1. INTRODUCTION

The development of Internet technologies and the concept of open source enable more and more people to be involved in generating content on the Web. Every day, an enormous amount of content is generated in various formats of audio, video and text. However, the rapidly growing volume of information makes it harder for users to identify the content they are interested in and that they can build upon. Recommender systems are one of the effective tools that can help users to cope with the information overload and can provide personalized suggestions. Typical recommender systems aiming to predict user interests are based on user profiles, demographic information and users content ratings [Barragáns-Martínez et al. 2010]. When explicit ratings are not available, implicit ratings such as users’ history of purchase and click-stream patterns become useful information for recommender systems.

With the development of Web 2.0, most of these user generated content communities utilize a particular family of applications known as Social Tagging or Collaborative Tagging Systems. These applications allow users to create and share lightweight metadata in the form of chosen keywords called tags to represent the created content [Golder and Huberman 2006]. Some popular examples of these communities that support collaborative tagging are Flickr and Photobucket for photos, Last.fm and ccMixter for music and Scratch for youth game development. The tags used in these communities help users to self-organize, share and find content they are interested in [Ames and Naaman 2007]. Since tags are local descriptions of content provided voluntarily by users, they represent additional personalized information both about the user and the created content which can later be used for the creation of recommender systems [Halpin et al. 2007, Tso-Sutter et al. 2008, Liang et al. 2008].

In this paper, we propose a recommender system for the Scratch online community. Scratch users create, share and remix projects by using the Scratch programming language developed by the Lifelong Kindergarten Group at the MIT Media Lab [Resnick et al. 2009, Roque et al. 2013]. The proposed recommender system utilizes project tag information to determine similarities between various users and then uses these relationships to identify the optimal set of items to be recommended. Our aim is twofold: (i) Create and evaluate recommendations based on two different types of input tags (explicit and implicit) and (ii) evaluate recommendations based on relevancy and diversity. We are including diversity in our recommendation algorithm under the assumption that recommending a more varied set of items will be more valuable to users than simply recommending similar items. Through a calculated combination of relevancy and diversity, our recommender system is aimed at leading users to explore further into the Scratch community and improving the productivity of “passive producers” by using the output of “active consumers” [Monroy-Hernández and Resnick 2008].

1.1 Related Work

Over the last few years, the recommender systems developed for various collaborative tagging systems have mostly been concentrated on recommending personalized tags to users for annotation related tasks [Heymann et al. 2008, Xu et al. 2006, Mishne 2006, Jäschke et al. 2008, Marinho and Schmidt-Thieme, 2008]. There hasn’t been much research on utilizing actual collaborative tagging information to recommend content to users. [Tso-Sutter et al. 2008] have proposed a collaborative filtering (CF) algorithm that converts tag information into two-dimensional user-tag and tag-item relationships. [Liang et al. 2008] have added a three dimensional relationship of user-item-tag to the existing two-dimensional user-tag and tag-item relationships to identify the most similar neighbors and items. In the [Barragáns-Martínez et al. 2010] study, a folksonomy showing the relationship between different tags was created and used in combination with ratings to make content recommendation to users.
While these tag-based recommender systems use tag information to enhance traditional algorithms used in recommender systems, they focus only on attaining relevance, i.e. recommending the most individually relevant items to the user. By prioritizing only relevance, such recommender systems return a set of monothematic items matching user interest; however if the user decides to avoid that one theme, a problem arises. The returned set of recommendations will be very similar not only to the target query, but also to each other and hence, may not satisfy user’s diverse interests [McNee et al. 2006, Ziegler et al. 2005]. Combining diversity with relevancy provides the user with optimal coverage of the information space without compromising the similarity of the recommended items [Smyth and McClave, 2001, Genc and Nickerson, 2013].

In set based information retrieval, diversity is related to how different the items are from each other. When a set is considered diverse, each item in the set is novel with respect to the rest of the set [Vargas and Castells, 2011]. Maximum Marginal Relevance (MMR) [Carbonell and Goldstein, 1998] is one of the most widely known methods for balancing relevancy and diversity in set based information retrieval. MMR is also used for result set diversification in the recommender system literature [Candillier et al. 2011, Guo and Sanner, 2010].

