Reciprocal Recommendation System and Formation of Learning Groups in Massive Open Online Courses (MOOCs)

by

Sankalp Prabhakar

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

Abstract

Online learning is an emerging education technology area with increasing demands. Massive open online courses (MOOCs) is one such platform where users with completely different backgrounds subscribe to various courses on offer. However, oftentimes these users are hesitant to approach other users for collaboration on certain tasks. In this paper, we propose a reciprocal recommender system that matches users who are mutually interested in, and likely to communicate with each other based on their profile attributes (age, location, gender, qualification, interests, grade etc.).

We also present a 'group formation' strategy by using the *particle swarm optimization* based algorithm which would automatically generate dynamic learning groups. To form effective groups, we consider two important aspects: a) *intra-group heterogeneity* and b) *inter-group homogeneity*. Intra-group heterogeneity advocates the idea of diversity inside a particular group of users whereas inter-group homogeneity recommends that each group should be similar to the other.

We test our algorithm on synthesized data sampled using the publicly available MITx-Harvardx dataset. We measure the quality of generated groups based on a certain *fitness* measure, which is then compared against the fitness of groups obtained using the popular standard clustering algorithm like k-means. Evaluation of the recommender system is based on our own defined measures of precision, recall and normalized discounted cumulative gain. Experimental results show that our system performs better than the baseline models, therefore it makes a promising case for such a system to be implemented within an actual MOOC.

Preface

This is a collaborative work with authorship shared between me and my supervisor, Osmar Zaiane. Also, Jerry Spanakis, who is a research scholar at Maastricht Unversity, has contributed with ideas related to building a reciprocal recommendation system for MOOCs. I have designed and conducted the experiments on my own and deduced important observations from the experimental results. The data for the experiments have been borrowed from the publicly available MITx-Harvardx dataset, which was partly synthesized to fulfill the requirements of our experiments. The other collaborators have provided feedback on the design of experiments and improvement of the methodologies.

To my maa, sister and my teachers
For being the inspirations in my life.

Perfecting oneself is as much unlearning as it is learning.

– Edsger W. Dijkstra.

Acknowledgements

I would like to thank my supervisor, Prof. Osmar Zaiane, for his support and guidance throughout this journey. His insightful discussions and guidance has broadened my thinking horizons and helped me to move in the right direction for my research. Right from the beginning, he has encouraged me to pursue different paths in search for a solution to any given problem and has helped me overcome any roadblocks that I faced.

A special thanks to the Osmar group members, especially Hamman Samuel. I thoroughly enjoyed our research meetings and learned a lot out of those extended sessions. Furthermore, I would like to thank Jerry Spanakis, whose ideas helped me to develop some of my research work.

I would also like thank my friends - Soham, Varun, Sanket, Megha, Vivek and Gautham for all the moral and technological support. I would like to thank my father, sister and extended family for their continuous support to pursue my dream. I could not have come this far to pursue my degree without the encouragement of all my friends, family and teachers who have contributed immensely to my overall development. Finally, this was impossible without my mother. Thanks to her for being the constant source of positivity in both good and bad times.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Higher education is an area that has thus far embraced, but arguably has not been fundamentally altered by the growth of the Internet. This has been rapidly changing over the last few years with the rise of Massive Open Online Courses (MOOCs) as a way of learning that enables students to participate on their own terms and conditions via Internet. In one of the first MOOC [7] developed by edX, the consortium led by MIT and Harvard, over 155,000 students initially registered, the MOOC was composed of video lectures, interactive problems, online laboratories, and a discussion forum. The number has only grown since then and according to data collected by Class Central[1], the total number of students who signed up for at least one course in the year 2015 has crossed 35 million - up from an estimated 16-18 million the previous year.

MOOC courses integrate the connectivity of social networks, the facilitation of an acknowledged expert in a field of study, and a collection of freely accessible online resources [37]. Whilst the hype over MOOCs has subsided, there continues to be an year on year growth in participants and proliferation of MOOCs utilization in different learning, research and marketing contexts. MOOCs have been embraced in a big way by elite universities and institutions, and are beginning to have a major impact on higher education. Success of MOOCs relies on the fact that they give students the opportunity to engage with learning in an open format via the Internet. Through the MOOCs'

[1] https://www.class-central.com

self-paced format, a student can engage with the material in a way that works best for him. At the same time, students can self-organize through the various tools offered via any MOOCs platform. MOOC learners are diverse, originating from many cultures across the globe in all ages and backgrounds [18]. Despite this diversity, three main attributes unite them: A desire to learn, a desire to connect to a global community and a desire to experience and consume content online.

However, there are legitimate concerns surfacing about MOOCs as their popularity spreads far and wide. In their study about behavioral patterns in MOOCs, the authors attribute the lack of effective student engagement as one of main reasons for a very high MOOC dropout rates [46]. Their study reveals that although many thousands of participants enroll in various MOOC courses, the completion rate for most courses is below 13%. Moreover, lack of effective collaboration between students further dissuade them from participating in projects and assignments which are an essential element for student evaluation in an online learning environment.

The first part of our research work focuses on encouraging peer-learning and student collaboration by developing a *reciprocal recommender system* that will recommend users to each other based on the potential of mutual likeliness, derived using attributes like *age*, *gender*, *location*, *qualification*, *interests* and *courses*. The system takes into account not only the relevance of these attributes between user profiles but also the degree of importance that each one of these attributes have for the user. Our algorithm looks at the attribute values and preferences of a user, based on which it generates a recommendation list of *top-N* users who are the best fit for the given user.

The emergence of MOOCs as a major source of learning in the modern world has also created challenges in terms of forming effective groups of learners. With MOOCs becoming more and more popular, the normal online class size is increasing exponentially. More people signed up for MOOCs in the year 2015 than they did in the first three years of the "modern" MOOC movement (which started in late 2011 when the first Stanford MOOCs took off) [54].

The students registered on MOOCs have a varied demographics in terms of the countries they originate from, languages they speak and their personality traits. Hence, it becomes even more demanding to form effective groups of students to enable healthy communication and team work thereby bringing out the best possible learning outcomes.

Keeping in mind the above sentiment, the second part of our research work focuses on exploring the possibilities of assisting MOOC learners in the process of self-organization (e.g. forming study groups, finding partners, encourage peer learning etc.) by developing a *group formation* strategy based on a pre-defined set of user attributes like *age*, *gender*, *location*, *qualification*, *interests* and *grade*. We use a modified particle swarm optimization [28] technique which helps in effective group formation by looking at the different user attributes. The grouping strategy is based on the conditions of *intra-heterogeneity* and *inter-homogeneity* among groups. The attributes are pre-defined to be either *homogeneous* or *heterogeneous* so that they fit into the group formation strategy accordingly. For instance, if we have 'grade' as an *inter-homogeneous* attribute, it would mean that all groups should have similar grade levels. Similarly, if we have 'interests' as an *intra-homogeneous* attribute, it would mean that the interests of students within each group should be aligned or similar. We believe that recommending learners to each other and having effective student groups will foster better collaboration and engagement, which in turn would help mitigate the dropout rates to some extent.

The remainder of the dissertation is organized as follows. Chapter 2 presents some background and related work on the criteria of reciprocal recommendations and group formation in MOOCs. In Chapter 3, we talk about the data and reciprocal recommendation model for generating and ranking recommendations. Soon after, in Chapter 4, we look at the group formation model to dynamically generate student groups based on the factors of *inter-group homogeneity* and *intra-group heterogeneity*. Lastly, Chapter 5 ends with a summary and conclusion.

Chapter 2

Background and Related Work

2.1 Background

In this section, we present some background on recommender systems, reciprocal recommendation and group formation strategies. *Recommender systems* have become extremely common in recent years, and are utilized in a variety of areas like e-commerce, search queries, social tags etc. The goal of a Recommender System is to generate meaningful recommendations to a collection of users for items or products that might interest them [38]. Suggestions for books on Amazon, or movies on Netflix, are real world examples of the operation of industry-strength recommender systems. Recommender systems differ in the way they analyze these data sources to develop notions of affinity between users and items which can be used to identify well-matched pairs. *Collaborative Filtering* systems analyze historical interactions alone, while *Content-based Filtering* systems are based on profile attributes; and Hybrid techniques attempt to combine both of these designs.

In the context of education, the goal of a recommender system would be to help the user or a group of users find suitable resources and learning activities for a better achievement of the learning goal and development of competences in less time [14] [13] [12]. However, unlike the traditional recommendation systems where its relatively easy to evaluate the quality of recommendations, it becomes difficult to assess the recommendations in educational context as the learning goals cannot be quantified accurately. There are some other *contexts* in which recommendation systems are utilized in the field of education, we

discuss it in detail in the next subsection.

Unlike the traditional recommender systems, where a *user* is provided recommendations to *items* which is likely to be of interests to the user, in *reciprocal recommendation* the likeliness has to be *mutual* [50]. In a reciprocal recommender, the *user* and the *item* have similar standing in the system, in that both have preferences that must be satisfied. For instance, in *online dating*, the preferences of both the users must be satisfied before they are recommended to each other. The recommendation in this case is bidirectional in nature. Most of the research work surrounding reciprocal recommender systems has been in the field of online dating and job recommendations. Given our problem of recommending learners to each other on various MOOC platforms, it becomes imperative to exploit the theories in the field of reciprocal recommendation for this given purpose.

Effective *group formation* strategy can be utilized to improve the learning outcome in an online environment. One of the early theories in group formation was formulated by Dr. Bruce Tuckman, which was published in his Forming Storming Norming Performing model [60] in 1965. Tuckman's model explains that as the team develops maturity and ability, relationships establish, and the leader changes leadership style. Beginning with a directing style, moving through coaching, then participating, finishing delegating and almost detached. At this point the team may produce a successor leader and the previous leader can move on to develop a new team[1]. Some of these ideas can be applied in educational context as well, in the sense that we need users who are naturally able to drive conversations or group discussions.

However, effective collaboration in learning is still a key issue. Unfortunately, there does not exists a defined methodology for group formation, resulting in important performance differences; some of them can accomplish the learning objectives and some of them are far from the minimum success criteria. The problem arises when there is not an identifiable leader, or there are non-compatible personalities. Due to the lack of diversity, some learning goals cannot be satisfied. Studies have shown that diverse online classrooms

[1]http://www.businessballs.com/tuckmanformingstormingnormingperforming.htm

can create benefits that are largely unavailable in a traditional classroom as it has been observed that globally diverse discussions boost student performance and engagement [32]. Research in many disciplines has shown that learning within groups improves the students' learning experience by enabling peers to learn from each other and by providing rapid feedback on things to improve [33]. To form groups, students can be either allocated to groups randomly, self-select each other, or be appointed to a group by the teacher based on some criteria related to the collaboration goals. However, forming groups manually can be both difficult and time consuming. This is why researchers have been investigating several techniques for automating this process using certain computer supported methods.

2.2 Related Work

There has been quite a few previous works on recommender system in educational contexts, but their main focus has been on recommending courses to learners [5], [4], helping students take decisions on their academic itineraries [64] etc. However, not much significant work exists on recommending learners to each other, in order to build a learner community within a MOOC. The use of *reciprocal recommendation* has found certain traction within the domains of online dating and job recommendations but the idea has not been explored much in the context of recommending learners in an online setting. We look at many different works of applying recommendation in various contexts, some of which inspired the work we did for our research.

Li et. al. [34] built a generalized framework of reciprocal recommender system which models the correlations of users as a bipartite graph that maintains both local and global 'reciprocal' utilities. The local utility captures users' mutual preferences, whereas the global utility manages the overall quality of the entire reciprocal network. Similar work was done by the authors of [51], in which they use collaborative filtering combined with a stochastic matching approach to build people to people match. They introduce a method to remove the popular user bias by ensuring that every user receives the same number

of recommendations as they are recommended to others. That is, a user A is only recommended to a user B if user B is also recommended to user A. This ensured that popular users are not recommended more often than unpopular ones.

Some of the theories that we built for our research have also been influenced by the ideas in the Recommender Systems Handbook [30]. The workbook describes some of the most popular and fundamental techniques used nowadays for building recommender systems, such as collaborative filtering, semantic-based methods, data mining, and context-aware methods. The work also focuses on methods and techniques for evaluating the performance and effect of recommender systems, and diverse applications of recommendation techniques.

In their work [2], the authors build a scalable People-to-People Hybrid Reciprocal Recommender Using Hidden Markov Models. They introduce a general framework for combining a Hidden Markov model (HMM) content-based reciprocal recommender system with collaborative filtering techniques to create a unified hybrid recommender. Evaluation of their system shows that it generates better recommendations than their counterparts in a time-efficient manner. While HMM is useful in building a hybrid recommendation system like above, it could not really fit in our context where the 'content' of the user in terms of user attributes is the governing factor behind any recommendation.

Some of the most significant work with respect to reciprocal recommendation has been done in the field of online dating. The subject is more relevant here because in an online dating setting, a successful match only occurs when both the recommended people like each other or reciprocate, much similar to our requirement where the learners should only be recommended if they satisfy the preferences of each other. In the next sub-section we look at some of the work related to this field.

2.2.1 Reciprocal Recommendation for Online Dating

Akehurst et. al. built a Content-Collaborative Reciprocal (CCR) system [1] for online dating. The content-based part uses selected user profile features and similarity measure to generate a set of similar users. The collaborative filtering part uses the interactions of the similar users, including the people they like/dislike and are liked/disliked by, to produce reciprocal recommendations. CCR addresses the cold start problem of new users joining the dating website by being able to provide recommendations immediately, based on their profiles.

A similar work was done by Pizzato et. al. [48], where people-to-people recommenders constitute an important class of recommender systems especially in the area of online dating. It is important to study and illustrate the distinctive requirements of reciprocal recommenders and highlight important challenges, such as the need to avoid bad recommendations since they may make users feel rejected. Our work to generate a list of similar learners based on user attributes was partly inspired from these ideas.

RECON - a reciprocal recommender system [49] for online dating was another idea which made use of user preferences to calculate compatibility scores of users with each other. It was observed that the predictive power gained by taking account of reciprocity, improves the success rate of the top ten recommendations significantly.

Xia et. al. in their work in reciprocal recommendation [67], introduce similarity measures that capture the unique features and characteristics of the heterogeneous online dating network. They build a preference model for each service user based on the attributes of users who have been contacted by the service user. They characterize the interest similarity between two users if they send messages to same users, and attractiveness similarity if they receive messages from same users. Based on these similarity measures, compatibility between a service user and potential dating candidates is computed and a recommendation list is generated to include candidates with top scores.

Lastly, Zhao et. al. did a case study for user recommendations in recipro-

cal and bipartite social networks [70], where they propose a new collaborative-filtering model. The model considers users' taste in picking others and attractiveness in being picked by others. Evaluation of the model on an online dating network shows that the approach offers good performance in recommending both initial and reciprocal contacts.

Overall, most of the existing work related to reciprocal recommendations for online dating revolves around building a metric to map one user to another on several attributes either specified by the user in their profile or based on their interaction history with other users. While these ideas have helped us to build our own similarity metric, the implementation varies as the user attributes for MOOC are inherently very different. In the next subsection, we look at some related work for reciprocal recommendation in the area of job recruitments.

