Competency, Compatibility and Preferences in Reciprocal Peer Recommendation

Boyd A. Potts
The University of Queensland
b.potts@uqconnect.edu.au

Hassan Khosravi
The University of Queensland
h.khosravi@uq.edu.au

Carl Reidsema
The University of Queensland
c.reidsema@uq.edu.au

ABSTRACT

Inviting learners to engage in peer learning and peer support has established benefits for both students and providers of education. Reciprocal recommender systems provide sophisticated filtering techniques that enable users to connect with one another. Recommender systems for technology enhanced learning have employed and tailored recommenders towards use in education, with a focus on recommending learning content rather than other users. In this paper, we discuss the role recommending reciprocal peers can play in educational settings and introduce our open-source course-level recommendation platform called RiPPLE and its capacity to provide reciprocal peer recommendation. The proposed algorithm is evaluated against key criteria such as scalability, reciprocality and coverage, showing improvement over a non-reciprocal recommender. Primary results indicate that the system can help learners connect with peers based on their knowledge gaps and reciprocal preferences, with designed flexibility to address key limitations of existing algorithms identified in the literature.

KEYWORDS

Reciprocal Recommender, Peer Learning, Peer Support, RecSysTEL

1 INTRODUCTION

Increased enrolments in higher education place a greater emphasis on designing more innovative and flexible learning options [2]. Creating blended learning environments not only creates opportunities for increased engagement but also new approaches to increase students' social/peer networks, both of which contribute to student success [11]. Providing students with the ability to leverage the inherent power of such large cohorts to improve their learning and university experience [12] can be achieved by adopting peer-learning software systems to support a community of inquiry pedagogy [4]. Supplementing the development of a learning community [5] with fit-for-purpose technology not only allows an increase in the frequency of interaction and communication between students, but creates an awareness of a community for them to draw upon or otherwise "belong" to [2].

Recommender systems [10] can contribute by providing sophisticated filtering techniques to help people find the resources that they need. Fundamentally recommenders entail some operationalised user preferences and seek solutions such as a list of objects that match those preferences to a higher degree than competing items. In reciprocal recommendation, items are usually other users whose preferences must also be fulfilled, requiring a higher level of complexity than other recommender systems [8]. Much of the associated research has been developed and evaluated in existing social networks, particularly online dating sites [1, 3]. We argue that the general framework of these reciprocal recommender systems can be adapted successfully to educational settings for social learning.

All systems incontrovertibly share the same fundamental goal to provide recommendations based on users' preferences in an otherwise overwhelming information environment where the likelihood of users successfully finding preferred items without technological assistance is very low. However the nature of domain-specific information and definition of a successful recommendation is so heavily context- and goal-dependent that little more than the general way of thinking can be adapted or generalised from existing systems to new domains. This is particularly true of the formulation of user preference models upon which recommendations are to be based, making them necessarily bespoke.

Building on previous research utilising peer learning and support for improving learning [6], the area remains fertile for many research and development opportunities. We introduce an open-source course-level platform called RiPPLE (Recommendation in Personalised Peer Learning Environments) that has the capacity of providing reciprocal peer recommendation, enabling learners to provide learning support, seek learning support, or find study partners, using competency-based preference models suitable for both on-campus and online courses.

In preparation for trialling RiPPLE in four large courses at a research-intensive university, an initial set of experiments were conducted. The evaluation concentrates on the feasibility of the recommender by evaluating the impacts of the size of the cohort, distributions of competencies, availabilities and willingness to collaborate. Synthetic data sets were created for this purpose. Primary results indicate that the system can help learners connect with peers based on their knowledge gaps and reciprocal preferences.

2 PROVIDING RECIPROCAL PEER RECOMMENDATION IN RIPPLE

This section introduces both the proposed platform and reciprocal recommender system. Section 2.1 provides an overview of the developed platform called RiPPLE. Section 2.2 presents a formal description of the problem under investigation using mathematical notation. Section 2.3 defines a compatibility function, which is used in the reciprocal peer recommendation algorithm introduced in Section 2.4.

2.1 Platform Description

RiPPLE is a course-level, student-facing platform that enables students to provide peer learning support, seek peer learning support, and find study partners.