2. Design
2.1. Methods
During the generation of personalized recommendations we considered two different data inputs: 1) User content tags; tags of the projects created by the user (shared tags) and 2) user favorite tags; tags of the other users’ projects that the user has explicitly favorited. Since users are free to choose any keywords as tags, we stemmed all the tags to reduce them to a common form. Using latent semantic indexing (LSI), we constructed a collaborative view of user profiles and created user-tag and user-user similarity matrices in a reduced dimensional space. Using MMR algorithm, 

\[ MMR(Q, Di) = \lambda Sim(Q, Di) - (1 - \lambda) \ max Sim(Dj, Di) \]

we iteratively selected the user’s with highest similarity to our query and then updated the remaining user similarity scores by computing the degree of dissimilarity between each user and the previously selected ones. In the equation, \( \lambda \) signifies the relevance to diversity ratio and when \( \lambda \) changes, the MMR weighted sum changes accordingly, and so does the ranking of users. When \( \lambda \) equals to 0, novel users rank higher. When \( \lambda \) equals to 1, highly similar users get ranked higher. The challenge for our recommender system is to balance this tradeoff between two possibly conflicting objectives of 1) recommending highest ranked similar users and 2) recommending highly diverse (novel) ones. In order to find the most efficient algorithm, we conducted three sets of controlled studies on Amazon Mechanical Turk, as will be described in the next section.

2.2. Conditions
In condition 1, we evaluate the shared tags of a randomly selected user and \( \lambda \) values of 0, 0.5 and 1. To test the quality of our final recommendations, a set of HITs were posted on Amazon Mechanical Turk. Each HIT was opened to 20 Amazon Mechanical Turk workers. Workers were presented with the project tags of our user and recommended users’ project tags. They were asked to review the recommendations and indicate their likelihood of viewing recommended projects.

In condition 2, we evaluate the favorite tags of our user and \( \lambda \) values of 0, 0.5 and 1. A new set of HITs was posted for condition 2 in the same format of condition 1.

In condition 3, we randomly selected 20 users from our user pool. 10 were assigned to the shared tags group and 10 were assigned to the favorite tags group. Turkers were asked to evaluate these users similar to conditions 1& 2.

3. Results
According to the means, the shared tag condition reveals that the likelihood of viewing recommended projects is highest when \( \lambda \) equals to 1 (similarity high – novelty low). However, p-values show no significant difference between the different values of \( \lambda \). The favorite tag condition, on the other hand, shows that the likelihood of viewing recommended projects is highest when \( \lambda \) equals to 0.5.
We compared the performance of the recommender system in regard to the input of the system. When viewing the recommended user's projects, the favorite tags group performs significantly better than the shared tags group (Table 3).

When compared with the control group, the recommendations based on shared tags do not perform well (Table 4). However, compared with the control group the recommender system performs better when using favorite tags (Table 5).

4. Conclusions and Future Work
One of the popular means for content delivery used by online communities is recommendation. Traditionally, recommender systems focus on generating recommendations that are very similar to users' interests. This can result in too analogous recommendations that fail to engage the users. Hence, this paper proposes a tag-based recommender system that generates diversified recommendations for Scratch online community. Our initial experimental results show that recommendations that balance the notions of relevancy and diversity \((\lambda =0.5)\) perform better. Our results also show that using favorite tags instead of shared tags may result in better recommendations. The act of "favoriting" might be a better indication of user interest and show a clearer direction to what future projects user would like to create next. In that sense, a favorites list could be used as a predictor and a repository for future project ideas.

Several things can be done to improve the recommender system for future studies. Different synthesized or natural datasets could be used to test the recommender algorithm. Using a dataset that Mechanical Turk workers are more familiar with (movies, books etc.) would help us better translate the recommendations onto the AMT platform. Additionally, a genetic algorithm can also be used in the optimization of the final recommendation set.

Another study we are interested in conducting is to generate Scratch project recommendations based on the similarities of the project codes. As assigning tags is not a requirement on Scratch, users are inconsistent when tagging their projects and as a result, our current recommender system would not be useful in those cases. Since all projects have codes and we are able to get information on remixes, project codes can be used for user profiling. A third study we might take on is developing a social matching system [Terveen and McDonald, 2005] for Scratch. Since Scratch is an online collaboration community, it would make a good platform for a social matching system that aims to increase social interaction and foster collaboration.

Acknowledgements
This material is based upon work supported by the National Science Foundation under Grants IIS-0968561 and IIS-1211084.
REFERENCES