2.2.2 Reciprocal Recommendation for Job Recruiting

Yu et. al [69] propose a reciprocal recommendation algorithm for the field of recruitment. The authors propose an implicit preference function calculation method, based on the resume delivery in recruitment systems and mining the potential preference information of users. Then given consideration that users' explicit and implicit preference information has different effects in computing similarity, a similarity calculation method based on the integration of these two preference information is proposed. At last, the reciprocal recommendation is achieved according to the reciprocal value from high to low.

Hong et. al. build a job recommender system based on user clustering [19]. At first, they look at some of the existing online job recommender systems (JRSs) from four different aspects: user profiling, recommendation strategies, recommendation output, and user feedback. They conclude that one of most important challenges lies in the design of recommendation strategies since different job applicants may have different characteristics. To address the aforementioned challenge, they develop an online JRS, *iHR*, which groups users into different clusters and employs various recommendation approaches

for different user clusters. As a result, *iHR* has the capability of choosing the appropriate recommendation approaches according to users' characteristics.

In another work [40], the authors propose a novel reciprocal recommendation method for job matching with bi-directional feedback. The method uses, bilateral messages between job seekers and recruiters, such as applying to a job, scout for a seeker, and reply to the offer, on the seekers-recruiters' user network as 'mutual feedback'. During job matching process, user agents, as delegate of their owners, send and receive those messages with each other. From those feedback messages, each user agent computes the popularity degree of its owner user: seeker or recruiter, and evaluation degree of each other from the popularity degree. Considering both the popularity and evaluation degrees, a similarity between the condition provided by its user and the profile of each candidate user is calculated, the agent dynamically updates a ranking list for recommendation of its owner user after every matching action.

Our research draws inspiration from some of the works mentioned above. These studies provided us a foundation with related ideas and established theories, in order to build a reciprocal recommender system for MOOCs. Such a system will assist learners with self-organization in MOOCs and possibly match students more effectively [42]. More specifically, our system takes into account one of the MOOC particularities: there is no extended history for learners' preferences, thus traditional collaborative filtering systems are not directly applicable. Moreover, the idea of reciprocity and peer recommendation is relatively new not only to the area of MOOC but also to the recommendation systems and gains more ground with many applications (like in group recommendation). With that note, next we review some of the related works in the area of effective group formation and collaboration in various contexts.

2.2.3 Group Formation Theories

Online courses in MOOCs offer the opportunity to create a highly social learning environment, characterized by participation and interactivity for both students and instructors. According to Kearsley in his work [27] , online learning

is as much a social activity as an individual one. However, the quality and frequency of interactivity on a MOOC discussion forum can vary depending upon many factors like type of course, importance of instructors in managing and regulating interactive content, demography of students etc. Social learning or learning as part of a group is an important way to help students gain experience in collaboration and develop important skills in critical thinking, self-reflection, and co-construction of knowledge [8].

A recent study [47] shows that different learning styles and cultures can be accommodated more easily because effective collaborative learning values diversity. Also, the skills gained from the experience of collaborative learning are highly transferable to team-based work environments [56]. The benefits of collaborative learning have been demonstrated in countless studies and several meta-analyses [26], [58], [59]. Compared to students taught traditionally, students taught in a manner that incorporates small-group learning achieve higher grades, learn at a deeper level, retain information longer, are less likely to drop out of school and acquire greater communication and teamwork skills [44]. Hence, the importance of forming groups of students in an online setting like MOOC cannot be more emphasized. However, it is essential that these groups have the right mix of students in order to extract the benefits of learning in groups as claimed by the existing studies.

In 1995, Hoppe introduced an intelligent tutoring system [20] that allowed the learners to initiate a group formation when they had a problem (a kind of learner-helper group). Based on the learners' models, the system displays a list of all potential peer learners that can help; the learner then selects a helper from the list, and the latter can accept or reject the invitation to help the learner. Parameters here were based on learning experience and competency criteria in the subject of the collaboration. In case of MOOCs, there are several parameters in terms of user attributes: *age*, *gender*, *location*, *grade* that must be optimized to create quality group of learners.

Inaba et. al. [23] introduced Opportunistic Group Formation (OGF) where an intelligent system detects the appropriate situation to start a collaborative

learning session and sets up a learning goal for the learner. The system takes into account the modeling of learning goals for each learner. Based on individual goals as well as the whole group, the system negotiates with the agents of all the learners in order to come to an agreement and to form a learning group so that each member of the group can obtain some educational benefit. In case of MOOCs, we have learners' preferences that can be aligned to match their overall learning goal. Users with similar mutual preferences can be thought of as having similar learning goals, hence it should make sense to cluster them in the same learning groups. All these studies make a good case to have effective learning groups on MOOCs which would not only help with better learning outcomes and mitigate dropouts but would also help prepare students gain a better understanding of the environment in which they will be working as professionals.

Although there are many studies mentioned above which confirm that group formation or collaborative learning is useful in the educational context, there are only few studies which focus on the method to be used for group formation. Most of the groups are formed without any criterion at all or by using simple random selection [22], this could lead to a well-known phenomenon: just a few groups are able to achieve high performance whereas the others are far from reaching the expected goals. This may occur when such a selection gathers students together, resulting in 'segregated' groups where all members exhibit desirable or undesirable characteristics [43] . To avoid such a problem, it is important to use group formation methods that not only seek the general performance of each group but also seek adequate results for individuals with different characteristics. In other words, the ideal situation should be having groups with members who are as similar among themselves as possible (inter-homogeneous), but also empowering the students' individual differences inside such groups (intra-heterogeneous).

Moreno et. al. developed a method based on a genetic algorithm (a metaheuristic inspired by the process of natural selection that belongs to the larger class of evolutionary algorithms) approach for achieving inter-homogeneous

and intra-heterogeneous groups [43]. An important feature of such a method is that it allows for the consideration of as many student characteristics as may be desired, translating the grouping problem into one of multi-objective optimization. This concept is similar to the requirements of grouping in MOOCs where learners have different characteristics in terms of their attributes, each of these attributes contribute towards forming inter-homogeneous and intra-heterogeneous groups hence it becomes a multi-objective optimization problem. In another similar study [3], the authors use a genetic algorithm to form groups of students with different levels of programming skills. Results show that the method used is capable in producing balanced groups, where each group consists of a mix of students with good, moderate and poor programming skills. However, a genetic algorithm although being effective in terms of quality of solution is computationally exhaustive [17], especially in cases where the volume of data is high and optimization involves a lot of constraints.

Graf et. al. propose a mathematical approach to form heterogeneous groups based on personality traits and the performance of students [15]. They make use of Ant Colony Optimization (ACO) algorithm in order to maximize the heterogeneity of formed groups. The Ant colony optimization is a swarm intelligence based probabilistic technique for solving computational problems which can be reduced to finding good paths through graphs. Although ACO has been successfully applied to solve combinatorial optimization problems, it still has some drawbacks such as stagnation behavior, long computational time, and premature convergence; these drawbacks become more evident when the problem size increases. Another swarm intelligence based algorithm called Particle Swarm Optimization (PSO) [28] has been used by researchers to find optimal solutions for clustering problems in less computational time. PSO optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. Van Der Merwe et. al. used PSO to cluster arbitrary data vectors [63]. On comparison with standard k-means clustering, the results showed that the PSO approaches have better convergence to lower quantization errors, and in general, larger inter-cluster distances and

smaller intra-cluster distances.

Furthermore, Cui et. al. propose a document clustering approach using PSO [10] which resulted in a higher compact clustering than with k-means. In another study [45], the authors use PSO for image clustering. The algorithm finds the centroids of a user specified number of clusters, where each cluster groups together with similar image primitives. Experimental results show that the PSO image classifier performs better than state-of-the-art image classifiers like K-means, Fuzzy C-means, K-Harmonic means and Genetic Algorithms in all measured criteria. For a large dataset, conventional PSO can conduct a globalized searching for the optimal clustering, but requires more iteration numbers and computation than the K-means algorithm does. The K-means algorithm tends to converge faster than the PSO algorithm, but usually can be trapped in a local optimal area. The hybrid PSO algorithm combines the ability of globalized searching of the PSO algorithm and the fast convergence of the K-means algorithm and avoids the drawback of both algorithms. This has been the idea behind using hybrid PSO to form students groups in MOOCs, as it generates better quality groupings and is also scalable.

The above studies and related work provided us with useful insights regarding recommender systems and group formation in the domain of education and other applicable areas. These studies influenced us greatly in formulating our own approach in building a reciprocal recommender and group formation strategy for effective collaboration. In the next couple of chapters, we look at : 1) an approach to build a reciprocal recommender system that generates a list of *top-N* like-minded recommendations for a given user and, 2) a modified PSO algorithm which generates student groups based on the factors of *inter-homogeneity* and *intra-heterogeneity*.

Chapter 3

Reciprocal Recommendation

This chapter is divided into four different sections: 1) Data: we look at the data used in our experiments and the description of the related data attributes, 2) Preference and Importance Modeling: here we describe the preference attributes of a user and the preference priority based on which the final recommendation ranking is generated, 3) Reciprocal Recommendation Algorithm: we delve into the details of the algorithm which matches users based on their preferences and generates the list of *top-N* recommendations and, 4) Experiments: lastly, we look at the experiments and analyze the results.

3.1 Data

The Data used in our research comes from the de-identified release from the first year (Academic Year 2013: Fall 2012, Spring 2013, and Summer 2013) of MITx and HarvardX courses on the edX platform [41]. These data are aggregate records (around 300k), and each record represents an individuals' activity in one edX course and contains many diverse information about the profile of the learner (e.g. age, gender, location, qualification, etc.) and the course activity (e.g. grade, forum posts, completed, etc.). For our analysis and without loss of generality, we selected records with attributes about age, location, qualification, gender and grade. Moreover, we enhance this information with synthesized data about learners' 'interests' and the current or past 'courses' undertaken by them. This information is not available via the mentioned dataset but is in line with the information available on MOOCs

for any learner profile, hence it deemed potentially useful to be used in our recommendation and group formation strategies.

A brief overview of the dataset attributes can be found in Table 3.1. The *user_id* is a numerical unique identifier for different learners, *age* of the learner is calculated using the year of birth obtained from the original dataset, *gender* is another binary attribute followed by *location*, which has information about the resident city of the learner. The *age* and *location* attributes have been further divided into levels, this is explained in the next subsection. Furthermore, the *qualification* attribute has been divided into 5 levels: less than secondary, secondary, bachelors, masters and doctorate. The *interest* attribute contains one or more values about learners' interest. Lastly, the *courses* attribute has information about the current and past courses undertaken by the learners. A sample of our dataset used to build our reciprocal recommender can be seen in Table 3.2.

Table 3.1: Dataset Attribute Description

Attribute	**Short**	**Type**	**Comment**
user_id	id	Numeric	Unique identifier
age	age	Numeric	Calculated using year of birth
gender	gen	Binary	M(ale)/F(emale)
location	loc	Categorical	City of the learner
qualification	qua	Ordinal	learners' qualification
interests	int	Hierarchical, Categorical, Multi-Value	Info about learners' interests
courses	crs	Categorical, Multi-Value	List of current/previous courses

Table 3.2: Dataset Sample

id	**age**	**gen**	**loc**	**qua**	**int**	**crs**
1	32	M	Frankfurt	Doctorate	ML	machine learning, java, python
2	28	M	Los Angeles	Bachelors	AI	java, python
3	27	F	Edmonton	Bachelors	Science	python, sociology
4	22	F	Las Vegas	Secondary	Soccer, AI	history, general studies

This was an overview of the data used in building the 'reciprocal recommender'. The data used in 'group formation' is slightly different, more details on this is provided in the next chapter.

3.2 Preference and Importance Modeling

When users sign up on a MOOC platform, we ask them to provide their preferences for the above mentioned attributes, based on which their recommendations would be generated. User preferences are based on value ranges for attributes in Table 3.1 and can include none, one or more (even all) of these attributes. A description of the value ranges of preferences for each of the attributes is mentioned below:

- **Age:** users can select an age range that they prefer learners' from, using these 5 choices: less than 20, 20-25, 25-30, 30-35, 35 and above.
- **Gender:** users can select between male or female gender options.
- **Location:** users have an option to select learners from the *same city* (if they prefer meeting them in person) or from the *same country or timezone* (in order to facilitate communication).
- **Qualification:** users can select one or more qualifications out of the five levels available: less than secondary, secondary, bachelors, masters and doctorate
- **Interests:** users can define their own interest preference which might or might not be similar to their own interest.

A sample of user preferences can be seen in Table 3.3. It must be noted that not all five attributes are required to be defined by a user for their preference profile. One or more (but not all) of these attributes can be left empty, at which point the algorithm simply ignores these in the recommendation process as it considers them irrelevant.

Moreover, users can further define whether they have some preferences that are more important to them i.e. if they have a priority for their preferences (highlighted in bold in the preference Table 3.3). For instance, looking at preference p_4 (preference profile of user_id:4) in Table 3.3, we can tell that the user prioritizes *location* and *qualification* over other attribute preferences.

This means that the recommendation list for this user should ideally have those users who have the same location as him, with a qualification less than equal to a 'bachelors' degree.

Table 3.3: Sample of User Preferences: x refers to unfilled preferences and **bold** denotes important preferences

pref	age	gen	loc	qua	int
p_1	**30-35**	**M**	same city	>= Masters	x
p_2	x	x	x	Bachelors	Football
p_3	**25-30**	F	x	x	x
p_4	<=25	x	**same timezone**	<=**Bachelors**	x

3.3 Recommendation Algorithm Description

In the next subsections, we discuss our recommendation algorithm in detail. In summary, at first we select users with similar course background (this is optional) and then build a similarity matrix which has the compatibility scores (based on preference) between these users, using which they are ranked. Lastly, we re-rank the users based on their preference priority.

3.3.1 Selection Phase

Since the number of new students signing up for a MOOC might be very large, we can build a query to select only those users that share at least 50% (this can be flexible) of the current or previously registered courses with the user whose recommendation is to be generated. In case the query yields insufficient number of users (less than the number of recommendations to be generated), the query can be further relaxed to select users who share at least one course. Furthermore, the recommendations are now generated from this selected pool of users. It is to be noted that this phase is optional. In cases where the registered user on MOOC is 'new' (*cold start problem*) [53], he might not have any course background. In that case we can skip the selection phase and move on to build the similarity matrix based on his user preferences.

For instance, in Table 3.2, when the 'selection phase' is run for user *id:1*,

then only user *id:2* will be selected (since they share at least 50% of courses). However, if the query is relaxed to select users with at least one matching course, in that case users [*id:2* and *id:3*] will be selected. In scenarios where a user does not have any history of current or previously registered courses, we consider the entire set of learners as a pool for generating recommendations.

