Chapter 6

Discussion and Future work

Although we have successfully displayed the working of the proposed recommendation methods, there are many more ways in which the recommendation system for a complex platform like Scratch could be developed. In the design of this item-based recommendation, we assume the item-item similarity and user rating are the defining characteristic of the user preference, however that may not be necessarily the case. In this chapter, we discuss the limitations of the current system and the possible improvements that could be made to improve the performance of current recommender system.

**Limitations**

**Over specialisation problem**

One of the major problem faced by collaborative filtering algorithms is the problem of over specialisation. A classic example of this is when one buys a lawn mover on amazon, he only sees recommendations for lawn mover. A good recommender should provide suggestions on a diverse set of items, rather than suggesting the same type of items that the user is already acquired or liked in past. The recommendation system proposed in this study suffers from this problem i.e. if a user has liked a certain type of project in Scratch, the recommended are generated for the project that has a similar characteristic. It prevents the user from discovering new items that might help them. Researchers have tried to solve the problem by neighbourhood based CF and hybrid recommender systems [51].

**Concentration bias**

A concentration bias arises when a recommender system proposes the items with higher ratings more than with low ratings.specially in CF algorithms which use user ratings to recommend the items, they often create the rich get richer effect for more popular items, this concentration bias holds back what may be more appropriate product for user from getting discovered. Many researchers have tried to eliminate concentration bias problem from classical recommendation algorithms like k-NN filtering [51].

**Cold Start problem**

A cold start problem occurs when a new user or item enters the system and there is no information about it. Cold start problem can be broadly divided into three types new item problem, new user problem and new system problem [53]. In the case of a new user, it is very difficult to give a recommendation as there is no information available about the user. In the case of projects in scratch when a new item enters the system there is some information about the project, hence we can at least compute the similarity matrix for the item. But without a user rating, we cannot calculate the prediction score. Content bases systems succeed in case of the new item they do not depend on the user for generating the recommendation.

**Dataset limitation**

There are many limitations to the available dataset that was shared by the Scratch community. Firstly it is an dataset scraped at the beginning of the Scratch project. The quality of projects and users may have changed over time as they develop more complicated projects. A more recent dataset would give a better understanding of the user's mindset and may probably give a different result for different similarity algorithms. The dataset also provides very little information on the user rating part. Some implicit information like the clicks the user makes and the

Over the years scratch has also gone through many iterations of development. The first version (Scratch 0.x to 1.x) of the scratch was implemented in Squeak. Later when Scratch 2.0 was launched in 2013 it shifted to ActionScript. The current version of scratch launched on January 2nd, 2019 i.e. Scratch 3.0 uses javaScript for its implementation. This development has led to many UI changes over time. UI may have a significant influence on user activity and development skills. Many new attributes and features have been added to the newer version of the project files that give us the parameters for item similarity computation. Hence the more recent data collections effort is needed in order to accurately calculate the item similarity between projects.

**Future work**

**Qualitative testing**

As discussed in the previous chapter we have evaluated and validated the prediction accuracy for the proposed recommendation techniques. Research in the recommendation system space mainly focuses on the prediction rating accuracy. And since accuracy only partly comprises the user experience of the recommender system, more qualitative testing is required for evaluating the usefulness of the technique. It needs to be verified against the need of the user who is getting the recommendation. One of the goals of developing this recommendation system for Scratch was to aid the user in their learning journey by suggesting projects that might improve their skills and the techniques they are learning. Some of the methods that could be used are an A/B testing amongst the users, collecting user data about the recommendations or by surveying the users for their opinion about the recommended items.

**A/B testing -** also known as the bucket test or split-run test is a method to compare two versions, A and B of the same technique, commonly done on users response to the different versions to determine which one is more effective. Using the A/B testing methods different versions of recommendation can be served to users to see which one is more useful to the user. For example, if the user chooses to view a more complex project instead of the more similar one to the original, this could give an insight of how to develop a recommendation which will help the user in discovering more complex projects.

**User Surveys -** a user could be conducted to get an understanding of what the user thinks of the recommendations given to him/her.This could help us understand the user needs and performance parameters upon which we could tune the recommendation system. As noted in [49] preference elicitation is a difficult problem since many time a user is unaware of his own personal preference, especially when he is beginning to use the system or is unaware of all the available choices. [49] also suggests asking users a small set of goals in the signup process to understand the user's recommendation parameters for the user.

Many novel methods have been proposed in [50] which focuses on the user-centric development of the recommendation systems. It proposes a framework that considers how the objective components of the system e.g. algorithms, are perceived subjectively and how these perceptions along with situational and personal characteristics, gives a different user experience leading to different interactions with the system. For example, if the user perceives the recommendation quality of these different algorithms differently, a higher perceived quality of the recommendation leads to higher interaction with the system.

**Mining text from the dataset**

While exploring the available dataset, we found that the text and code Dataset contains much interesting information that might be useful in building a stronger recommender system. The comments posted on a project and galleries are available in pcomments\_text and gcomments\_text datasets. Projects\_strings contains information about all the user-generated strings that have been used in the project. All these text-based information could yield some meaningful information if explored correctly.

A careful sentiment analysis of the comments might help us understand the kind of responses a project is getting. Based on which we could infer project usefulness to various users. In addition to this, comments on different project by the same user could help us understand user’s personal taste which could help in predicting user choices.

Strings inside projects could help in categorising the project into various groups and sections, which in turn could help in building a cluster-based recommendation.

Also, there is information available about the tag\_text which is the tags assigned to the project by different users. This could also help in categorising the projects. TFIDF algorithms could be used to find the frequency of terms and then classify the projects into different categories.

**Social network analysis**

Scratch also has a large community website and a community forum where users interact with one another, asks for help and post useful stuff. Users can follow one another if they find the profile interesting. A social network graph analysis can be done on this information to find users with similar interests and user groups that are closely tight to identify similar interest groups. Followers can be recommended base don this information. Projects can also be recommended amongst the groups themselves based on the idea of being from the same circle.

**Recommendation based on heuristics**

CF-based and content-based recommender systems are well known in the field. However, they may not be perfect for every recommendation problem. Many different approaches need to be explored for building a good recommendation for scratch. One of the more simple and easy to implement approach is the heuristics based. In a heuristics-based system, one could implement a different set of rules to recommend projects to users. For example, if the user is new in the system, we might recommend projects that guide the user in getting started and simpler projects. If the user is an experienced one and has built many complex projects, recommendations for projects from other users who have more complex projects in their profile may be recommended.