Individuals nominate their availability between 08:00 and 20:00, Monday to Friday, and their preferences for providing or seeking peer learning support and finding study partners across the range of course-relevant topics. An indicator of the individual's competency is provided in the form of a coloured bar chart superimposed over the list of topics. Competency levels are derived initially from student's responses to multiple choice questions (MCQs) which are also contained in the RiPPLE platform. Competencies can be updated using the individual's cumulative performance on assessment items and quizzes progressively during the teaching period.

2.2 Problem Formulation

Let U denote the number of learners that are using the platform, and u, u_1 and u_2 refer to arbitrary learners. Let L denote the number of distinct topics that are covered in the course, and l to denote an arbitrary topic. Let T denote the number of weekly available time slots for scheduling a study session, and t to denote an arbitrary time slot. Finally, let Q present the number of different roles a learner can have during a study session and q present an arbitrary role; in the current setting, Q is set to three: q=1 is to provide peer learning support, q=2 is to seek peer learning support, and q=3 is to search for study partners.

Let's assume that the following information can be collected through the platform:

- **Requests,** $R_{U \times L \times Q}$: A three-dimensional array where R_{ulq} = 1 indicates that user u has indicated interest in participating in a study session on topic l with role q.
- **Competencies,** $C_{U \times L}$: A two-dimensional array in which C_{ul} shows the competency of user u in topic l. Values in C are in the range of 0 to 100.
- **Availability,** $A_{U \times T}$: A two-dimensional array in which A_{ut} = 1 shows that user u is available at time t, and A_{ut} = 0 shows that user u is not available at time t.
- **Preferences,** $P_{U \times Q}$: A two-dimensional array in which P_{uq} shows the competency preference of user u in role q.

The aim of the platform is to provide a list of up to k recommendations for each user, where a recommendation is of the form [u_1, u_2, [l], [q], t, s] indicating that user u_1 receives a recommendation to connect with user u_2 on a list of topics [l] on a list of roles [q] at time t with a reciprocal score of s. The output of the recommender system is a list of N **Recommendations,** $Recoms_N$ that include up to k recommendations for each learner. The platform can use *Recoms* to display a set of recommendations to each learner.

2.3 Defining a Competency Preference Model and Compatibility Function

In the current system, compatibility is a function of learners' requests (R), competencies (C), availability (A), and preferences (P). R and A are considered as hard constraints. As such the compatibility score $s_{u_1u_2}$ of two users u_1 and u_2 that have incompatible requests or availability is set to a small constant value ϵ. This is done via Algorithm 1, which is defined in Section 2.4.

$s_{u_1u_2}$ is computed as the product of two factors: (1) the effectiveness of a study session based on the learners' competencies and (2) the preferences of u_1. In computing $s_{u_1u_2}$, only the preferences of u_1 are considered. Section 2.4 later discusses how $s_{u_1u_2}$ and $s_{u_2u_1}$ may be combined towards computing a reciprocal score.

The first factor proposes that for a session to facilitate effective learning, the peers' joint competency should be above a certain threshold, set by a parameter τ. This may be set by one of the course instructors or using a validation set. We define the joint competency of a partnership between u_1 and u_2 on a topic l as the magnitude of the vector of their competencies on topic l in a two-dimensional Cartesian space given by:

$$J_{u_1u_2l} = \sqrt{C^2_{u_1l} + C^2_{u_2l}} \tag{1}$$

$J_{u_1u_2l}$ is used in a logistic function H to compute the extent to which the partnership of u_1 and u_2 on topic l meets the expected desirable threshold determined by τ. A scaling parameter α is used to determine the leniency of this measure for a pairing of u_1 and u_2 on a topic l such that $J_{u_1u_2l} < \tau$. This leniency may be useful in reducing orphaned users during implementation in sparse cohorts.

$$H(u1, u2, l, \tau, \alpha) = \frac{1}{1 + e^{-\frac{J_{u_1u_2l} - \tau}{\alpha}}} \tag{2}$$

The second factor is based on the preferences of u_1, which are shown by a vector P_{u_1}. In this vector, P_{u_1q} shows the competency preference of user u_1 for role q. For example for, p_{u_11} = -10 means that u_1 prefers providing support to peers with a competency of around 10 less C_{u_l1} and p_{u_12} = 30 means that u_1 prefers seeking support from peers that have a competency of around 30 more than C_{u_1l}. To be able to provide meaningful recommendations, we constrain eligibility by role such that users (1) provide support to less competent learners, (2) seek support from more competent learners and (3) find study partners with relatively similar competency to that of their own.