3.3.2 Building Similarity Matrix

Given the preferences of a user, we compute the "distance" or "compatibility" of this user, with every other 'selected' user based on his/her preferences and the characteristics or attribute values of the other users. It is to be noted that, "the lower the distance score, the greater the similarity". For instance, using the data sample in Table 3.2, distance of a user (with id=1) to other users in the table is computed as follows:

- **Ordinal Variables** (age, qualification): Preferences for *age* and *qualification* attributes are divided into levels in such a way that adjacent levels have a distance of 1. The 5 levels for *age* attribute preferences are: "less than 20", "20-25", "25-30", "30-35", "35 and above". Similarly, the 5 levels for *qualification* attribute preferences are: "less than secondary", "secondary", "bachelors", "masters" and "doctorate". In case of *age* levels, we code: "less than 20" as 1, "20-25" as 2, "25-30" as 3, "30-35" as 4, "35 and above" as 5. Similarly, in case of *qualification*, we code: "less than secondary" as 1, "secondary" as 2, "bachelors" as 3, "masters" as 4 and "doctorate" as 5. Once the *age* distance d_{age} or the *qualification* distance d_{qua} between users is calculated, the distance is then normalized in range $[0-1]$ by dividing it by the maximum distance possible (which is $5-1=4$, for both of these attributes).

  On modelling the distances d_{age} and d_{qua} for *age* and *qualification* preference attributes respectively for user *id:1* (see Table 3.2) based on his/her preferences defined in Table 3.3 (p_1), we get the following:

  $d_{age}(userid{:}1, userid{:}2) = 1/4 = 0.25$ (as they differ by one *age* range)

$d_{age}(userid{:}1, userid{:}4) = 2/4 = 0.50$ (as they differ by two *age* range)

$d_{qua}(userid{:}1, userid{:}2) = 1/4 = 0.25$ (as they differ by one *qua* range)

$d_{qua}(userid{:}1, userid{:}4) = 2/4 = 0.50$ (as they differ by two *qua* range)

From the distance calculated based on *age* and *qualification* attributes alone, we can see that *userid:2* is the best match for *userid:1* as they have the least 'distance' between them, hence they are more 'compatible'.

- **Nominal Variables** (gender, location): Preferences for *gender* and *location* attributes are mapped to a binary distance metric. For instance, if the *gender* of two users are the same, then the distance *'d'*$_{gen}$ is 0, otherwise 1. Similarly, if the *location* preference of a user matches the *location* attribute value of other user, the distance *'d'*$_{loc}$ is 0, otherwise it is 1.

  On modelling the distance d_{gen} and d_{loc} for *gender* and *location* preference attributes respectively for *userid:1* (see Table 3.2) based on his/her preferences defined in Table 3.3 (p_1), we get the following:

  $d_{gen}(userid{:}1, userid{:}2) = 0$ (as *gender* preference of user *id=1*, 'M', matches the *gender* attribute of user *id=2*, 'M')

  $d_{gen}(userid{:}1, userid{:}4) = 1$ (as *gender* preference of user *id=1*, 'M', does not match the *gender* attribute of user *id=4*, 'F')

  $d_{loc}(userid{:}1, userid{:}2) = 1$ (as *location* preference of user *id=1*, 'same city', does not match the *location* attribute of user *id=2*, 'Los Angeles')

  $d_{loc}(userid{:}1, userid{:}4) = 1$ (as *location* preference of user *id=1*, 'same city', does not match the *location* attribute of user *id=4*, 'Las Vegas')

  In this case, both users [*id:2 and id:4*] have the same distance '1' for *location* with respect to *userid:1*, however they differ in distance for the *gender* attribute. Using the sum of distances calculated based on *gender* and *location* attributes, we can see that *userid:2* is the best match for *userid:1* as they have the least 'distance' between them, hence they're more 'compatible'.

- **Hierarchical Variables** (interests): For preference attributes that come from a hierarchy there is a similarity measure based on the hierarchy tree, such as the Rada measure [52]. This measure was founded on the basis that we can calculate the similarity based on the hierarchical links such as 'IS-A'. To calculate the similarity of two concepts in a hierarchy tree, we must calculate the number of the minimal arcs which separate them. This measure, based on the edge counting between nodes by the shortest way, presents a method to evaluate the semantic similarity in a hierarchical structure tree. The hierarchy we used for *interests* of users is based on WordNet [39] and the similarity measure used is based on the Wu and Palmer method [66]. It calculates relatedness by considering the depths of the two synsets in the WordNet taxonomies, along with the depth of the LCS (Least Common Subsumer). The formula is given as:

$$score = 2 * depth(lcs)/(depth(s1) + depth(s2)) \tag{3.1}$$

  This means that $0 < score <= 1$. The score can never be zero because the depth of the LCS is never zero (the depth of the root of a taxonomy is one). The score is one if the two input synsets are the same. Since we are implementing our system in a distance measure (and not similarity) the final value of distance between the interests is $[1 - score]$.

  For instance, a user with interest in 'football' will be more compatible with another user with interest in 'volleyball', in comparison to a user whose is interested in 'computer science'. The Wu and Palmer similarity ($wup\_similarity$) score for these instances are given below.

  $wup\_similarity(football, volleyball) = 0.80$

  $wup\_similarity(football, computer\_science) = 0.38$

Finally, the 'compatibility score' of a user x with any other user y is the mean of the attribute distances:

$$compatibility\_score(x, y) = \frac{\sum_{i=1}^{N} d_i(x, y)}{N} \tag{3.2}$$

where d_i is the distance for attribute i between users x and y and N represents the total number of attributes (in our case $N = 5$). For instance, the 'compatibility score' of user 3 (*id=3*) with the other users can be computed as follows:

$compatibility\_score(userid : 3, userid : 1) = \frac{1/4+1}{5} = 0.25$, (as *age* range difference is 1 and *gen* difference is 1)

$compatibility\_score(userid : 3, userid : 2) = \frac{0+1}{5} = 0.2$, (as *age* range is same, but *gen* is different)

$compatibility\_score(userid : 3, userid : 4) = \frac{1/4+0}{5} = 0.05$, (as *age* range difference is 1 and *gen* is same)

Based on the 'compatibility' score calcuated above, we can say that *userid:3* is most compatible with *userid:4*. Table 3.4 below shows the 'Similarity Matrix' with compatibility scores between all users in the sample dataset (Table 3.2). The cells marked as 'x' means self-compatibility, which is omitted as we are looking for other compatible users for each given user.

Table 3.4: Similarity Matrix

user_id	1	2	3	4
1	x	0.3	0.5	0.6
2	0.2	x	0	0.15
3	0.25	0.2	x	0.05
4	0.45	0.1	0.3	x

Once we have the similarity matrix, we can rank users based on their mutual likeness, which is calculated by taking the 'harmonic mean' of their compatibility scores. This is shown in the next subsection.

3.3.3 Ranking Recommendations by Importance

After the user preferences and the compatibility scores are computed, the list of recommended users generated for user x are as follows: Every user y will receive a compatibility score that reflects how much the preferences of user x match with the attributes of user y, and how much the preferences of user y match with the attributes of user x. We call this measure 'reciprocal score'.

The reciprocal score between user x and user y is the harmonic mean of the compatibility scores between them. It is to be noted that compatibility scores of zero are replaced by a small value like 0.001 in order for the harmonic mean to be computed as a negligible value would still mean high compatibility. A ranking is generated using the reciprocal scores (harmonic mean) where it is checked if the preference priority for attributes as denoted by the user is satisfied or not.

For instance, the reciprocal score for user *id:3* is shown in Table 3.5. Column $p(3, y)$ indicates the similarity of user *id:3* with all other users y and column $p(y, 3)$ indicates the similarity of all other users y with the user *id:3*. Note that the reciprocal score is symmetric, i.e. *y's* score in the recommendation list for x is the same as *x's* score in the list for y. However, as the lists contains only the top-N recommendations, user y may be in the top-N recommendations for user x but the opposite may not be true.

Table 3.5: Reciprocal Score for user *id:3*

y	**p(3,y)**	**p(y,3)**	**harmonic_mean**
1	0.25	0.5	0.333
2	0.2	0.001	**0.002**
4	0.05	0.3	0.086

Given the reciprocal scores in Table 3.5, the list of top-3 recommendations for user *id:3* will be: [2, 4 and 1]. Furthermore, user *id:3* has noted preference priority for *age* attribute (see bold values in Table 3.3). Since user *id:2* satisfies this criterion, it will remain at the topmost position and users *id:4* and *id:1* will follow. If this was not the case, then a re-ranking of recommended users is done based on the preference priority of the user for the given attributes.

3.4 Experiments

We conducted experiments with sampled dataset of 100, 1000 and 5000 user records. The algorithm was run on a computer with 2.7 GHz Intel Core i5 processor and 8 GB RAM. In the next few subsections, we look at some of

the evaluation metrics used to validate our findings and analyze our results in terms of measuring precision, recall and discounted cumulative gain.

3.4.1 Evaluation Metrics

As seen in the related work chapter, not much research has been carried out in the field of reciprocal recommender systems, thus there is limited exploration of how to effectively evaluate the recommendation results. The goal of our current work is to primarily explore the role of reciprocity in the formulation of recommendations for MOOCs. It should be noted that an actual evaluation of a (reciprocal) recommender system requires on-line deployment and integration with one of the existing MOOC platforms. Since this was not possible in our case, we had to build measures based on the data available.

Furthermore, the recommendations generated using traditional recommender systems (non-reciprocal) would yield a different ranking from our system as those do not take into account the 'reciprocity' factor. More specifically, our results indicate that our system performs better than the baseline model, this is described in detail in the 'experiments' subsection.

For a reciprocal system (like the one in our case) we need to define 'what is a successful recommendation'. We say that, "a learner y is a successful (reciprocal) recommendation (out of the K-total) for learner x, if and only if x is also in the top-K recommendations of learner y". This condition factors the reciprocity element which is essential to measure the performance of a reciprocal system like ours. Using this idea, we modify the definitions of precision and recall [55] for each learner as follows: "In order to compute the precision for learner x, we divide the number of successful recommendations of learner x by the total number of generated recommendations (i.e. K) for leaner x". "Similarly, in order to compute the recall for learner x, we divide the number of successful recommendations of learner x by the total number of learners that have x in their top-K generated recommendation list". These definitions can be formalized in the following equations:

$$P_x = \frac{N_x}{K} \tag{3.3}$$

$$R_x = \frac{N_x}{N*_x} \tag{3.4}$$

where:

- P_x is the precision for learner x,
- R_x is the recall for learner x,
- N_x is the number of successful recommendations for learner x (as defined before),
- K is the total number of recommendations generated for learner x,
- $N*_x$ is the number of learners that have x in their recommendation list

The total precision and recall of the dataset based on the recommendation algorithm is defined as follows:

$$P = \sum_{i=1}^{M} \frac{P_i}{M} \tag{3.5}$$

$$R = \sum_{i=1}^{M} \frac{R_i}{M} \tag{3.6}$$

where:

- P_i and R_i are the precision and recall respectively for learner i (as declared previously),
- M is the total number of learners

Moreover, in order to evaluate the rankings of the algorithm, we utilize a modified definition of the Discounted Cumulative Gain (DCG) [25], a popular measure of ranking quality. DCG originates from information retrieval where ranking positions are discounted logarithmically. Since for our system,

we only care about the rank alignments and not the relevance of ranking positions (since the algorithm already generates relevant ranking), hence we do not require the logarithm discounting. When applied to our case, 'DCG' is the measure of 'reciprocity' or 'rank alignment'. In other words, a perfect rank alignment between users of generated recommendations is when - "if for all learners i, present at a position j in the list of top-N recommendations of learner u, u is also present at the same position j in the list of top-N recommendations of i".

Assuming each learner u has a "gain", g_{ui} from being recommended another learner i, then the average Discounted Cumulative Gain (DCG) for the recommendation list of K learners is defined as follows:

$$DCG = \frac{1}{M} \sum_{u=1}^{M} \frac{\sum_{j=1}^{K} g_{ui_j}}{S} \tag{3.7}$$

where:

M is total number of learners,

S is the number of successful recommendations,

j denotes the position in the ranking list,

g_{ui_j} is the gain of learner i (in position j) for learner u.

Division by the number of successful recommendations guarantees that maximum DCG will be 1, provided that a user has successful recommendations, otherwise it is automatically 0.

The gain g_{ui} is considered to be 0 if learner u is not in the top-K recommendation list for learner i (no gain for the reciprocal recommendation system here) and if is present, then the gain is defined as follows:

$$g_{ui} = \frac{1}{1 + |dif_{ui}|} \tag{3.8}$$

where dif_{ui} is the difference in positions between the ranking of user i in the recommendation list of user u and the ranking of user u in the recommendation list of user i. If the difference in ranking is zero, the gain g_{ui} results in 1, otherwise the gain is discounted.

Finally, DCG can be divided by the ideal DCG for the recommender system which would lead to the normalized discounted cumulative gain (NDCG). Ideal DCG is 1 provided that all users have at least one successful recommendation (each user can have a maximum DCG of 1, so divided by the number of users that gives 1), otherwise it is a reduced value.

$$NDCG = \frac{DCG}{DCG*} \tag{3.9}$$

For example, consider the following Table 3.6 of six learners: [1, 2, 3, 4, 5, 6] with successful recommendations highlighted in circles.

Table 3.6: Ranked Recommendations, K=3

rank/learner	1	2	3	4	5	6
1	2	③	①	6	1	3
2	③	4	②	⑤	④	2
3	4	5	④	③	6	1
4	5	1	5	1	2	4
5	6	6	6	2	3	5
Precision	0.33	0.33	1.00	0.67	0.33	0.00
Recall	0.33	0.33	0.75	0.50	0.50	0.00
DCG	0.50	0.50	0.67	1.00	1.00	0.00

Overall precision for this system is 0.44, recall is 0.40 and the NDCG is 0.73 (DCG is 0.61 and DCG* is 0.83).

In the next subsection, we analyze the results obtained using different data samples, based on the metrics defined above.

3.4.2 Results

As mentioned previously, we sampled our data using the MITx-Harvardx dataset. We select samples of [100, 1000, 5000] data records for our experiments . From each of the three samples we rank users by comparing their reciprocal scores and recommend the top-N [5,10,15,20] users in the list. The objective of our research is not only to provide the best possible reciprocal recommendation but also to make sure that the solution is scalable, this is especially important since the number of students who register on MOOC has been growing exponentially every year [54]. Our 'precision', 'recall' and

'DCG' scores are compared against the 'baseline', wherein the reciprocity factor was not accounted for. The 'baseline' model simply builds the list of top-N recommendations without looking at reciprocity, very similar to a traditional recommender system. As per the evaluation metrics defined in the previous section, we see the results obtained for these sampled datasets in the next subsections.

