whom recommendations would be useful. A small

N (Top N IP addresses)	Total % traffic by these IP addresses
100	0.4078
1000	0.5858
3000	0.6991
5000	0.7597

Figure 2: A small fraction of users compose most of the traffic on Simtk.

fraction of users on the site make up most of the traffic.

Now we turn to the actual algorithm for generating recommendations. In any recommender system, a utility matrix is essential to generate relevant recommendations. Our definition of user utility is approximated by a pageview on a project. We decided to employ a form of item-item filtering, using user navigational patterns to measure project similarity. A more traditional sense of item similarity would have been to compare project keywords or description, an approach like the one employed by Jun Wang et. al. [5]. They approximated item similarity by using Pearson correlation coefficient in addition to cosine similarity as a cumulative similarity measure. However, this would not have worked well here given the low quality of keywords and sparse information. To proceed with pageviews, we first we preprocess the log data, then populate the utility matrix, and finally output a list of recommendations for each project. The pseudocode for the algorithm is below.

```
1.  Generate an ordered list of all projects browsed by an IP address.
2.  For each project list p, populate the utility matrix as follows:
Consider the next k projects.  For each pair (p, k), increment the similarity
score (count) between these projects.
3.  Calculate the average similarity score of every pair of projects.
4.  Normalize:  subtract the similarity of two projects by the average computed.
5.  Output up to n = 5 recommendations for each project, requiring a minimum number
of downloads.
```

The above algorithm is parameterized by the **window size k** and the **number of recommendations per project, n**. The idea of the window size is needed to capture the idea of a browsing session. For example, if a user visits two projects a week apart, we intuitively would not consider the two projects as part of the user navigation pattern. We analyzed two methods for selecting window size, which revealed user behavior on Simtk. First, we utilize the timestamp information in the log data, incrementing the similarity score between (p, k) only if project k was visited within some number of seconds that project p was visited. The tradeoff of larger session time is discussed in the results section.

The other method of selecting window size simply selects 4 projects at a time, disregarding time. This approach has the advantage of populating that the utility matrix with a fixed amount of data. Even for IP addresses which only visit a few projects at a time, the number of increments to the utility matrix will always be 4. On the other hand, this does not address the issue that projects may not actually be visited in the same session. Selecting a fixed number of projects roughly approximates the concept of session by disregarding disjoint project visits.

The other parameter is the number of projects to show, n. Tuning n is important because we only want to show high-confidence recommendations for similar, high-quality projects.

We discuss in the next section the results of changing the minimum number of downloads for each project. Note that one consequence of the algorithm is that the recommendations are static at the project level- they will not change according to the user.

The number of downloads controls the number of recommended projects in output. Although the number of project downloads is a clear indication of high user utility, one issue with downloads is that they occur much less frequently than pageviews. Therefore, there is too little data and too much variance across projects to use download information alone as a basis for reasonable recommendations. The effects of this kind of data sparsity are thoroughly investigated by Badrul Sarwar et. al. [6]. They dealt with the data sparsity problem by introducing a new similarity scheme effective at increasing the prediction accuracy, and hence the quality of recommendations. More data, however, could allow us to compare the download patterns of two different users in order to generate recommendations personalized for users.

One important part of the algorithm is step 4, to perform normalization on the project similarity scores. As shown in the network analysis, there are a few projects on the site which command the vast majority of downloads. Therefore, when running the algorithm without normalization, the same 4 or 5 projects would get recommended for the majority of projects. The reason is that these projects showed up in the browsing windows almost universally. Suppose the number of visits to a project q from source p is $f(p,q)$ and I is the set of all projects. For a project pair (p,q), the normalized score S becomes:

$$S = f(p,q) - \frac{\sum_{i \in I} f(i,q)}{|I|}$$

This quantity is an asymmetric measure of how similar p is to q. It is asymmetric by design since we define a source and destination project in the utility matrix. If this quantity S is positive, it means users that visit p are more likely than average to visit project q. By subtracting the average, we are able to identify truly similar projects accounting for project popularity. The interface and sample results of the recommendations is shown in Figure 3.

5 Results

The recommender system as described above is implemented and running on the testing server `testsimtk.stanford.edu/` but not on the live site. In this version, the generated database table contains 2443 recommendations across 509 projects; the average number of recommendations per project given no download threshold is 4.79. As discussed in the Herlocker paper, a fitting evaluation metric for our purposes is classification accuracy metrics [2]. This measures the frequency with which the recommender system makes a correct or incorrect decision about whether an item is good. In this context, since downloading a project is time-consuming, we would like to minimize the false-positive rate. That is, the rank of recommendations is not as important as having confidence that the recommendations we show are all high quality.