The contribution to the compatibility score $s_{u_1u_2}$ in a topic l with a role q is calculated as the height of the Gaussian function G at value C_{u_2l} with centre $C_{u_1l} - P_{u_1q}$ with a standard deviation of σ on a 100 point scale. The parameter σ models the leniency of users in terms of being matched with peers that do not exactly fit their specified preferences.

$$G(u_1, u_2, q, l, P_{u_1q}, \sigma) = 100e^{\frac{-(C_{u_2l} - (C_{u_1l} - P_{u_1q}))^2}{2\sigma^2}} \tag{3}$$

Finally, $s_{u_1u_2}$ is calculated as the product between the two values from G and H, summed over all matched topics and u_1's associated role in each topic as shown below.

$$s_{u_1u_2} = \sum_{(l_i, q_i) \in ([l],[q])} H(u_1, u_2, l_i, \tau, \alpha) \times G(u_1, u_2, q_i, l_i, P_{u_1l_iq_i}, \sigma) \tag{4}$$

2.4 Reciprocal Peer Recommendation

Algorithm 1 presents a reciprocal peer recommendation algorithm. This algorithm takes R, C, A, P and k as input and generates a list of up to k recommendations for each user in the entire cohort. The algorithm selects a user $u1$, then for each other user (referred to as $u2$) uses A to find a mutually convenient time slot for a session between $u1$ and $u2$. R is used to find a set of matching roles and associated matching topics. The reciprocal score, $Score[u2]$, between users is calculated as the harmonic mean of the compatibilities from user $u1$ to $u2$ and vice-versa (ref: competency preference model in section 2.3). Compared to the arithmetic mean, the harmonic mean guarantees to provide a much smaller reciprocal score for users whose compatibilities differ considerably. Therefore, the algorithm would prioritise recommendations that benefit both users [7]. For users that don't satisfy constraints R and A, $Score$ is set to ϵ.

Figure 1 shows the distribution of competency preferences and the reciprocal scores for $u1$ and $u2$ on topic l, where $P_{u_1 1} = -10$ ($u1$ is providing support and prefers users who have competency 10 points lower than $C_{u_1 l}$) and $P_{u_2 2} = 80$ ($u2$ is seeking peer support with a preference for those who have competency 80 points higher than $C_{u_2 l}$). The extent to which both users are recommended to each other is defined by the harmonic mean distribution shown in the third frame of Figure 1.

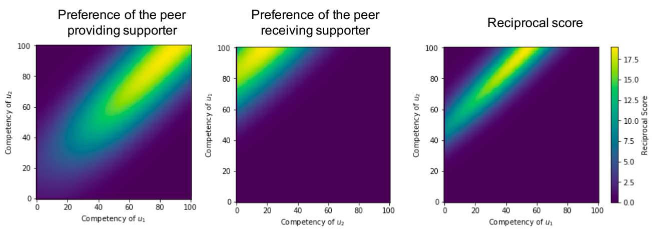


Figure 1: Distribution of competency preferences and the reciprocal scores for $u1$ and $u2$.

Once all of the reciprocal scores in $Score[U]$ have been computed, the k users with the highest score are selected. For any of these users that are compatible with $u1$ (i.e., their reciprocal score is bigger than ϵ), a recommendation is appended to $u1Recoms$. The set of the recommendations for each user are then added to *Recoms*, which store the recommendations for the entire cohort.

Algorithm 1: Generating a list of reciprocal recommendations for the entire cohort

Input : $R, C, A, P, k, \tau, \alpha, \sigma$
Output: *Recoms*

$U \leftarrow getusers(R)$;
$Recoms \leftarrow []$;
foreach $u_1 \in U$ **do**
  $u_1Recoms \leftarrow []$;
  $score[U] \leftarrow \{0\}$; $topics[U] \leftarrow \{\}$; $roles[U] \leftarrow \{\}$; $time[U] \leftarrow \{\}$;
  foreach $u2 \in \{U - u1\}$ **do**
    $time[u_2] \leftarrow findMatchingTime(u_1, u_2, A)$;
    $topics[u_2] \leftarrow findMatchingTopics(u_1, u_2, R)$;
    $roles[u_2] \leftarrow findMatchingRoles(u_1, u_2, R)$;
    $RevRoles \leftarrow findMatchingRoles(u_1, u_2, R)$;
    if $(time[u_2] == NULL)$ $or($ $topics[u_2] == \emptyset)$ **then**
      $score[u_2] \leftarrow \epsilon$
    else
      $Su_1u_2 \leftarrow Compatibility(u_1, u_2, C, P$ $topics[u_2], roles[u_2], \tau, \alpha, \sigma)$;
      $Su_2u_1 \leftarrow Compatibility(u2, u1, C, P$ $topics[u_2], RevRoles, \tau, \alpha, \sigma)$;
      $score[u_2] \leftarrow \frac{2}{Su_1u_2^{-1} + Su_2u_1^{-1}}$
    end
  end
  $topK \leftarrow topKScores(score)$;
  foreach $u \in topK$ **do**
    if *score[u]>0* **then**
      $u1Recoms.append([u1, u, topics[u], roles[u], time[u], score[u]])$;
    end
  end
  $Recoms.extend(u_1Recoms)$
end
return *Recoms*;