Recommendation Precision with/without "Selection Phase"

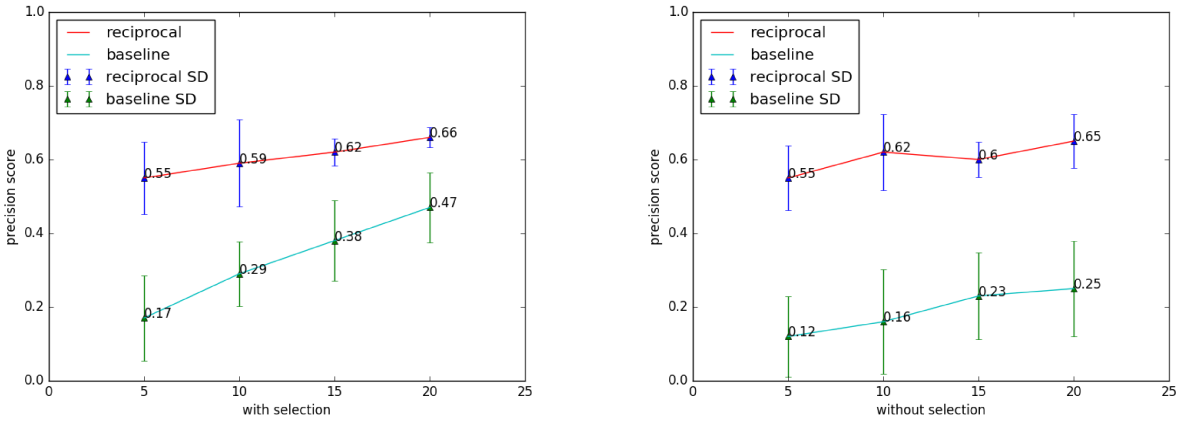
Precision is a measure of the number of "successful recommendations" by the total number of recommendations (top-N). We computed the mean precision value of the reciprocal recommender system by averaging the precisions of all the users in the data sample for top-N recommendations. As expected, the precision increases with the value of 'N', which means that if a learner y is present in the top-N recommendation list for learner x, then the chances that x is also present in the recommendation list of y increases with increasing value of 'N'.

We also measure the scalability of the solution by comparing the performance time for datasets with [100, 1000, 5000] records. The evaluation process was run with and without the "selection phase" of the algorithm. As expected, the solution has better performance in terms of 'execution time' when 'selection' is applied as in this case it filters down the relevant users to recommend from. Moreover, the precision value with 'selection' is slightly less than without 'selection' for any top-N recommendation. This is because we're restricting the number of users to generate recommendations from, when 'selection' is applied. The mean precision values and standard deviation (SD) for top-N recommendations with/without 'selection' can be seen in Figure 3.1. It can be deduced that the precision scores for 'reciprocal' model far exceeds the scores for 'baseline', across all values of top-N recommendations.

The average precision scores for *top5 to top-20* recommendations for 'reciprocal' model 'with selection' are [0.55, 0.59, 0.62, 0.66] with standard deviation [.098,.118,.036,.027] respectively. However, the corresponding average precision scores for *top5 to top-20* recommendations for 'baseline' model 'with se-

lection' are [0.17, 0.29, 0.38, 0.47] with standard deviation [.116,.088,.109,.095] respectively.

Similarly, the average precision scores for *top5 to top-20* recommendations for 'reciprocal' model 'without selection' are [0.55, 0.62, 0.60, 0.65] with standard deviation [.088,.103,.048,.073] respectively. However, the corresponding average precision scores for *top5 to top-20* recommendations for 'baseline' model 'without selection' are [0.12, 0.16, 0.23, 0.25] with standard deviation [.109,.142,.117,.129] respectively.



(a) with Selection

(b) without Selection

Figure 3.1: Precision Graph

Recommendation Recall with/without "Selection Phase"

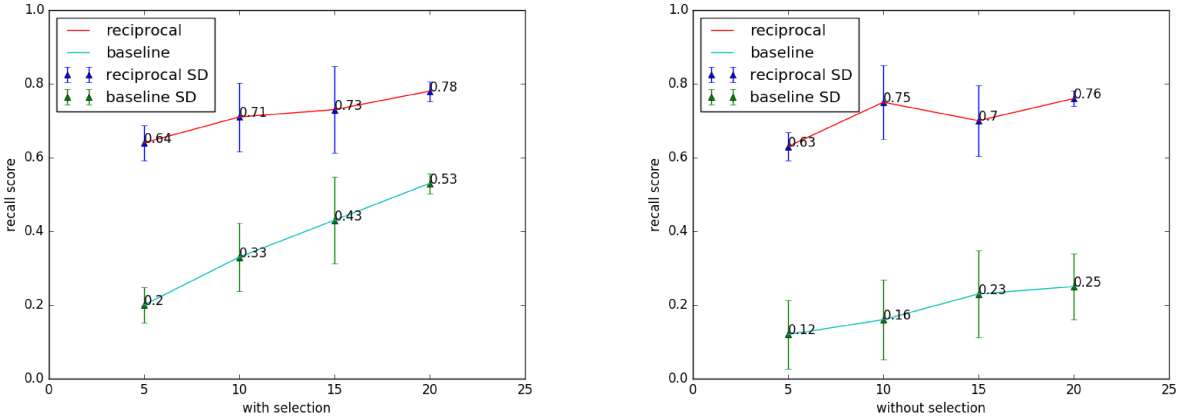
Recall is a measure of the number of "successful recommendations" by the total number of learners that have the given learner in their top-N recommendation list. We computed the mean recall value of the reciprocal recommender system by averaging the recall of all the users in the data sample for top-N recommendations. As expected, the recall increases with the value of 'N', which means that if a learner y is present in the top-N recommendation list for learner x, then the chances that x is not present in the list of learners who are not part of the recommendation list of x, decreases with increasing value of 'N'.

We also measure the scalability of the solution by comparing the performance time by running the evaluation for datasets with [100, 1000, 5000]

records. The evaluation process was run with and without the "selection phase" of the algorithm. As expected, the solution has better performance in terms of 'execution time' when 'selection' is applied as in this case it filters down the relevant users to recommend from. Moreover, the recall value with 'selection' is slightly less (but still comparable) than without 'selection' for any top-N recommendation. This is because we're restricting the number of users to generate recommendations from, when 'selection' is applied. The mean recall values and standard deviation (SD) for top-N recommendations with/without 'selection' can be seen in Figure 3.2. It can be deduced that the recall scores for 'reciprocal' model far exceeds the scores for 'baseline', across all values of top-N recommendations.

The average recall scores for *top5 to top-20* recommendations for 'reciprocal' model 'with selection' are [0.64, 0.71, 0.73, 0.78] with standard deviation [.048,.093,.117,.027] respectively. However, the corresponding average recall scores for *top5 to top-20* recommendations for 'baseline' model 'with selection' are [0.20, 0.33, 0.43, 0.53] with standard deviation [.109,.083,.097,.115] respectively.

Similarly, the average recall scores for *top5 to top-20* recommendations for 'reciprocal' model 'without selection' are [0.63, 0.75, 0.70, 0.76] with standard deviation [.039,.101,.095,.021] respectively. However, the corresponding average recall scores for *top5 to top-20* recommendations for 'baseline' model 'without selection' are [0.12, 0.16, 0.23, 0.25] with standard deviation [.092,.108,.117,.089] respectively.



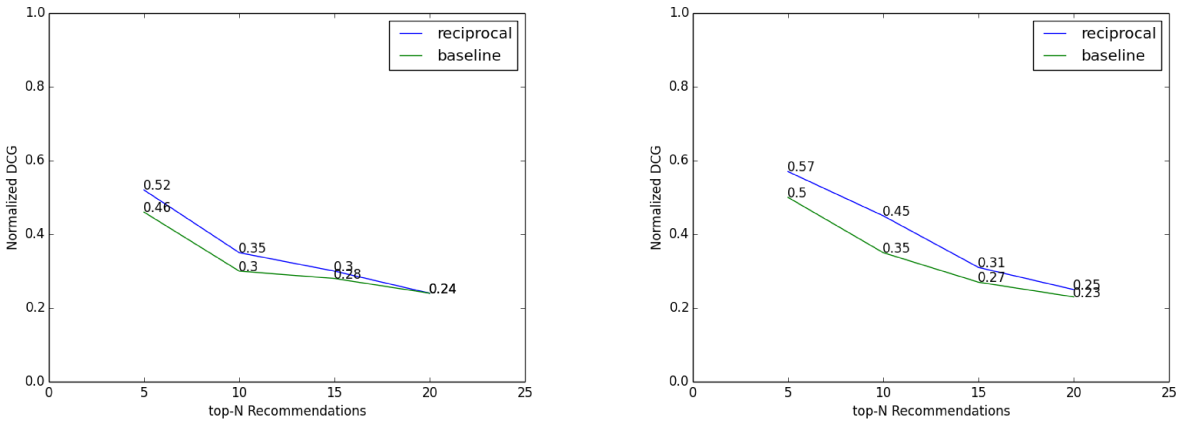
(a) with Selection

(b) without Selection

Figure 3.2: Recall Graph

Discounted Cumulative Gain with/without "Selection"

Discounted Cumulative Gain (DCG) is the measure of 'reciprocity' or 'rank alignment'. In other words, a perfect rank alignment is when - "for all learners y, present at a position p in the list of top-N recommendations of learner x, if x is also present at the same position p in the list of top-N recommendations of y". The value of 'DCG' is 1 in case of perfect rank alignment and 0 when there are no 'successful recommendations'. We calculate the 'Normalized DCG' using the formula given in equation 3.9. As shown in Figure 3.3, 'NDCG' value decreases if the 'top-N' recommendations increase. This makes sense because with higher number of recommendations, the difference in ranks for two positions in the recommendation list will increase, thereby resulting in an overall decrease in 'gain'. The 'NDCG' values for top-N recommendations with/without 'selection' can be seen in Figure 3.3. The reciprocal model slightly outperforms the 'baseline' model. While this suggests that the reciprocal model is only a little better than the baseline model when it comes to rank alignments, it should also be noted that rank alignments has less significance when it comes to 'reciprocity', because if a user x satisfies all the preferences of user y, it might not be true vice-versa. Hence, their rank alignments will be different.



(a) with Selection

(b) without Selection

Figure 3.3: NDCG Graph

Measuring Scalability with/without "Selection"

The tremendous growth of users in MOOCs in the recent years poses some key challenges for any recommender system. It should not only be able to generate high quality recommendations but also be able to perform better in terms of execution time for millions of users. In this subsection, we look at the performance of our 'reciprocal recommender' measured in terms of 'minutes' taken to generate recommendations.

We ran five simulations on different dataset for each of the [100,1000,5000] data samples. The average time to generate top-20 recommendations for each of these data samples is shown in Fig 3.4. The execution time in 'minutes' for generating *top-20* recommendation using 'selection' for [100, 1000 and 5000] data records are [62.1, 157.7 and 454] respectively. However, understandably the execution time increases to [139.9, 355.7 and 1080.5] respectively when the 'selection' is not applied. It is to be noted that execution time will depend several factors like dimension of data, number of data records, sparsity of data etc. Hence, the execution time might change given any of the above factors are changed.

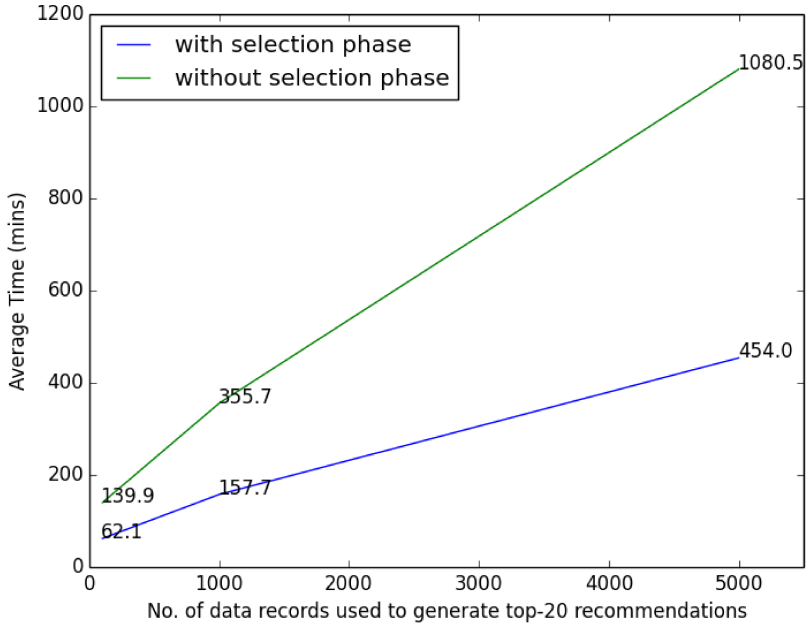


Figure 3.4: Average Execution Time for top-20 recommendations

The results show that the our system performs better than the baseline system on the measures of precision, recall and discounted cumulative gain. Moreover, these tests have also shown that the system is very robust to take into account the large datasets for top-N recommendations. This makes for a promising case for the algorithm to be deployed within an actual MOOC to get a more realistic estimate of its performance. In the next chapter we describe the group formation strategy for automatically generating small groups of learners using an optimization algorithm.

Chapter 4

Group Formation

This chapter is divided into four different sections: 1) Data: we look at the data used in our experiments and description of the data attributes, 2) Data Attributes Modeling: here we describe each of the data attributes and they way they are tuned to fit into the grouping criteria, 3) Group Formation Algorithm: we delve into the details of building a hybrid particle swarm optimization algorithm which groups users based on a predefined set of attributes and, 4) Experiments: lastly, we look at the experiments and analyze the results.

4.1 Data

The Data used in 'group formation' is similar to the one used in our 'reciprocal recommender system' except that we added the 'grade' attribute to the set of existing data attributes like age, gender, location, qualification, interests etc.

A brief overview of the dataset attributes can be found in Table 4.1. The *user_id* is a numerical unique identifier for different learners, *age* of the learner is calculated using the year of birth obtained from the original dataset, *gender* is another binary attribute followed by *location*, which has information about the resident city of the learner. Furthermore, the *qualification* attribute has been divided into 5 levels: less than secondary, secondary, bachelors, masters and doctorate. The *interest* attribute contains one or more values about learners' interest. Lastly, the grade attribute has information about users' grade between 0(Min) and 1(Max). A sample of our dataset can be seen in Table 4.2.

Table 4.1: Dataset Attribute Description

Attribute	Short	Type	Comment
user_id	id	Numeric	Unique identifier
age	age	Numeric	Calculated using year of birth
gender	gen	Binary	M(ale)/F(emale)
location	loc	Categorical	City of the learner
qualification	qua	Ordinal	5 levels
interests	int	Hierarchical, Categorical, Multi-Value	Info about learners' interests
grade	grade	Numeric	graded between 0(Min) and 1(Max)

Table 4.2: Dataset Sample

id	age	gen	loc	qua	int	grade
1	32	M	Frankfurt	Doctorate	ML	0.1
2	28	M	Berlin	Secondary	AI	0.4
3	27	F	Edmonton	Bachelors	Science	0.8
4	22	F	Las Vegas	Masters	Soccer, AI	1.0

4.2 Data Attributes Modeling

Users signing up on a MOOC platform have varied attributes which have to be aligned or modeled based on the proposed algorithm to form effective learning groups. Each attribute in Table 4.1 has its own underlying structure in the way it contributes towards the grouping logic. A description of how each attribute is modelled is mentioned below:

- **Age:** age range of the users' are segregated in these five levels: less than 20, 20-25, 25-30, 30-35, 35 and above.
- **Location:** location attribute is categorized into 3 options: *same city*, *same country* or *same timezone*
- **Gender:** male or female gender options.
- **Qualification:** the *qualification* attribute has been divided into 5 levels: less than secondary, secondary, bachelors, masters and doctorate
- **Interests:** the *interest* attribute contains one or more values about learners' interest.

- **Grade:** the *grade* attribute has the average 'grade' of a learner from previous courses, between 0(min) and 1(max).