A comparison of our intuitive judgement of project relatedness shows a fairly effective recommender system. For example, the project *Competition to Predict In Vivo Knee Loads* has top recommended projects of *MB Knee: Multibody Models of the Human Knee*, followed

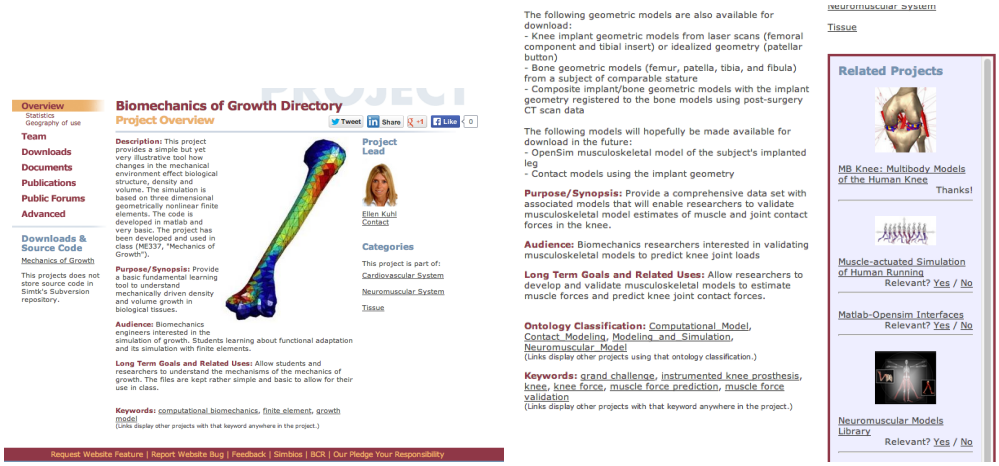
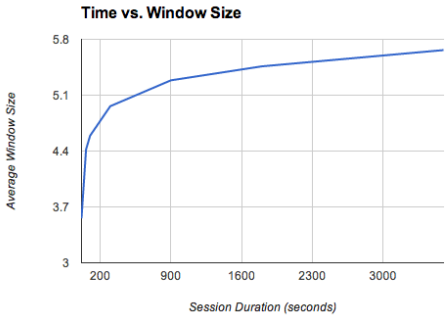


Figure 3: A project without recommendations on the left, and with recommendations on the right.



Window Size	# recommendations
4	2443
5	2450
6	2457

Figure 4: Left: the increase in average number of projects browsed as a function of session time. Right: larger window size leads to more recommendations since we have more data.

by *Muscle-actuated Simulation of Human Running*, as shown in Figure 3. We added a binary feedback button for each recommendation so the user can record whether the recommendation was valid or not. However, since the recommender system is not on the live site yet, we do not currently have this relevance feedback. Instead, we ran several experiments to see differences in algorithm output.

First we analyzed how changing the duration of a session changed the average window size k. Increasing the window size is a trade-off between exploration vs. exploitation. With a large window size, the more popular projects on the site begin to dominate the recommendations (exploitation). A small window size limits the number of recommendations we are able to create, since we have less data in the utility matrix. The increasing number of recommendations are shown in Figure 4.

In order to guarantee that the projects we show are at some minimum level of quality, step 5 of the algorithm checks that every project must be downloaded a minimum number of times. This ensures a strong classification accuracy because we minimize the false-positive rate of, for example, recommending a test project with no downloads. A project may have no downloads either because the project was created for testing purposes, or it was intended to advertise a symposium/meeting. Since there is no administrative oversight for the creation,

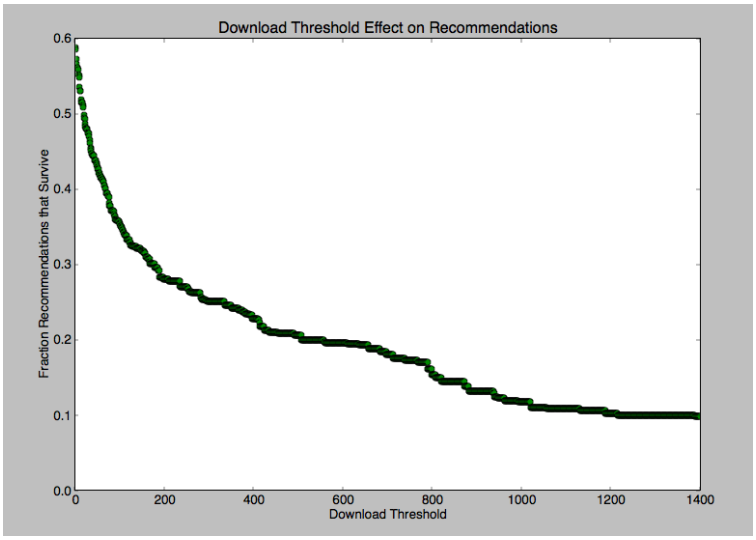


Figure 5: Fewer recommendations survive as we increase the download threshold since projects with few downloads do not get shown.

