Final Report

Jaime D. Peña
jdpena@gatech.edu

1 INTRODUCTION

Educational institutions of all levels offer different forms of academic assistance to address the unmet needs of students. Common examples of the services offered are academic advising, supplemental instruction (tutoring), student organizations, remedial courses, mentorship, and counseling, among others. These programs have been shown to foster a sense of community among students, bolster their confidence, and improve their academic performance (Todd, 2002). Casazza and Silverman (2013) further note that meaningful access to academic assistance is "imperative to increase graduation rates, develop an educated workforce, strengthen the economy, and compete globally".

The last decade has brought a fundamental paradigm shift to how education is delivered. Typical classroom settings are rapidly being replaced for their online counterparts, as evidenced by the swift rise of Massively Open Online Courses (MOOCs) and Online Master of Science (OMS) programs (Toven-Lindsey, Rhoads, and Lozano, 2015). Naturally, this change has also affected the way academic assistance is provided to students. Services that were once offered only in-person have now adapted to be delivered through online mediums. Some services have even succumbed to automation – removing the need for advisors, instructors, teaching assistants, and the like. Selecting which colleges to apply to or which courses to take, are common examples of academic assistance services that previously required the aid of a guidance counselor or advisor, but no longer do. Thus, academic assistance can now be delivered on-demand and at scale.

Recent trends and advances in technology and algorithmic development have given way to academic assistance delivered via recommender systems (RSs). As its name implies, an RS is a software, application, or technique, that provides suggestions, or recommendations, to users (Ricci, Rokach, and Shapira, 2011). Recommender systems are usually comprised of two components: the data and its recommendation mechanism (algorithm). An RS that assists students in course selection would, for example, use historical student course rankings (the data) and a clustering-based algorithm (recommendation mechanism) to find similarities between the current student user and other students in the data to recom-

mend courses. Recommender systems are typically categorized into four distinct classes, namely, collaborative, content-based, knowledge-based, and hybrid approaches (Al-Badarenah and Alsakran, 2016). These classes differ based solely on how their recommendations are made, each having its own strengths and weaknesses. Their flexibility and practicality, among other benefits, have led to the growth in popularity of RSs and their ubiquitous use.

As grandiose as they seem to be, however, RSs are not foolproof. Recommender systems are typically known as black-box models because the data they're trained on, their recommendation mechanism, and the scoring criteria they use to make suggestions, are usually unknown to the user. This can be problematic to students because academic assistance needs to support their individual needs. In addition, RSs can output recommendations that are not tuned, or optimal, for students as they aim to generalize from data – and are therefore subject to bias (Adomavicius et al., 2014) – or use generic scoring mechanisms. Moreover, as will be described in Section 2, there are a slew of RSs that provide different forms of academic support, however, they are scattered in one-off projects, leaving students wondering as to which one to use and how to use it. Thus, there is clear room for improvement for RSs that aim to provide valuable academic assistance.

To address these shortcomings, I present the `Recommender Room`, or `Rec Room`[1], for short, a platform for providing students with a wide variety of academic assistance from a unified collection of recommender systems. To do this, the `Rec Room` makes available two interfaces: one for the user and one for RSs. The former allows students to select which RSs they want to use and let them tune their output based on their personal preferences. The latter provides standard requirements for RSs to interact with the `Rec Room` and allow them to share datasets. The modular platform will enable additional RSs to be incorporated over time, providing students a continuous ability to receive academic assistance. The `Rec Room` will be more thoroughly discussed in Section 3.

The remainder of this paper is organized as follows. Section 2 provides a review of current to recommender systems, their application to academic assistance, as well as their benefits and limitations. Section 3 describes the `Rec Room` in depth.

1 The `Rec Room` abbreviation is intentional, as it is also a common abbreviation for the term Recreation Room, "a room used for a variety of purposes, such as parties, games and other everyday or casual use" (Wikipedia, 2020).

Two RS use cases are presented in Section 4. Lastly, Section 5 reviews my accomplishments in the class and Section 6 concludes.

2 RELATED WORK

The objective of a recommender system is simple: provide users with suggestions on a particular problem using domain knowledge and relevant user information. Recommender Systems have been around been around for several decades, though the concept of providing recommendations have been around for several millennia (Sharma and Singh, 2016). Their tremendous success in the commercial space and their potential impact in academic assistance are discussed below.