A sample of data vectors can be seen in Table 4.3. The 'x's in the table represent null value. It must be noted that not all six attributes are required to be used for any kind of grouping. Our proposed algorithm is flexible enough to take one or more of these attributes for group formation.

Moreover, we can tune the way each of these attributes contribute in group formation in terms of *intra-group heterogeneity* and *inter-group homogeneity*. For instance, a reasonably heterogeneous group would refer to a group where student-grades reveal a combination of low, average and high student-grades. This is justified by the recommendation of Slavin [57] who proposed that students should work in small, mixed-ability groups.

Therefore, it makes sense that each of the generated groups have a good mix of students with high, average and low grades. A group having majority of students with high grades is undesirable and so is a group having most students with low grades. Hence, it is necessary that grade distribution is even across all groups i.e. the average grades of students across all groups should be same (*inter-group homogeneity*) while maintaining that within each groups the grades are diverse (*intra-group heterogeneity*).

Another factor to consider into account during group formation in collaborative learning is the interest of the group members, since the interest has the potential to change the involvement of individuals in learning [35]. A group with common interests will have more interactivity and discussions which is likely to make the learning process more engaging. Same can be said about the 'location' attribute. Students residing in the *same city, country or timezone* will be able to collaborate better due to minimal time differences as opposed to students living across different continents having major time difference between them.

To understand how a group would be heterogeneous regarding the level of 'grade', but with the same 'interests', see Figure 4.1 [61]. Each square in the figure represents a learner, the number inside the square represents

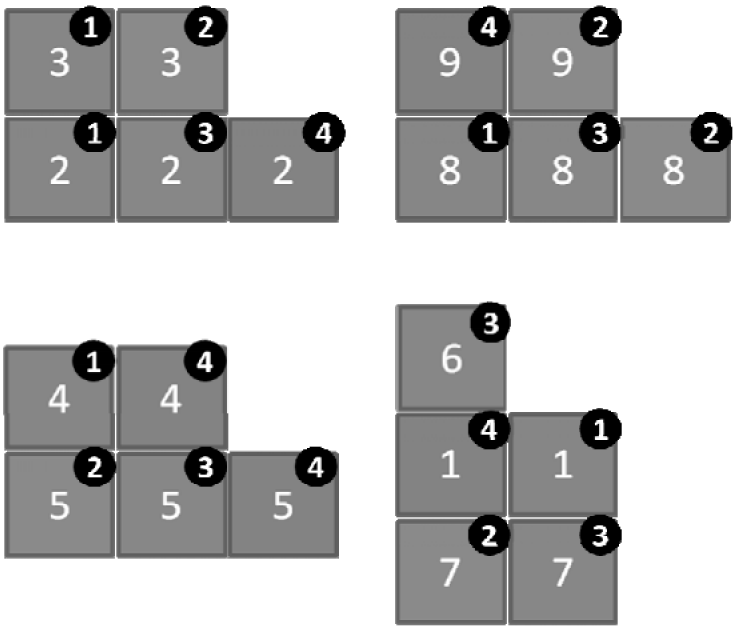


Figure 4.1: Groups formed with different 'grades' but same 'interests'

users' 'interest' and the number inside the small circle represents the 'grade' of each learner. It can be seen that users are grouped together in such a way that their 'interests' (squares in the figure) are similar, however their 'grades' (circles in the figure) being diverse within a group. As mentioned before, the average grade across all groups should be same in order to foster equal learning opportunities for students within different groups.

Table 4.3: Sample Data Vectors

userid	**age**	**gen**	**loc**	**qua**	**int**	**grade**
1	30-35	M	same city	>= Masters	x	0.1
2	x	x	x	Bachelors	Football	0.4
3	25-30	F	x	x	x	0.8
4	<=25	x	same timezone	<=Bachelors	x	0.1

Similar to the 'grade' attribute, the 'age', 'gender' and 'qualification' of students in a group should be diverse, this would enable the students to learn from each others' differences, thereby making the group more heterogeneous.

4.3 Algorithm

In the next subsections, we'll discuss our grouping algorithm in detail. In summary, at first we use a modified K-means grouping algorithm [16] to generate groups whose centroids are used to seed the initial swarm of particles. Next, we use a hybrid 'Particle Swarm Optimization' technique to generate the final group of learners. The general schema of the algorithm can be seen in figure 4.2

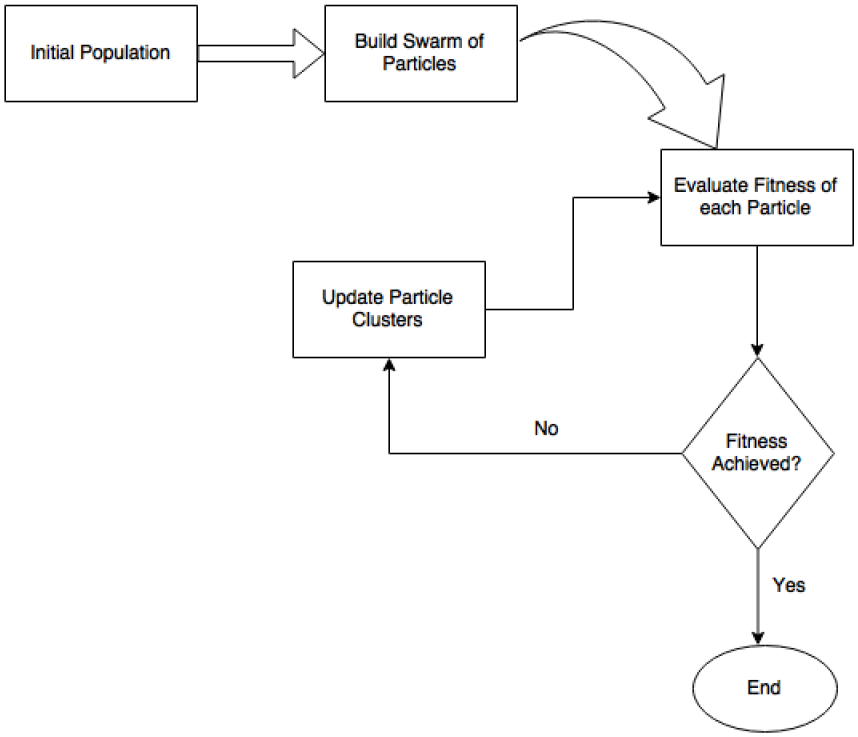


Figure 4.2: General Schema of Hybrid PSO

4.3.1 Modified K-means

One of the most important components of any clustering algorithm is the measure of similarity between different clusters generated using the algorithm. Using a traditional K-means algorithm [16], you can generate predefined number of clusters based on the Euclidean distance similarity measurement. All the data points within a particular cluster have small Euclidean distances between them, which means they're very similar. In contrast, the data vectors

between two different clusters will have large Euclidean distances, indicating that they're less similar. Every cluster has a 'centroid' or the 'mid-point' which is the mean of all the data vectors within that cluster. Below are a few notations which we use to describe K-means:

- N_d - denotes the input data dimension, i.e. the number of attributes of each data vector
- N_m - denotes the total number of data vectors to be clustered
- N_c - denotes the number of centroids of the clusters, i.e. the number of clusters to be generated, provided by the user
- z_p - denotes the p-th data vector
- m_j - denotes the centroid of cluster j
- n_j - denotes the number of data vector in cluster j
- C_j - subset of data vectors that form cluster j

Using the above mentioned notations, the modified K-means algorithm is summarized below:

1. At first, we randomly initialize N_c cluster centroids using the data vectors
2. **Repeat**
   (a) In traditional k-means, each data vector is assigned to that cluster whose centroid has the least Euclidean distance with the data vector as per the equation below:

$$d(z_p, m_j) = \sqrt{(\sum_{k=1}^{N_d}(z_pk - m_jk)^2)} \tag{4.1}$$

   where k represents the dimension.

   However in our case, we handle the distance calculation for each data vector in a different way. We develop a scoring system wherein

the distance between each attribute of a data vector to that of its corresponding attribute of all centroids is calculated. The data vector is then assigned to that cluster where it has the least distance with its corresponding centroid. A sample case for distance calculation is shown below:

i. The *age* and *qualification* attributes are divided into levels in such a way that adjacent levels have a distance of 1, which is shown further. The 5 levels for *age* attribute are: "less than 20", "20-25", "25-30", "30-35", "35 and above". Similarly, the 5 levels for *qualification* attribute are: "less than secondary", "secondary", "bachelors", "masters" and "doctorate". In case of *age* levels, we code: "less than 20" as 1, "20-25" as 2, "25-30" as 3, "30-35" as 4, "35 and above" as 5. Similarly, in case of *qualification*, we code: "less than secondary" as 1, "secondary" as 2, "bachelors" as 3, "masters" as 4 and "doctorate" as 5. Once the *age* distance or the *qualification* distance between data vector and centroid is calculated, the distance is then normalized in range $[0-1]$ by dividing it by the maximum distance possible (which is $5-1=4$, for both the attributes).

   On modelling the distance d_{age} and d_{qua} for *age* and *qualification* attributes respectively, for user *id:1* (see Table 4.2) with different centroids of clusters (assuming user *id:2* and *id:4* are randomly selected as centroids), we get the following:
   $d_{age}(userid{:}1, userid{:}2) = 1/4 = 0.25$ (as they differ by one *age* range)
   $d_{age}(userid{:}1, userid{:}4) = 2/4 = 0.50$ (as they differ by two *age* range)
   Based on the *age* distance alone, user *id:1* would be assigned to user *id:2* (assuming it's a cluster centroid) as opposed to user *id:4*, since it has the least distance with the former.
   Now if we also consider the *qualification* attribute, for distance

calculation, we get:

$d_{qua}(userid{:}1, userid{:}2) = 3/4 = 0.75$ (as they differ by one *qua* range)

$d_{qua}(userid{:}1, userid{:}4) = 1/4 = 0.25$ (as they differ by two *qua* range)

Total distance T_d based on [*age*, *qualification*] attributes for user *id:1* with respect to user *id:2* and *id:4* are:

$T_d(userid:1, userid:2) = 0.25 + 0.75 = 1.00$

$T_d(userid:1, userid:4) = 0.50 + 0.25 = 0.75$

User *id:1* would be assigned to user *id:4* (assuming it's a cluster centroid) as it has the least total distance as shown above.

ii. The *location* attribute has three categorical values: *same city*, *same country* and *same timezone*. If user *id:1* and user *id:2* (centroid of a cluster) are in the same city, the distance is 0, otherwise it's 1 if they've the same country, else it's 2 if they've the same timezone, lastly the distance is 3 if non of the cases above is satisfied. We then normalize the calculated distance by dividing it by the max. possible distance which is 3.

On modelling the distance d_{loc} for *location* attribute for user *id:1* (see Table 4.2) with different centroids of clusters (assuming user *id:2* and *id:4* are randomly selected as centroids), we get the following:

$d_{loc}(userid{:}1, userid{:}2) = 1/3 = 0.33$ (same city)

$d_{loc}(userid{:}1, userid{:}4) = 3/3 = 1.00$ (none of the 3 conditions match)

Therefore, by the distance of the *location* attribute alone, user *id;1* should belong to the cluster corresponding to user *id:2* as it has the least distance with it.

iii. The *gender* attribute is mapped to a binary distance metric. For instance, if the *gender* of two users are same, then the distance d_{gen} is 0, otherwise 1.

On modelling the distance d_{gen} for *gender* attribute for user *id:1* (see Table 4.2) with different centroids of clusters (assuming user *id:2* and *id:4* are randomly selected as centroids), we get the following:

$d_{gen}(userid{:}1, userid{:}2) = 0$ (same gender)

$d_{gen}(userid{:}1, userid{:}4) = 1$ (different gender)

Therefore, by the distance of the *gender* attribute alone, user *id:1* should belong to the cluster corresponding to user *d:2* as it has the least distance with it.

iv. For *interest* attribute, we use a similarity measure based on the hierarchy tree, like the Rada measure [52]. This measure was founded on the basis that we can calculate the similarity based on the hierarchical links such as 'IS-A'. To calculate the similarity of two concepts in a hierarchy tree, we must calculate the number of the minimal arcs which separate them. This measure, based on the edge counting between nodes by the shortest way, presents a method to evaluate the semantic similarity in a hierarchical structure tree. The hierarchy we used for *interests* of users is based on WordNet [39] and the similarity measure used is based on the Wu and Palmer method [66]. It calculates relatedness by considering the depths of the two synsets in the WordNet taxonomies, along with the depth of the LCS (Least Common Subsumer). The formula is given as:

$$score = 2 * depth(lcs)/(depth(s1) + depth(s2)) \qquad (4.2)$$

.

This means that $0 < score <= 1$. The score can never be zero because the depth of the LCS is never zero (the depth of the root of a taxonomy is one). The score is one if the two input synsets are the same. Since we are implementing our system in a distance measure (and not similarity) the final value of distance between the interests is $[1 - score]$.

v. The *grade* attribute distance measure between a data vector and centroid is simply the difference between their grade values. On modelling the distance d_{grade} for *grade* attribute for user *id:1* (see Table 4.2) with different centroids of clusters (assuming user *id:2* and *id:4* are randomly selected as centroids), we get the following:

$d_{grade}(userid{:}1, userid{:}2) = |0.4 - 0.1| = 0.3$

$d_{grade}(userid{:}1, userid{:}4) = |0.1 - 1.0| = 0.9$

Therefore, by the distance of the *grade* attribute alone, user *id:1* should belong to the cluster corresponding to user *id:2* as it has the least distance with it.

(b) In traditional k-means algorithm, the centroids are typically recalculated by taking the average sum of all the data vectors present within a cluster, as represented by the equation below:

$$m_j = 1/n_j \sum_{\forall z_p \epsilon C_j} z_p \tag{4.3}$$

until a stopping criteria is reached.

However in our case, centroids are recalculated in a different way based on each attribute value of every data vector within a particular cluster, as shown below:

i. For [*age*, *grade*] attributes, the centroid value corresponding to these attributes is simply the mean of *age* and *grade* of every data vector present within the cluster. This is very similar to how centroids in traditional k-means are recalculated (see equation 4.3)

ii. In case of [*location*, *gender*, *qualification*, *interests*] attributes, the centroid values corresponding to these attributes would be the value which is present most number of times i.e. the most common, within the data vectors belonging to a particular cluster.

end repeat when a stopping criteria is reached.

K-means clustering process ends when any one of the following stopping criteria is reached: when the maximum number of iterations has been exceeded or when there is little to no change in the centroid vectors over multiple iterations.

We use k-means for two different purpose: 1) To formulate a baseline cluster model, which we later use to compare the group formation quality against the actual model generated using our algorithm and 2) To initialize one of the particles in our hybrid particle swarm optimization method, described in the next sub-subsection.

Formulating a baseline model: We use two different baseline models for result comparison.

- Number of clusters/groups (k) is specified: In this case, the number of clusters to be formed using k-means is specified by the user. Each cluster obtained after running k-means will have data vectors which are very similar to each other. However, in order to have *intra cluster heterogeneity* we need to have diverse data vectors within a cluster. To build an unbiased baseline model, we create equal number (k) of empty clusters. Then using the first cluster obtained via k-means, we evenly distribute the data vectors in them to each of these empty clusters. We repeat this process using the data vectors from all other clusters obtained using k-means. In the end, we have a new set of k clusters with data vectors which are diverse and can be used as a good baseline for result comparison.