3 EVALUATION

Data Sets The synthetic data algorithm generated matrices for R, C, A, and P by specifying U, L, and T for the system. Additional user inputs included the minimum and maximum number of topics (L_{min}, L_{max}) and time slots (T_{min}, T_{max}) as well as the standard deviation for the competencies of the users σ_c. Within the system, competencies where expressed as a value between zero and 100 inclusive. The competency of each user on a topic was generated using a truncated normal distribution using a randomly generated mean competency and the competency standard deviation input. These topics where then sorted by ascending competency and a random number of roles between the bounds supplied in the input were assigned to create a request. Providing support roles were assigned to the highest competencies, seeking support roles to the lowest competencies and co-studying roles to the topic with the median competency of the available topics. Time availability was determined by randomising the number of available time slots within the bounds provided by the input. In the absence of an empirical basis for the amount by which two users should differ in competency to be successful in a peer learning support partnership, we generated learners' competency difference using a vector p_q, the size of which depends on users' preferences for who they are comfortable peer supporting and who they wish to receive peer support by in terms of competency difference, and assume a Normal distribution with a standard distribution of σ_p for generating values in P. **Evaluation Metrics:** The outcome of the system is evaluated using the following criteria: (1) *Scalability*: Based on the runtime and percentage of users that have been recommended at least once by algorithm 1 and (2) *Reciprocality*: Based on precision of reciprocal recommender systems as described in [9] and Section 3.2.

Parameter settings In all experiments the parameters are set using the following default values if not otherwise stated: U = 1000, L = 10, T = 10, R=3, k=5, τ=40, α = 10, σ=10, ϵ = 1, L_{min}=1, L_{max}=5, T_{min}=1, T_{max}=10, σ_c=20, p_q=[-20, 40, 0], and σ_p = 5.

3.1 Scalability: Runtime and Coverage

Figure 2 shows the runtime with respect to different values of U and k. As expected, the runtime of the algorithm increases as U is increased. The runtime of the algorithm is quadratic, $O(n^2)$. The value of k does have a significant impact on the runtime of the algorithm. This is expected as the for loop adding the k recommendations runs in constant time.

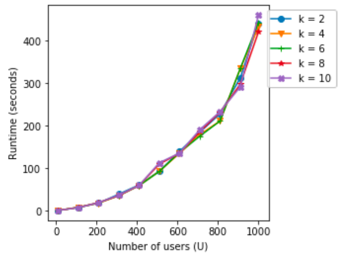
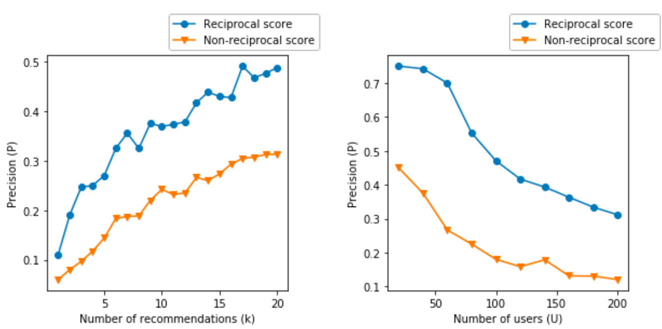


Figure 2: Runtime of Algorithm 1 as the number of learners is increased.