2.1 Recommender Systems

Lü et al. (2012) note that the ongoing rapid expansion of the Internet has greatly increased the use and effectiveness of recommender systems. This can be evidenced some of the largest technology and internet companies. Google's search engine provides users with recommendations on web-pages based on their query (Brin and Page, 1998). Amazon recommends items for purchase on its website based on a customer's previous purchases (Linden, Smith, and York, 2003). YouTube recommends "personalized sets of videos to users based on their activity on the site" (Davidson et al., 2010). Similarly, Spotify creates highly personalized song and playlist suggestions based on "historical and real-time listening patterns" (Jacobson et al., 2016). Lastly, Netflix – a company often considered synonymous with the term "recommender system", recommends movies based on similar user's movie preferences (Gomez-Uribe and Hunt, 2015).

2.2 Recommender Systems for Academic Assistance

Unfortunately, RSs that provide academic support do not have the exorbitant amount of data that is available to the aforementioned commercial companies. Instead, they must focus on specific tasks relying on student preferences, surveys, and data published by academic institutions.

For example, in course recommendation, Al-Badarenah and Alsakran (2016) and Huang et al. (2019) created RSs that focused on recommending optional (elective) courses to students based on rules and heuristics that used cross-student preferences and grade distributions among similar students. In contrast, the course RSs by Grewal and Kaur (2016) and Bydžovská (2016) aimed at recommending a

series of courses, both required and optional, by clustering student preferences (i.e., job interests, degrees, etc.) and training neural networks to output course recommendations for an entire curriculum. Wormald and Guimond (2012) proposed an RS that suggests course scheduling for university administrators using constraint-based satisfaction techniques.

Sikka, Dhankhar, and Rana (2012) stated that development of RSs in the area of e-learning, or online education, should focus on finding "beneficial learning activities to enhance online learning" and "shortcuts ... to resources to help users better navigate the course materials". Comprehensive reviews by Klašnja-Milićević, Ivanović, and Nanopoulos (2015) and Tarus, Niu, and Mustafa (2018) discussed an assortment of techniques RSs use for improving e-learning by recommending relevant and useful learning materials to students. Lin et al. (2019) proposed a similar approach for recommending learning resources to students and teachers, noting that their approach was "instructive for the intervention and guidance of students with weak foundation".

Engin et al. (2014) propose an RS for recommending scholarships to students. The RS presented by Fong and Biuk-Aghai (2009) assists students in selecting which universities to apply to. Park (2017) recommends college majors, minors, and concentrations, to students with the proposed RS. Pan and Li (2010) developed an RS that recommends research publications which "greatly help researchers to find the most desirable papers in their fields of endeavor". Several job RSs are surveyed by Siting et al. (2012) to aid students find jobs post-graduation.

It is evident that there are a myriad ways that RSs can provide students with academic assistance. As discussed previously, however, these RSs are decentralized and their recommendation techniques and scoring criteria are unknown to the user. This problem is further exacerbated by the fact that the RSs themselves and their data are not often made available to students, but rather just used for research and publication purposes. These issues leave students with several hurdles to climb if they are to benefit from them. The two RS platforms, of notable success, that are most similar to the `Rec Room`, are MyMediaLite by Gantner et al. (2011) and CourseRank by Koutrika et al. (2009). The former is a "fast and scalable, multi-purpose [recommender system] library", which provides a centralized location of algorithms used to build general-purpose RSs. The latter is a, now defunct, social website that aimed to provide crowd-sourced course recommendations to students at Stanford University. The `Rec Room` differs from these

platforms as it focuses on providing a unified location for recommendations specific to academic assistance, in an interpretable fashion, tuned to student needs, and not doing so as part of a social network.

3 THE REC ROOM

The `Rec Room` is a platform that provides students with a wide variety academic assistance through a unified collection of recommender systems. These RSs provide users valuable, on-demand access to the same academic support services traditionally offered only in-person. Thus, students can use the `Rec Room` to gather useful, high-quality, and relevant recommendations to assist them navigate their academic careers.

The `Rec Room` was inspired by OpenAi's Gym, a toolkit for "developing and comparing different reinforcement learning (RL) algorithms" (Brockman et al., 2016). The Gym developers noted there was a need for providing the RL research community with a standardized RL environment and the ability to compare their agent's performance across different benchmarks consistently and efficiently. In brief, Gym provides a unified RL platform where users can select from an assortment of environments they wish to solve (i.e., Atari games, MuJoCo, lunar lander, etc.), and then develop RL agents to solve them.