- Number of users (α) in a cluster/group is specified: In this case, the number of users in each cluster or group is pre-decided. In order to account for *intra cluster heterogeneity*, we run the same process which we did in the formulation of previous baseline model. To build an unbiased baseline model, we create an empty cluster of size α. Next, we use a data vector from each of the clusters obtained using k-means to fill-up the empty cluster until the α value is reached. Then we create another

empty cluster and repeat the same process until this new cluster is again filled up with α data vectors. This process is repeated until all the data vectors have been exhausted in creation of new clusters of size α. In the end, we have a new set of clusters (size α) with data vectors which are diverse and can be used as a good baseline for result comparison.

In the next subsection, we'll see details about particle swarm optimization algorithm. We modify the algorithm for MOOCs and combine it with k-means to build a hybrid algorithm for group formation.

4.3.2 Particle Swarm Optimization

Particle swarm optimization (PSO) [28] is a population based stochastic optimization technique inspired by social behavior of bird flocking or fish schooling [29]. PSO shares many similarities with evolutionary computation techniques such as Genetic Algorithms (GA) [11]. PSO maintains a set of particles with some population, each particle represents a potential solution to an optimization problem.

As mentioned before, PSO can simulate the behavior of bird flocking [21]. Let's assume the following scenario: a group of birds are randomly searching food in an area. There is only one piece of food in the area being searched. All the birds do not know the exact location of food. But they know how far the food is in each iteration. So what's the best strategy to find the food? The effective one is to follow the bird which is nearest to the food, it is observed that the bird that chirps the loudest is typically the one which is closest to the food. PSO is based on this principle wherein each single solution is a 'bird' in the search space, called as "particle". All the particles have certain level of fitness which is evaluated by the fitness function to be optimized, and have velocities which direct the flying of the particles. The particles fly through the problem space by following the current optimum particles.

In summary, in the context of PSO, a swarm refers to a number of potential solutions to an optimization problem, wherein each solution is referred to as a particle. The aim of PSO is to find an optimum solution based on a

certain *fitness* function. Every particle is evaluated with respect to this *fitness* function, the fittest particle is accepted as solution.

The algorithm keeps track of three global variables:

- Target value or condition
- Global best (**gBest**) value indicating the particle which is closest to the target
- Stopping value indicating when the algorithm should stop if the target is not found

Also, each particle i in PSO maintains the following information:

- $\mathbf{x}_i$: the current position of the particle
- $\mathbf{v}_i$: the current velocity of the particle
- $\mathbf{pBest}_i$: the personal best position of the particle.

Using the above notations, a particle's position is adjusted according to the following equations:

$$v_{i,k}(t+1) = wv_{i,k}(t) + c_1 r_{1,k}(t)(pBest_{i,k}(t) - x_{i,k}(t)) + c_2 r_{2,k}(t)(gBest_{i,k}(t) - x_{i,k}(t)) \quad (4.4)$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (4.5)$$

where w is the inertia weight, c_1 and c_2 are the acceleration constants, $r_{1,k}(t)$, $r_{2,k}(t)$ are random numbers between (0,1), and $k = 1, ..., N_d$.

From equation 4.4, we can see that the velocity of the particle is dependent of 3 different components: 1) fraction of particles' previous velocity, 2) cognitive component, which is the separation of the particle in terms of the distance of its current position from its personal best position, 3) social component, which is the separation of particle from the global best particle. This

is best shown in Figure 4.3. The particle x's velocity is updated to V_{t+1}, based on the current velocity V_t, personal best *pBest* and global best *gBest*.

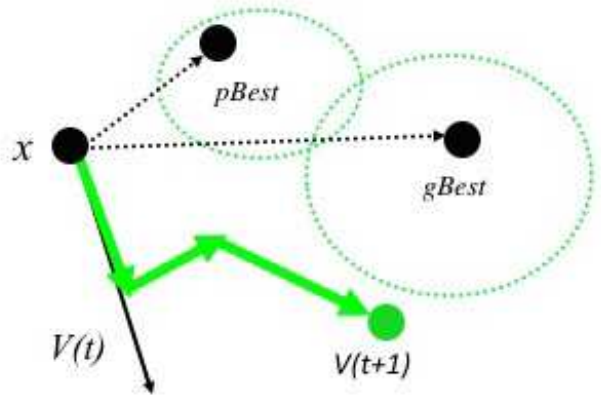


Figure 4.3: Particle Velocity Update

PSO is repeatedly executed until a stopping criteria has been reached, which could either be the maximum number of iterations exceeded or change in velocity over multiple iterations being negligible. A pseudocode for PSO algorithm can be seen in Figure 4.4.

```
Input: ProblemSize, Population_size
Output: P_g_best
Population ← ∅
P_g_best ← ∅
For (i = 1 To Population_size)
  P_velocity ← RandomVelocity()
  P_position ← RandomPosition(Population_size)
  P_p_best ← P_position
  If (Cost(P_p_best) ≤ Cost(P_g_best))
    P_g_best ← P_p_best
  End
End
While (¬StopCondition())
  For (P ∈ Population)
    P_velocity ← UpdateVelocity(P_velocity, P_g_best, P_p_best)
    P_position ← UpdatePosition(P_position, P_velocity)
    If (Cost(P_position) ≤ Cost(P_p_best))
      P_p_best ← P_position
      If (Cost(P_p_best) ≤ Cost(P_g_best))
        P_g_best ← P_p_best
      End
    End
  End
End
Return (P_g_best)
```

Figure 4.4: Pseudocode for particle swarm optimization

4.3.3 Hybrid Particle Swarm Optimization

Based on group formation as a facilitator of any learning process, it was found that proper training of a group allows better interaction between members and these interactions are important for learning [43], which means that the potential for interactions in a group is determined by the process through which groups are formed [65].

Typically, there are three different ways of forming a collaborative group: 1) self-organized: where students have the freedom to form groups by teaming

up together, 2) organized by instructor: groups chosen by teacher of the course 3) groups formed based on some algorithms. The first strategy allows having groups with high empathy among the members, but may entail two undesirable consequences: it usually prevents students from working with classmates who do not belong to their regular study groups; and it may produce groups formed not by pedagogical criteria, but by other criteria of a different nature (friendship, for example). The second strategy allows for mixing all students together with the hope of reaching heterogeneity inside the groups. This hope however is not always fulfilled because such a strategy may obtain groups where all members exhibit desirable or undesirable characteristics. Lastly, when using a defined algorithm based on a certain grouping logic, it provides for a more balanced grouping of students which includes a mix of strong and weak students over different competency levels like grade, gender, location and qualification.

Therefore, using computational techniques to support the formation of groups in education system becomes inherently important to improve the overall learning process amongst students. Many researchers in the field of computing science have proposed mathematical models and algorithms which has facilitated the work of group formation in classrooms under different perspectives among which the cognitive perspective, roles and interests are used mostly [36] [15] [31]. Particle swarm optimization (PSO) has been used to solve various problems of the level of complexity NP-Hard [9] [24] [68]. The results of these studies show that PSO has been very effective in solving problems of this level of complexity. Given our problem, which involves similar complexity in terms of optimization of different students attributes to form effective groups, we concluded that it would be a good approach to use PSO.

In the context of grouping, a single particle in PSO represents the N_c group centroid vectors, wherein each particle $\mathbf{x}_i$ is constructed as follows:

$$\mathbf{x}_i = (\mathbf{m}_{i1}, \ldots \mathbf{m}_{ij} \ldots, \mathbf{m}_{iNc}) \tag{4.6}$$

where $\mathbf{m}_{ij}$ refers to the j-th group centroid vector of the i-th particle in the group C_{ij}. Therefore, a swarm represents a number of candidate solutions

as each particle in itself is a solution.

We use the modified k-means to initialize the N_c centroid vectors of one of the particles of the swarm. The centroid vectors of remaining particles are initialized randomly using the data vectors. As mentioned before, the goal is to have heterogeneous groups (based on some attributes) which are similar to each other on 'grading' levels. Within each of these groups, we want students having similar 'interests' and 'location' but with diverse 'gender, 'age', 'qualification' and 'grade'. Moreover, the average 'grade' across all groups should be similar, to prevent undesirables grouping of low grade or high grade students together. This is the grouping logic we have used in our experiment. However, it is upto the instructor to decide on the way these attribute contribute in group formation, our algorithm is flexible enough to accommodate for any grouping changes based on the way these attributes are used.

Using the above mentioned grouping logic, we build N_c groups for each particle. For each group in a particle, we have data vectors which are similar in terms of *location* and *interests*, but diverse in *age*, *gender*, *qualification* and *grade*. Each data vector is allocated to the group where the distance between each attribute of the centroid vector with the corresponding attribute of the data vector is minimal for 'interests' and 'location'. For the remaining attributes, we want them to be diverse to account for heterogeneous groups.

Once the groups of all particles are initialized, we calculate the *fitness* of each particle, which is measured using the following fitness error functions:

$$fitness(P_{i_{grade}}) = \forall c_{ij} \epsilon N_c : |max_{(}c_{ij_{grade}}) - min_{(}c_{ij_{grade}})| \tag{4.7}$$

$$fitness(P_{i_{loc,int}}) = \frac{\sum_{j=1}^{N_c} [\sum \forall z_p \epsilon C_{ij}, d(z_p.m_j)/|C_{ij}|]}{N_c} \tag{4.8}$$

where d is Euclidean distance defined in equation 4.1, $|C_{ij}|$ is the number of data vectors belonging to group C_{ij}.

Above mentioned equations 4.7 and 4.8 are *fitness* measures of a particle in terms of 'grade' and ['location', 'interest'] attributes respectively. The less the fitness error, the better the quality of groups formed. In our case, we say

that a particle is *fit* if, 1) the groups within the particle are *inter-homogeneous* for 'grade' attribute and, 2) the Euclidean distance is minimum for 'interest' and 'location' within each group of a particle.

More specifically, if the *grade* difference between the *max* and *min* grade value for all groups within a particle is less than a threshold t (t=0.1), the particle is *fit*. For instance, if each group belonging to a particle has similar grade levels, then the *difference* between *max* and *min* grade values will be negligible, hence we can say that the grouping is ideal since it means that no particular group has an advantage over the other in terms of smart students being grouped together and vice versa. In case there is disparity in grade levels, we can try to minimize this disparity in the next iteration. Similarly, we try to minimize the distances between data vectors and group centroids for '*location* and *interest* attributes. We select the *gBest* (global best) and the *pBest* (personal best) particles based on the combination of fitness acheived using equations 4.7 and 4.8.

The particle with least 'grade' difference and minimum 'location' and 'interest' distances, is selected as the *global best*. Also, each particle stores its *local best* state, which has the least grade difference and minimum 'location' and 'interest' distances, in any given iteration. Next, we update the group centroids of each particle using equations 4.4 and 4.5. However, the update for each attribute of the centroids is different from each other. In case of *age* and *grade* attributes, the updated values depend on the *age, grade* values of *global best* and *personal best* particle whereas for all other attributes [location, gender, qualification and interests], the updated values depend on the most common values of all data points in respective groups. This entire sequence completes one iteration of the algorithm. PSO is usually executed until a specified number of iterations has been exceeded or if a certain level of fitness has been achieved.

Below is the summary of group formation using hybrid particle swarm optimization (PSO):

1. Initialize each particle with N_c randomly selected group centroids, except

one particle whose centroids are initalized using modified k-means.

2. for *iteration* $t = 1$ to t_{max}

   (a) for each particle i do

   (b) for each data vector $\mathbf{z}_p$

      i. calculate the attribute distances d($\mathbf{z}_p$,$\mathbf{m}_{ij}$), between the data vector $\mathbf{z}_p$ with each group centroid $\mathbf{m}_{ij}$, for all group centroids C_{ij}.

      ii. assign $\mathbf{z}_p$ to the Cluster C_{ij} where the distance is minimum.

      iii. calculate the fitness of particle using equations 4.10 and 4.7.

   (c) update the *global best* particle in the swarm along with the *personal best* of each particle.

   (d) update the group centroids of each particle using equations 4.4 and 4.5.

4.4 Experiments

4.4.1 Dataset

Similar to reciprocal recommendation, our dataset is based on real data from MITx-Harvardx dataset [41]. We also synthesized an additional data column like user *interests* to cover our cases as we believe that it can be potentially important attribute for group formation. We conducted experiments with sampled dataset of 100, 1000 and 5000 user records.

4.4.2 Evaluation Metrics

In this section, we define different metrics to compare results of the baseline models generated using the modified k-means and hybrid particle swarm optimization. The main purpose is to compare the quality of groups generated using the modified k-means and hybrid PSO based on the following three criteria:

- the calculated fitness error as defined in equations 4.10 and 4.7.

- intra-group distances, i.e. the measure of diversity inside each group, the objective is to maximize intra-group distances for *grade* attribute so that we've students with different grades grouped together. It is calculated by measuring the distance between data vectors within a group for the mentioned attribute.

- inter-group distances, i.e. the measure of similarity between different groups, the objective is to minimize inter-group distances for *grade* attribute while at the same time maximizing the inter-group distances for 'interests' and 'location' attributes.

The objective of these metrics is to help us to measure diversity inside each of the group while at the same time making sure that every group is similar to the other based on the grading levels.

4.4.3 Results

Our experiments employed a series of testing to analyze the effectiveness of the PSO algorithm for groups formation in MOOCs. The data used for experiments was partially synthesized from the original dataset released by MITx-Harvardx [41]. The algorithm was run on a computer with 2.7 GHz Intel Core i5 processor and 8 GB RAM. In order to examine the effectiveness of the PSO, five different sets of data for each [100, 1000, 5000] samples were generated randomly from the original dataset which has around 300k records.

The parameters used for velocity update (refer equation 4.4) are: $w = 0.72$ and $c_1 = c_2 = 1.49$. These values were chosen to ensure good convergence [62]. Also, the number of particles predefined is [10, 20, 50] respectively for data with volumes of [100, 1000, 5000] records. This was chosen based on the study [6] that any number of particles between 10 to 100 are capable of producing results that are clearly superior or inferior to any other value for a majority of the tested problems. The results reported is averaged over 5 different simulations, each simulation was run with different data samples. Our results will be analyzed on two different baseline models: 1) Number of

groups/groups (k) is specified and, 2) Number of users (α) in a group/group is specified.