41.2% of projects actually had 0 downloads.

Figure 5 shows how the number of recommended projects changes as a function of the mandated number of recommendations. The plot starts with 58.8% of the 2443 projects which have been downloaded at least once, then flattens out quickly. This makes sense given the stratification of projects on the site: a select group of projects are downloaded many times (so they will always be shown regardless of download threshold), while the large majority are downloaded less than 100 times.

Finally, we explored the effect and importance of normalization by analyzing the distribution of project traffic on Simtk. There are several projects which command a disproportionate amount of user traffic, but we want to recommend a large diversity of projects. Figure 6 shows the extent of this problem, that only a handful of projects command much of the traffic. The graph on the right shows the analogous trend as a log-log plot for pageviews: there are a few power users which are responsible for most of the traffic on the site. One of the goals, of the recommender system is therefore to identify and highlight lesser known projects to allow for discovery. The ability of our recommender system to do this is shown in Figure 5, dependent on how we tune the download threshold.

6 Future Work

A first version of the recommender system is complete, but other approaches could be explored. A defining assumption in this paper was that pageviews were a good proxy for user utility of a project. Other metrics that could be included in this measure of utility include number of forum posts, time spent on the project page, and number of times the project is downloaded. We did not have access to this data since we had a flat log file. Having user session info would resolve ambiguity about which projects to include in the browsing window.

Another major area of work is the need to evaluate the comparative success of each

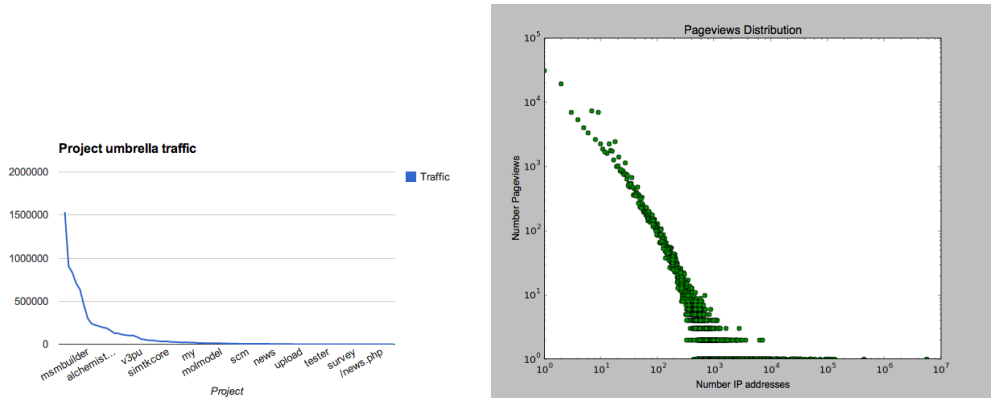


Figure 6: A few projects receive the most pageviews, and a few users are responsible for most of these pageviews.

parameterized algorithm, and furthermore how to measure the effectiveness of the recommendations. One difficulty in selecting the parameters n and k optimally is that it is not clear when one set of recommendations is better than another. The high level goal of the system is to enable researchers to quickly and more accurately identify the projects of interest to them. Increased engagement is an indirect way to evaluate success; whether users browse more projects or if more downloads occur. Given more time and the ability to collect log data on the live site after deployment, these numbers would have been revealing.

7 References

[1] Joy P. Ku and Jeanette Schmidt. Simtk.org Website Users Manual. Published August 10, 2010, Release 1.0. 10/16/2013. `https://simtk.org/user_guide.pdf`

[2] Simtk.org News Page. 10/16/2013. `https://simtk.org/news.php/site`

[3] Jure Leskovec, Ajit Singh, and Jon Kleinberg. 2006. Patterns of influence in a recommendation network. In Proceedings of the 10th Pacific-Asia conference on Advances in Knowledge Discovery and Data Mining (PAKDD'06), Wee-Keong Ng, Masaru Kitsuregawa, Jianzhong Li, and Kuiyu Chang (Eds.). Springer-Verlag, Berlin, Heidelberg, 380-389. `http://dx.doi.org/10.1007/11731139_44`

[4] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. 2004. Evaluating collaborative filtering recommender systems. ACM Trans. Inf. Syst. 22, 1 (January 2004), 5-53. `http://doi.acm.org/10.1145/963770.963772`

[5] Jun Wang, Arjen P. de Vries, Marcel J.T. Reinders. 2006. Unifying User-based and Item-based Collaborative Filtering Approaches by Similarity Fusion. `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.101.692`

[6] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-Based Collaborative Filtering Recommendation Algorithms. `http://dl.acm.org/citation.cfm?id=372071`