3.2 Reciprocality: Precision

We use the definition of precision from Prabhakar [9]: "learner $u1$ is a successful (reciprocal) recommendation (out of the K-total) for learner $u2$, if and only if $u1$ is also in the top k recommendations of learner $u2$", and evaluate the effectiveness of the reciprocal score against a baseline non-reciprocal score defined as the $S_{u_1 u_2}$ described in Section 2.3. Precision for learner $u1$ is obtained by dividing the number of successful recommendations by k, and the precision of the system is defined as the average precision across all users.

Figure 3a examines precision for U=200 based on different values of k. Figure 3b compares the precision of the platform for k=5 based on different values of U. In each case the precision of the reciprocal score far exceeds the non-reciprocal baseline score. Increasing U leads to a lower precision for both scores. This is expected as an increase in U, while k is kept constant, reduces the probability of a successful reciprocal recommendation.



(a) Changing k **(b) Changing U**

Figure 3: Precision of the platform for a reciprocal and a non-reciprocal score.

4 CONCLUSION AND FUTURE WORK

The findings in these primary results indicate that RiPPLE can provide recommendations across a wide range of competency levels and cohort sizes. Coverage and runtime were heavily dependent on the number of users, number of recommendations per user and threshold settings. Importantly, the precision of recommendations was higher compared to non-reciprocal recommendations for all cohort sizes and variations to the number of recommendations per user.

Following implementation in large introductory courses next semester, subsequent evaluations will address the most significant limitation of the current study - validation in real users - to provide compelling evidence of RiPPLE's capacity to make meaningful recommendations. RiPPLE is designed to support A/B testing so that parallel-group double-blind randomised experiments may be conducted. Our goal is to validate the platform with a control group that would receive random peer recommendations and an experimental group that would receive targeted peer recommendations using the proposed algorithm. This will allow us to determine the impact and quality of recommendations, and also provide insight into the manner in which users select preferences, the conditions upon which they accept or ignore recommendations, how they choose from a list of k recommendations, and otherwise interact with the system.

Variable parameters allow for the recommender to be highly customisable for both administrators, as in setting a minimum joint competency, and for users who can specify their competency preferences according to their perceived needs and/or confidence in providing support. Empirical studies of the competency differences between users providing and receiving peer support through RiPPLE recommendations, and the relationship of these differences with successful learning have further potential to inform new theories of peer learning and shape this aspect of technology enhanced learning systems in future work.

REFERENCES

[1] Lin Chen and Richi Nayak. 2013. A Reciprocal Collaborative Method Using Relevance Feedback and Feature Importance. In *Proceedings of the 2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT) - Volume 01 (WI-IAT '13)*. IEEE Computer Society, Washington, DC, USA, 133–138. https://doi.org/10.1109/WI-IAT.2013.20

[2] Shane Dawson. 2006. A study of the relationship between student communication interaction and sense of community. *The Internet and Higher Education* 9, 3 (2006), 153–162.

[3] Sangeetha Kutty, Richi Nayak, and Lin Chen. 2014. A people-to-people matching system using graph mining techniques. *World Wide Web* 17, 3 (2014), 311–349.

[4] Matthew Lipman. 2003. *Thinking in education*. Cambridge University Press.

[5] Terry Mayes and Sara De Freitas. 2004. Review of e-learning theories, frameworks and models. JISC e-learning models study report. (2004).

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

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

[8] Luiz Pizzato, Tomek Rej, Thomas Chung, Kalina Yacef, Irena Koprinska, and Judy Kay. 2010. Reciprocal recommenders. In *8th Workshop on Intelligent Techniques for Web Personalization and Recommender Systems, UMAP*.

[9] Sankalp Prabhakar, Gerasimos Spanakis, and Osmar Zaiane. 2017. *Reciprocal Recommender System for Learners in Massive Open Online Courses (MOOCs)*. Springer International Publishing, Cham, 157–167. https://doi.org/10.1007/978-3-319-66733-1_17

[10] Francesco Ricci, Lior Rokach, and Bracha Shapira. 2011. *Introduction to Recommender Systems Handbook*. Springer US, Boston, MA, 1–35. https://doi.org/10.1007/978-0-387-85820-3_1

[11] Paula Wilcox, Sandra Winn, and Marylynn Fyvie-Gauld. 2005. âĂŸIt was nothing to do with the university, it was just the peopleâĂŹ: the role of social support in the first-year experience of higher education. *Studies in higher education* 30, 6 (2005), 707–722.

[12] Chun-Mei Zhao and George D Kuh. 2004. Adding value: Learning communities and student engagement. *Research in higher education* 45, 2 (2004), 115–138.