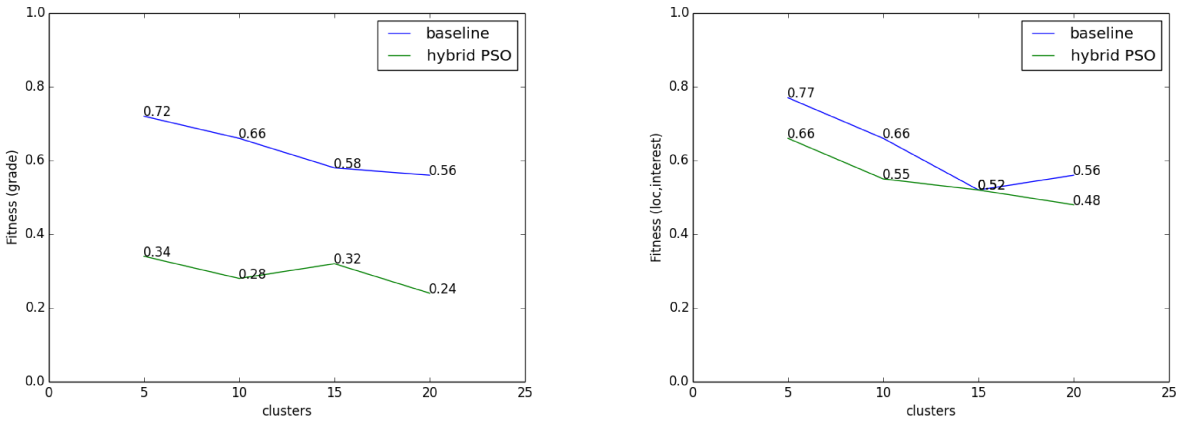
Fitness Analysis

Figure 4.5 shows the effect of varying the number of groups on the fitness values for 'grade difference', 'interest' and 'location' distances for 100 data records. As expected, the fitness error should go down as the number of groups increase. We calculate the grade fitness based on equation 4.10, wherein the difference between the maximum and minimum grade values is taken from all the groups. This difference is represented as the *fitness score* in figure 4.5 (a). It is seen that the fitness score decreases with increase in number of groups which means that the quality of groups formed increase as the number of groups increase. However, this observation is to be taken with a grain of salt since it is not always true, as is seen in case where the fitness score increases from 0.28 to 0.32 even when the group size increases from 10 to 15. Moreover, the hybrid PSO outperforms the baseline model in terms of fitness achieved and thereby the quality of groups formed.

Next, we calculate the 'location' and 'interests' fitness based on equation 4.7, wherein the distance of each data vector to its corresponding centroids is calculated for these attributes. The less the distance, the better the *fitness* or quality of the groups formed. If the 'location' of the data vector is same (same city) as the location of its corresponding centroid, then the distance is 0; if location is in the same country (different cities), then the distance is 1; if the location is in same timezone, the distance is 2; else the distance is 3. We then normalize the total distance by dividing it by the max. possible distance which is 3. This is repeated to calculate location distance for all data vectors with their corresponding centroids.

Similarly, 'interests' distance is also calculated for every data vector. The total distance for 'location' and 'interest' is then normalized to produce a fitness score which is shown in Figure 4.5 (b). It is seen that the fitness score decreases with increase in number of groups which means that the quality of groups formed increase as the number of groups increase. Again, this observa-

tion is not always to be taken as true. The hybrid PSO model performs better than the baseline model, which means that the quality of groups formed is better in terms of location and interests fitness score.



(a) Grade Fitness

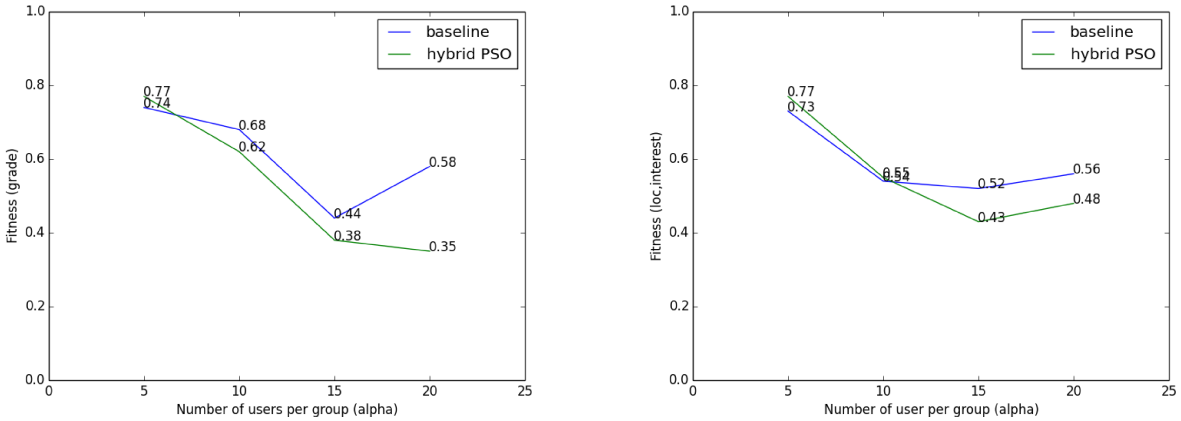
(b) Fitness for Location, Interests

Figure 4.5: Effect of different number of groups on Fitness

We also compare the fitness results when the number of users in a group (α) or group is predetermined, the results are shown in Figure 4.6. Looking at the grade fitness graph (Figure 4.6 (a)), the fitness error decreases with increase in the number of users per group. This is expected because with more users the chances of 'grade' scores being skewed decreases, hence the grade fitness increases. However, the grade, location and interests fitness for the hybrid and baseline model is close for low values of (α). This can be attributed to the fact that with lower number of learners in a group, the chances of similar values for the mentioned attributes within a group decreases. The hybrid PSO model performs better than the baseline when the number of users are more. However, the performance is almost similar for less number of users per group.

Similarly, for 'location' and 'interest' fitness (Figure 4.6(b)), the hybrid PSO models performs the same as the baseline model when the number of users per group are less. However, it outperforms the baseline when the number of users per group increase. However, the overall fitness error may increase even with the increase in number of users. It can be seen that the fitness score increases from 0.43 to 0.48 when the number of users per group increase

from 20 to 25. Overall, the hybrid PSO performs better than both the baseline models, which tells us that the algorithm is useful in generating quality groups.



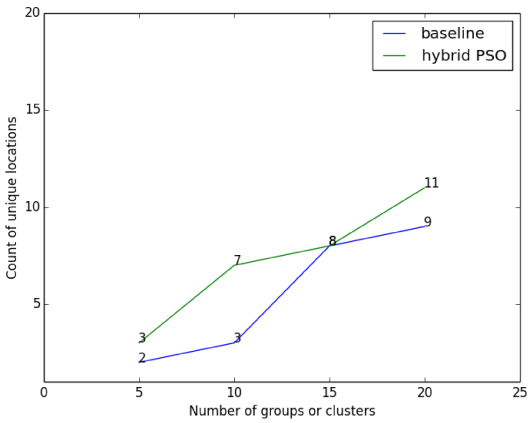
(a) Grade Fitness

(b) Fitness for Location, Interests

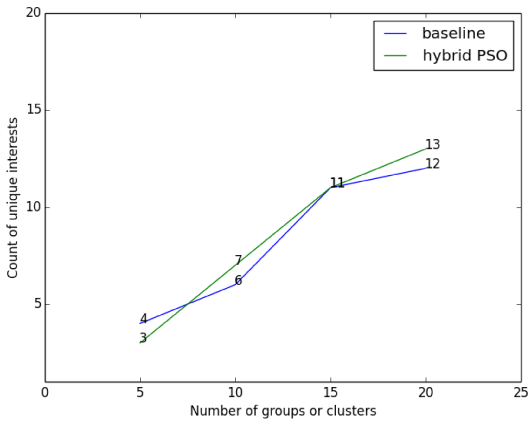
Figure 4.6: Effect of number user per group (α) on Fitness

Inter-Group Distance Analysis

Here we try to compare the distances between the centroids of groups or groups. The objective is to maximize the distances between groups for 'location' and 'interest' attributes. We wanted to group students who have similar interest and location, which means that ideally different groups should have students with different interests and location. This was analyzed by comparing the *interests* and *location* attributes for centroids of various groups. We counted the unique location values from the total location values of centroids of various groups formed. The more the count of unique locations, the more the distance between centroids, thereby increasing the inter-group distance between centroids for location attribute. We apply the same logic for 'interests' attribute as well. The result using 100 data records is shown in Figure 4.7. From the graphs, it is evident that inter-group distances for 'location' is more for the hybrid PSO model as the unique count is more. However, the inter-group distances for 'interest' is only slightly better for hybrid PSO at higher number of groups.



(a) Unique Location Count



(b) Unique Interest Count

Figure 4.7: Effect of number user per group (α) on Fitness

Intra-Group Distance Analysis

We also calculate the Intra-group distances to measure the 'grade' distribution inside a group. Ideally, we would want to have a good mix of high and low grade students (heterogeneous) for a positive learning outcome. We use a 'heterogeneity factor' (HF) as a measure to calculate the heterogeneity of group with respect to the 'grade' attribute [15]. The measure of HF can be computed as follows. Let AG_i be the average grade of the maximum and the minimum student-grade in the i-th group.

$$AG_i = \frac{max_{(C_{grade_i})} + min_{(C_{grade_i})}}{2} \tag{4.9}$$

where, s_j is the grade of j-th student belonging to group C_i.

The measure of 'heterogeneity factor' is then defined as:

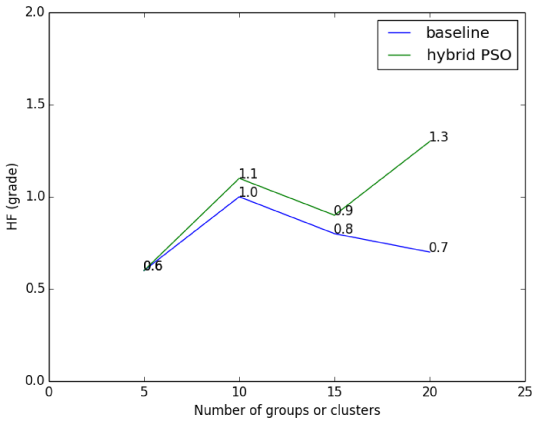
$$HF_i = \frac{max_{(C_{grade_i})} + min_{(C_{grade_i})}}{1 + \sum_j |AG_i - grade(s_{j(i)})|} \tag{4.10}$$

where,

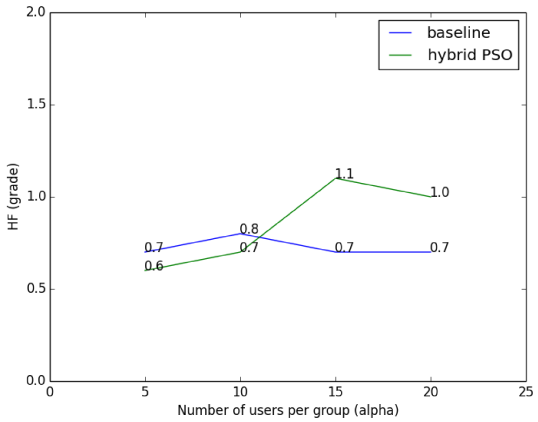
$s_{j(i)}$ is the grade of j-th student belonging to group C_i.

For any group with high 'heterogeneity', the HF_i should be higher than 1. It is trivial to show that $HF_i = 0$ when all students in a group have equal student-grades; $HF_i < 1$ when there is less heterogeneity in the group,

meaning student-scores are at two extremes. The greater the HF_i, the better the heterogeneity. Figure 4.8 below shows the HF_i for different groups or groups generated via baseline and hybrid PSO models using 100 data records. As can be seen in Figure 4.8(a), PSO hybrid outperforms baseline for grade heterogeneity, especially at the higher number of groups. Similarly, in the graph 4.8(b), the hybrid PSO performs better than the baseline model at higher number of users (α) per group.



(a) HF per the number of groups



(b) HF per the number of users per group (α)

Figure 4.8: Heterogeneity Factor comparison with different baseline models

Measuring Scalability

When it comes of group formation, not only the quality of groups matter but also it is important that the hybrid PSO algorithm scales well. In the given time, when the number of students registering for MOOCs is rising every year, it becomes absolutely necessary that the algorithm is robust enough to accommodate the rise in number of students and generate dynamic learning groups.

We ran five simulations on different dataset for each of the [100,1000,5000] data samples. The average time to generate learning groups with different data records is shown in Fig 4.9. The execution time in 'minutes' for generating [5, 10, 15 and 20] student groups using hybrid PSO are [66.4, 144.2, 173.5 and 252.5] respectively. Same as in reciprocal recommendation, the execution

time to generate groups will depend on several factors like dimension of data, number of data records, sparsity of data etc. Given the change in any of the mentioned factors, the execution time will change accordingly.

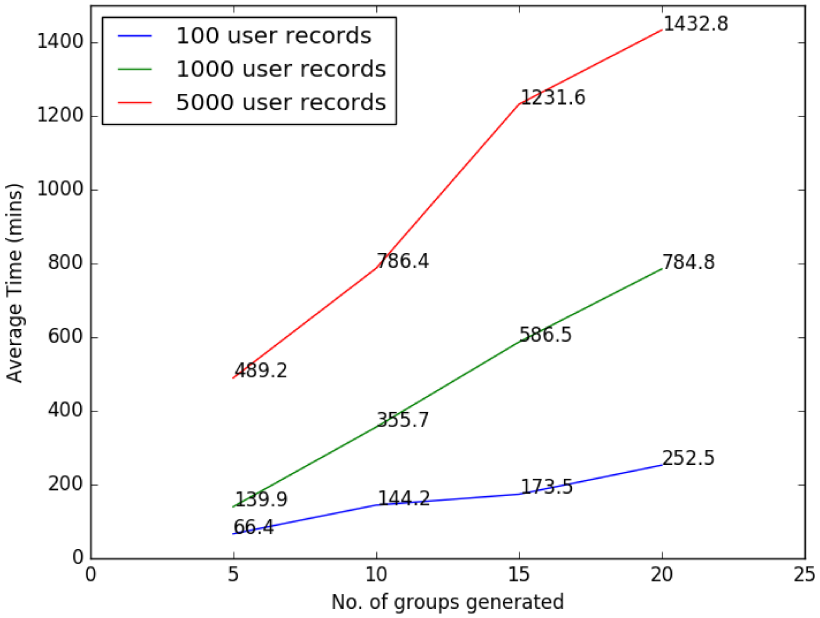


Figure 4.9: Average execution time for generating groups using 'selection'

The above results show that the hybrid PSO models performs better than both the baseline models in most use cases, thereby generating better quality of groups. Although, the algorithm could not be tested real-time on an actual MOOC platform, these results nevertheless provide insights about the group formation technique using the user attributes. Hence, it would be worthwhile to integrate it within an actual MOOC to get a realistic opinion on its performance.

Chapter 5

Summary and Conclusions

The first part of our research involved building a novel reciprocal recommendation system for MOOCs in which learners are recommended to each other based on their preferences and profile attributes which satisfy those preferences. The evaluation of the proposed algorithm was done using the modified definitions of precision, recall and discounted cumulative gain (DCG). Several runs were made to compare the precision, recall and DCG for top-N recommendations with/without the 'selection' phase of the algorithm. Also, a comparison of the execution time was measured for large dataset samples.

The results show that the our system performs better than the baseline system on the measures of precision, recall and discounted cumulative gain. Moreover, these tests have also shown that the system is robust enough to take into account the large datasets for top-N recommendations. Reciprocal recommendation provides a quick and easy way for the users to connect with their preferred learners in a large pool of users. It has a lot of benefits in terms of peer learning and improving user engagement which could sustain the interests of learners and mitigate the dropouts rates in MOOCs. The proposed algorithm aims at allowing learners to reach out and communicate with other similar learners, it thereby makes a promising case for our system to be deployed within an actual MOOC.