Similarly, the `Rec Room` presents users with a variety of recommender systems (environments), which provide academic assistance. The RSs provide targeted support to address the user's needs. Furthermore, the platform is transparent about how each RS comes to a suggestion, by allowing users to view the data the RS uses and granting them the ability to tune the RS scoring criteria. Users will be more inclined to follow (or not follow) the recommendation of an RS if they could see what factors contributed to their decision making. Like Gym, the `Rec Room` is extendable by exposing a common, shared interface, enabling new RSs to be integrated continuously. The overall architecture of the `Rec Room` is presented in Figure 1 and consists of six major components: (1) the User, User Preferences, and the GUI; (2) the REST API; (3) the Datasets and RS Catalog; (4) the Recommendation Engine; the (5) Output and User Feedback; and the (6) User-defined Models. Each will be discussed further in the following subsections.

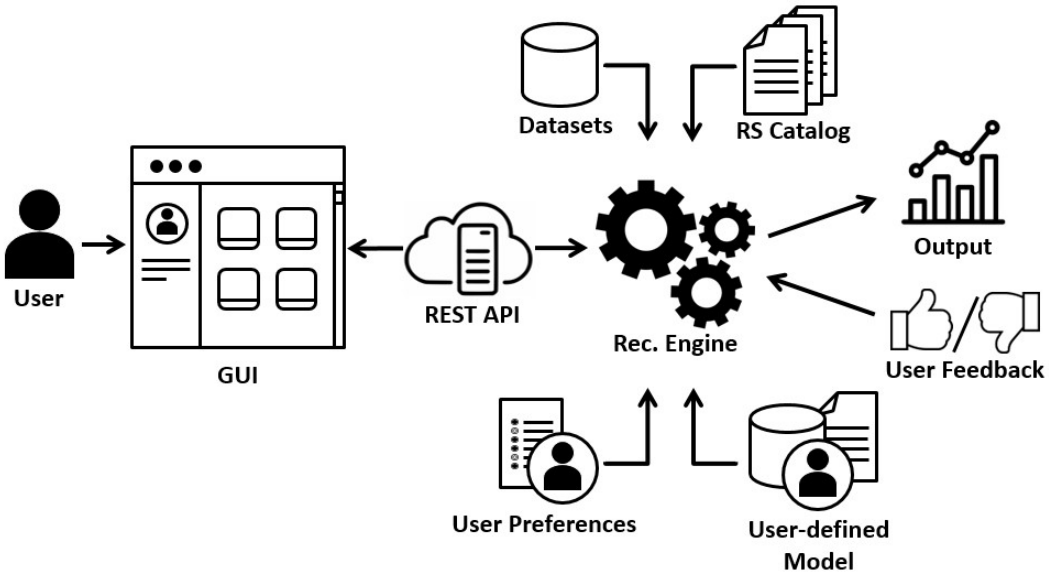
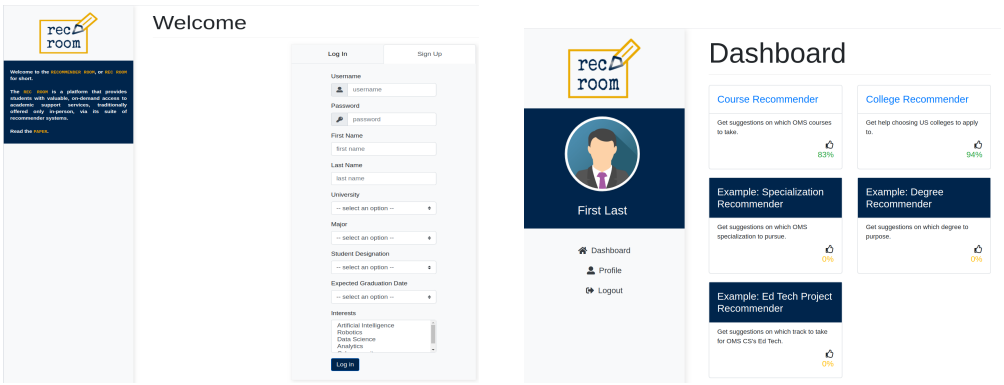


Figure 1—The `Rec Room` architecture. The general workflow of this architecture enables users to set their preferences, select the RS that meets their needs, receive interpretable recommendations, and provide feedback to the system.

3.1 The User, User Preferences, and the GUI

The web-based, graphical user interface (GUI) will be the primary interaction method between users and the `Rec Room`. The GUI was created using a combination of HTML, Bootstrap CSS, and JavaScript/jQuery. In the Login page, users can create an account and answer a general student questionnaire. The questionnaire will provide the `Rec Room` with background information on the user, such as their name, major, interests, etc. These user preferences will be shared amongst the available RSs for potential use. Once the user has created their profile, they will be redirected to the Dashboard page, where they can proceed to choose any of the academic support services offered via the `Rec Room`'s RS catalog. Figure 2-a and -b are screenshots of the Login and Dashboard pages, respectively.