As future work, we plan to incorporate some more user attributes like 'communication frequency' of users, their 'leadership ability' etc., based on their historical interaction on various MOOC forums. It would possibly help

in improving the list of recommendations. Moreover, we plan to conduct tests on an actual MOOC platform to measure the quality of recommendations.

In the second part of our work, we presented a framework using the hybrid particle swarm optimization to form student groups based on certain pre-specified attributes like [age, gender, location, qualification, interests and grade]. The evaluation of the proposed algorithm was done using two different baseline models to determine the overall quality of groups formed in terms of *fitness*, *intra-cluster heterogeneity* and *inter-cluster homogeneity* measures.

The results showed that the group quality was better when compared to the baseline models of groups formed using the modified k-means method. Also, the hybrid group formation algorithm was able to generate groups for relatively large datasets with close to 5000 records. Therefore, the tests show that the algorithm is robust and scalable taking into account the different variations of datasets. The proposed algorithm can help the instructors to automatically generate suitable groups of students for online classes which may cause the participating students to be more involved in the knowledge construction process by increasing the level of interaction and thus improve learning.

As a future work we plan to conduct tests on an actual MOOC platform to get a real-time assessment of the quality of student groups formed based on the proposed algorithm. The algorithm can also be improved to add more attributes like 'communication frequency' of users, their 'leadership quality', 'interactivity within forums' etc., which could potentially increase the chances of forming better quality groups. These attributes could be derived based on the past courses that the students had registered for, or in some form of a feedback from students themselves based on a certain questionnaire.

Our algorithm would also work on students portal smaller than MOOCs. However, there would be some challenges in terms of data privacy, ethical concerns etc., in order to integrate our system with any given MOOC. Our algorithm is flexible enough to include any changes with respect to user attributes such that we can prioritize the importance of one attribute over the

other, if required. Case studies reveal that the number of participating users in MOOCs is increasing every year, hence it becomes quite challenging to establish the same kind of communication that exists within a classroom. However, with these proposed techniques, we believe we can bridge that gap to some extent.

Bibliography

[1] Joshua Akehurst, Irena Koprinska, Kalina Yacef, Luiz Augusto Sangoi Pizzato, Judy Kay, and Tomasz Rej. Ccr-a content-collaborative reciprocal recommender for online dating. In *IJCAI*, pages 2199–2204, 2011.

[2] Ammar Alanazi and Michael Bain. A people-to-people content-based reciprocal recommender using hidden markov models. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 303–306. ACM, 2013.

[3] Zhamri Che Ani, Azman Yasin, Mohd Zabidin Husin, and Zauridah Abdul Hamid. A method for group formation using genetic algorithm. *International Journal on Computer Science and Engineering*, 2(09):3060–3064, 2010.

[4] Rel Guzman Apaza, Elizabeth Vera Cervantes, Laura Cruz Quispe, and José Ochoa Luna. Online courses recommendation based on lda. In *SIMBig*, pages 42–48. Citeseer, 2014.

[5] Fatiha Bousbahi and Henda Chorfi. Mooc-rec: A case based recommender system for moocs. *Procedia-Social and Behavioral Sciences*, 195:1813–1822, 2015.

[6] Daniel Bratton and James Kennedy. Defining a standard for particle swarm optimization. In *2007 IEEE swarm intelligence symposium*, pages 120–127. IEEE, 2007.

[7] Lori Breslow, David E Pritchard, Jennifer DeBoer, Glenda S Stump, Andrew D Ho, and Daniel T Seaton. Studying learning in the worldwide classroom: Research into edx's first mooc. *Research & Practice in Assessment*, 8, 2013.

[8] Jane Brindley, Lisa Marie Blaschke, and Christine Walti. Creating effective collaborative learning groups in an online environment. *The International Review of Research in Open and Distributed Learning*, 10(3), 2009.

[9] Wei-Neng Chen, Jun Zhang, Henry SH Chung, Wen-Liang Zhong, Wei-Gang Wu, and Yu-Hui Shi. A novel set-based particle swarm optimization method for discrete optimization problems. *IEEE Transactions on Evolutionary Computation*, 14(2):278–300, 2010.

[10] Xiaohui Cui, Thomas E Potok, and Paul Palathingal. Document clustering using particle swarm optimization. In *Proceedings 2005 IEEE Swarm Intelligence Symposium, 2005. SIS 2005.*, pages 185–191. IEEE, 2005.

[11] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[12] Hendrik Drachsler, Hans GK Hummel, and Rob Koper. Personal recommender systems for learners in lifelong learning networks: the requirements, techniques and model. *International Journal of Learning Technology*, 3(4):404–423, 2008.

[13] Hendrik Drachsler, Hans GK Hummel, and Rob Koper. Identifying the goal, user model and conditions of recommender systems for formal and informal learning. *Journal of Digital Information*, 10(2), 2009.

[14] Salvador Garcia-Martinez and Abdelwahab Hamou-Lhadj. Educational recommender systems: A pedagogical-focused perspective. In *Multimedia Services in Intelligent Environments*, pages 113–124. Springer, 2013.

[15] Sabine Graf and Rahel Bekele. Forming heterogeneous groups for intelligent collaborative learning systems with ant colony optimization. In *International Conference on Intelligent Tutoring Systems*, pages 217–226. Springer, 2006.

[16] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[17] Rania Hassan, Babak Cohanim, Olivier De Weck, and Gerhard Venter. A comparison of particle swarm optimization and the genetic algorithm. In *Proceedings of the 1st AIAA multidisciplinary design optimization specialist conference*, pages 18–21, 2005.

[18] Andrew Dean Ho, Isaac Chuang, Justin Reich, Cody Austun Coleman, Jacob Whitehill, Curtis G Northcutt, Joseph Jay Williams, John D Hansen, Glenn Lopez, and Rebecca Petersen. Harvardx and mitx: Two years of open online courses fall 2012-summer 2014. *Available at SSRN 2586847*, 2015.

[19] Wenxing Hong, Siting Zheng, Huan Wang, and Jianchao Shi. A job recommender system based on user clustering. *Journal of Computers*, 8(8):1960–1967, 2013.

[20] H Ulrich Hoppe. The use of multiple student modeling to parameterize group learning. 1995.

[21] Xiaohui Hu. Pso tutorial. `http://www.swarmintelligence.org/tutorials.php`, 2006.

[22] Mark Huxham and Ray Land. Assigning students in group work projects. can we do better than random? *Innovations in Education and Teaching International*, 37(1):17–22, 2000.

[23] Akiko Inaba, Thepchai Supnithi, Mitsuru Ikeda, Riichiro Mizoguchi, and JunâĂŹichi Toyoda. How can we form effective collaborative learning groups? In *International Conference on Intelligent Tutoring Systems*, pages 282–291. Springer, 2000.

[24] Bassem Jarboui, N Damak, Patrick Siarry, and A Rebai. A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems. *Applied Mathematics and Computation*, 195(1):299–308, 2008.

[25] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002.

[26] David W Johnson, Roger T Johnson, and Mary Beth Stanne. Cooperative learning methods: A meta-analysis, 2000.

[27] Greg Kearsley. *Online education: Learning and teaching in cyberspace*, volume 91. Wadsworth Belmont, CA, 2000.

[28] James Kennedy. Particle swarm optimization. In *Encyclopedia of machine learning*, pages 760–766. Springer, 2011.

[29] James Kennedy, James F Kennedy, Russell C Eberhart, and Yuhui Shi. *Swarm intelligence*. Morgan Kaufmann, 2001.

[30] Irena Koprinska and Kalina Yacef. People-to-people reciprocal recommenders. In *Recommender Systems Handbook*, pages 545–567. Springer, 2015.

[31] Karel Kreijns, Paul A Kirschner, and Wim Jochems. Identifying the pitfalls for social interaction in computer-supported collaborative learning environments: a review of the research. *Computers in human behavior*, 19(3):335–353, 2003.

[32] Chinmay Kulkarni, Julia Cambre, Yasmine Kotturi, Michael S Bernstein, and Scott R Klemmer. Talkabout: Making distance matter with small groups in massive classes. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*, pages 1116–1128. ACM, 2015.

[33] Chinmay E Kulkarni, Michael S Bernstein, and Scott R Klemmer. Peerstudio: rapid peer feedback emphasizes revision and improves performance. In *Proceedings of the Second (2015) ACM Conference on Learning@ Scale*, pages 75–84. ACM, 2015.

[34] Lei Li and Tao Li. Meet: a generalized framework for reciprocal recommender systems. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 35–44. ACM, 2012.

[35] Lisa Linnenbrink-Garcia, Kevin J Pugh, Kristin LK Koskey, and Victoria C Stewart. Developing conceptual understanding of natural selection: The role of interest, efficacy, and basic prior knowledge. *The Journal of Experimental Education*, 80(1):45–68, 2012.

[36] Dascalu Maria-Iuliana, Bodea Constanta-Nicoleta, and Burlacu Alexandru. Platform for creating collaborative e-learning communities based on automated composition of learning groups. In *Engineering of Computer Based Systems (ECBS-EERC), 2013 3rd Eastern European Regional Conference on the*, pages 103–112. IEEE, 2013.

[37] Alexander McAuley, Bonnie Stewart, George Siemens, and Dave Cormier. The mooc model for digital practice. 2010.

[38] Prem Melville and Vikas Sindhwani. Recommender systems. In *Encyclopedia of machine learning*, pages 829–838. Springer, 2011.

[39] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[40] Tsunenori Mine, Tomoyuki Kakuta, and Akira Ono. Reciprocal recommendation for job matching with bidirectional feedback. In *Advanced Applied Informatics (IIAIAAI), 2013 IIAI International Conference on*, pages 39–44. IEEE, 2013.

[41] MITx and HarvardX. Harvardx-mitx person-course academic year 2013 de-identified dataset, version 2.0. 2014.

[42] On MOOC. Education letter.

[43] Julián Moreno, Demetrio A Ovalle, and Rosa M Vicari. A genetic algorithm approach for group formation in collaborative learning considering multiple student characteristics. *Computers & Education*, 58(1):560–569, 2012.

[44] Barbara Oakley, Richard M Felder, Rebecca Brent, and Imad Elhajj. Turning student groups into effective teams. *Journal of student centered learning*, 2(1):9–34, 2004.

[45] M Omran, Andries Petrus Engelbrecht, and A Salman. Particle swarm optimization method for image clustering. *International Journal of Pattern Recognition and Artificial Intelligence*, 19(03):297–321, 2005.

[46] Daniel FO Onah, Jane Sinclair, and Russell Boyatt. Dropout rates of massive open online courses: behavioural patterns. *EDULEARN14 Proceedings*, pages 5825–5834, 2014.

[47] Rena M Palloff and Keith Pratt. *Collaborating online: Learning together in community*, volume 32. John Wiley & Sons, 2010.

[48] Luiz Pizzato, Tomasz Rej, Joshua Akehurst, Irena Koprinska, Kalina Yacef, and Judy Kay. Recommending people to people: the nature of reciprocal recommenders with a case study in online dating. *User Modeling and User-Adapted Interaction*, 23(5):447–488, 2013.

[49] Luiz Pizzato, Tomek Rej, Thomas Chung, Irena Koprinska, and Judy Kay. Recon: a reciprocal recommender for online dating. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 207–214. ACM, 2010.

[50] Luiz Pizzato, Tomek Rej, Thomas Chung, Kalina Yacef, Irena Koprinska, and Judy Kay. Reciprocal recommenders. *ITWP 2010*, page 53, 2010.

[51] Luiz Augusto Pizzato and Cameron Silvestrini. Stochastic matching and collaborative filtering to recommend people to people. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 341–344. ACM, 2011.

[52] Roy Rada, Hafedh Mili, Ellen Bicknell, and Maria Blettner. Development and application of a metric on semantic nets. *Systems, Man and Cybernetics, IEEE Transactions on*, 19(1):17–30, 1989.

[53] Andrew I Schein, Alexandrin Popescul, Lyle H Ungar, and David M Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 253–260. ACM, 2002.

[54] Dhawal Shah. By the numbers: Moocs in 2015. `https://www.class-central.com/report/moocs-2015-stats/`, 2015.

[55] Guy Shani and Asela Gunawardana. Evaluating recommendation systems. In *Recommender systems handbook*, pages 257–297. Springer, 2011.

[56] S Shaw. New reality: Workplace collaboration is crucial. *Eedo Knowledgeware Whitepaper*, 2006.

[57] Robert E Slavin. Developmental and motivational perspectives on cooperative learning: A reconciliation. *Child development*, pages 1161–1167, 1987.

[58] Leonard Springer, Mary Elizabeth Stanne, and Samuel S Donovan. Effects of small-group learning on undergraduates in science, mathematics, engineering, and technology: A meta-analysis. *Review of educational research*, 69(1):21–51, 1999.

[59] Patrick T Terenzini, Alberto F Cabrera, Carol L Colbeck, John M Parente, and Stefani A Bjorklund. Collaborative learning vs. lecture/discussion: Students' reported learning gains. *Journal of Engineering Education*, 90(1):123, 2001.

[60] Bruce W Tuckman. Developmental sequence in small groups. *Psychological bulletin*, 63(6):384, 1965.

[61] Matheus RD Ullmann, Deller J Ferreira, Celso G Camilo, Samuel S Caetano, and Lucas de Assis. Formation of learning groups in cmoocs using particle swarm optimization. In *2015 IEEE Congress on Evolutionary Computation (CEC)*, pages 3296–3304. IEEE, 2015.

[62] Frans Van Den Bergh. *An analysis of particle swarm optimizers*. PhD thesis, University of Pretoria, 2006.

[63] DW Van der Merwe and Andries Petrus Engelbrecht. Data clustering using particle swarm optimization. In *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*, volume 1, pages 215–220. IEEE, 2003.

[64] César Vialardi, Javier Bravo Agapito, Leila Shila Shafti, and Alvaro Ortigosa. *Recommendation in higher education using data mining techniques*. Barnes, T., Desmarais, M., Romero, C., & Ventura, S., 2009.

[65] Noreen M Webb. Testing a theoretical model of student interaction and learning in small groups. *Interaction in cooperative groups: The theoretical anatomy of group learning*, 102:119, 1992.

[66] Zhibiao Wu and Martha Palmer. Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pages 133–138. Association for Computational Linguistics, 1994.

[67] Peng Xia, Benyuan Liu, Yizhou Sun, and Cindy Chen. Reciprocal recommendation system for online dating. In *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*, pages 234–241. ACM, 2015.

[68] Peng-Yeng Yin and Jing-Yu Wang. A particle swarm optimization approach to the nonlinear resource allocation problem. *Applied mathematics and computation*, 183(1):232–242, 2006.

[69] Hongtao Yu, Chaoran Liu, and Fuzhi Zhang. Reciprocal recommendation algorithm for the field of recruitment. *Journal of Information & Computational Science*, 8(16):4061–4068, 2011.

[70] Kang Zhao, Xi Wang, Mo Yu, and Bo Gao. User recommendations in reciprocal and bipartite social networks–an online dating case study. *Intelligent Systems, IEEE*, 29(2):27–35, 2014.