Once a user has clicked on a specific RS, they will be forwarded to the RS-specific page. Here, the user will be presented with 3 key components: the RS Metadata, Parameters, and Results. The Metadata will present the user with additional information about the RS and provide a clickable link to view its dataset(s). The Parameters will render the RS-specific criteria (as HTML) it needs to produce a recommendation. Lastly, the Results will display the recommendation output as



(a) Login Page

(b) Dashboard Page

Figure 2—The `Rec Room`'s Login and Dashboard pages. (a) Shows the questionnaire users are prompted with when creating a new account. (B) Displays the RSs available from the RS Catalog.

an RS-specific chart (plot). Figure 3-a and -b in Section 4 depict the RS-specific webpages.

3.2 REST API

The backend web-server of the `Rec Room` consists of a REST[2] API developed in Python 3.7 using the Flask[3] web application framework. The REST API is the intermediary between the GUI and the `Rec Room`'s Recommendation Engine. As such, the REST API is tasked with parsing user input, delegating actions appropriately to the Recommendation Engine, and formatting the returned output. This ensures a strong decoupling between frontend GUI and backend model. The REST API exposes four (4) endpoints which receive and respond to HTTP GET and POST requests. The full specs of each endpoint are documented in the `Rec Room`'s project github[4].

3.3 Datasets and RS Catalog

The RS Catalog, akin to a library, stores the collection of RSs that are currently integrated into the `Rec Room`. Figure 2-b, specifically, depicts how the RS catalog is rendered to the user. Each RS "card" contains a title, a brief description of the RS, and a favorability (%) rating. The title of each RS is a link to its specific RS page

2 Representational state transfer (REST) software architecture
3 https://flask.palletsprojects.com/en/1.1.x/
4 https://github.gatech.edu/jpena34/rec-room

– a page where users can obtain recommendations based on what the RS offers. This enables the `Rec Room` to provide academic assistance on a wide variety of tasks, since each RS is developed independently of each other. Therefore, users can search for a specific RS that meets their needs, click it, and proceed to receive recommendations on their chosen topic. Section 4 will present two examples of how the RSs work once a user has clicked on them.

3.4 The Recommendation Engine

The central piece of the `Rec Room` is its Recommendation (Rec) Engine, where information regarding the user's preferences, the RS selected, the data to be used, and what scoring criteria to use, is brought together to output a recommendation. The Rec Engine interacts with each RS in the RS Catalog, by calling specific properties (attributes) and methods (functions). To ensure compatibility with the `Rec Room`, each RS must comply with the Rec Engine's implementation requirements (API), by inheriting from an abstract base class and defining appropriate properties and methods. This ensures each RS follows a uniform interface which can be called at runtime to generate content dynamically. More concretely, the RSs must define a *recommend*, *render*, and *visualize* method. The output of *recommend* triggers a recommendation given the user input, while the latter two populate the Parameters and Results components of each RS-specific page, respectively, as discussed in Section 3.1. Thus, the Rec Engine makes no assumption about the implementation details of any of the RSs, leaving the specific recommendation techniques and/or paradigms used to be decided by the RS developer.

3.5 Output and User Feedback

Each RS will output a single or set of recommendations. How these recommendations are rendered, however, is entirely RS-specific. As mentioned in Section 3.4, each RS must define specific methods to be called at runtime. In this case, the *visualize* method is invoked, which returns a configuration that will parameterize a chart.js[5] chart. These charts could be, for example, a radar plot or bar graph. Furthermore, once a recommendation is returned, a user can provide feedback as to whether they like or dislike the result by clicking the "thumbs-up" or "thumbs-down" button. Each RS can use this feedback to bolster its recommendation capabilities.

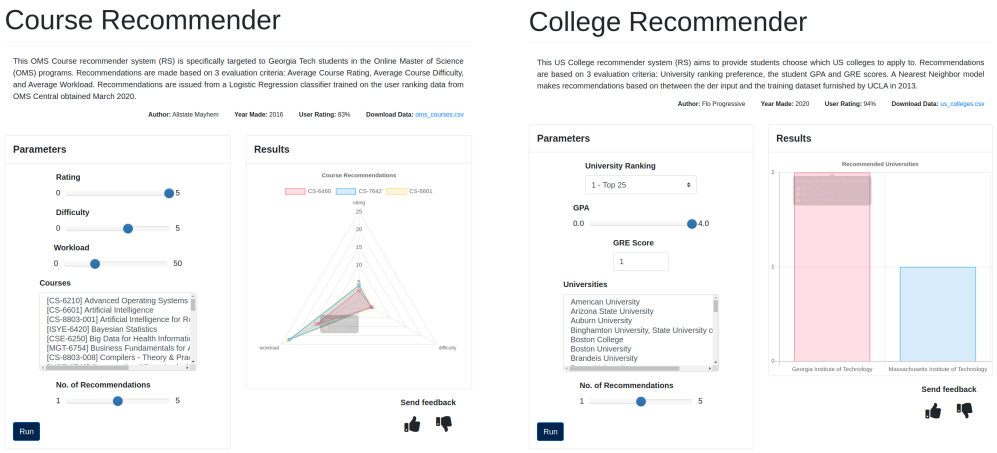
5 https://www.chartjs.org/

3.6 User-defined Model

Lastly, the `Rec Room` is designed to be a modular and extendable platform. This means that new RSs can be developed and be integrated into the `Rec Room` easily and efficiently, as long as they adhere to the required API. This is a key feature of the `Rec Room` because it allows for continuous updates to existing RSs and the development of new and different RSs. A user, now being a developer, can upload their own datasets (or leverage one already provided) and create a new RS. The resulting RS can be integrated into the `Rec Room` platform and be available for others to use.

4 EXAMPLE RS USE CASES

This section presents two recommender systems that were developed from scratch and integrated into the `Rec Room`. Namely, the **OMS Course Recommender System** and the **US College Recommender System**. These RSs intend to showcase the flexibility, extendibility, and more so, the potential of the `Rec Room`.



(a) OMS Courses RS (b) US College RS

Figure 3—The two recommender systems integrated into the `Rec Room`. (A) The output corresponds to a 5 Rating, 3 Difficulty, and Workload of 15, with CS-6460, CS-7642, and CS-6601 as the courses being considered. CS-6460 (red) was the best recommendation followed by CS-7642 (blue), and CS-6601 (yellow). (B) the results reflect user input of "1 - Top 25" for University Ranking, 3.5 GPA, and 640 GRE, with Georgia Tech, MIT, and Brandeis University being considered (only the top-two choices were selected). Georgia Tech (red) was the best choice, followed by MIT (blue).

4.1 OMS Courses Recommender System

The OMS Courses RS aims to help students enrolled in Georgia Tech's Online Master of Science (OMS) programs choose their courses. OMS students are remote learners. As such, it can be difficult for them to receive formal advising about which courses to take as compared with their on-campus counterparts. Consequently, OMS students must resort to various external and informal resources to inform their decisions, such as Reddit or OMS Central[6]. This, too, is not ideal, as the information they need is scattered between different webpages, links, posts, reviews, etc. Furthermore, these resources offer only guidance. This RS simplifies matters by providing OMS students with a simple tool where they can obtain guidance, and better yet, specific course recommendations.

The OMS Courses RS makes use of data[7] from OMS Central. It outputs course recommendations based on 3 criteria: average student (favorability) rating, average course difficulty, and average course workload (hours per week). Rating is defined on a scale between 1 – 5, with 1 being the least favorable and 5 being the most favorable. Difficulty is calculated similarly, with 1 being the least difficult and 5 being most. Workload spans 0 – 50 hours per week. Users can subsequently select all courses they are interested in taking from the OMS program. This criterion is displayed to the user in the Parameters component of the OMS Courses RS page. Lastly, they can set the number of results they would like returned, and run the recommendation. The OMS Courses RS is a logistic regression classifer, trained on the corpus of OMS Central data, which takes in the user-specified criterion and outputs course recommendations. In this example, the RS renders a radar chart in the Results component. The legend color-codes each recommendation (course) and orders them from best fit to least fit (left to right). The three axis of the radar chart correspond to the three criterion used by the RS. Upon hovering, a user can see tooltip metadata about the specific rating, difficulty, or workload a course had. It is important to note that this process makes use of the required methods defined in Section 3.4. That is to say, the OMS Courses RS specifies the input criteria required, rendered in the manner it wants (i.e., sliders and multi-select box), and displays the results in its own unique way (radar chart). When finished, the user can proceed to "like" or "dislike" the recommen-

6 https://omscentral.com/

7 kindly provided by Mehmet Bajin, current maintainer for OMS Central on March 2020.

dation but clicking the appropriate thumbs up/down button. Figure 3-a depicts the output of a recommendation from the OMS Courses RS.

4.2 US Colleges Recommender System

The US College RS aims to help students in selecting which US colleges/universities to apply to. Students are faced with difficult choices when deciding which colleges to apply to. This is true for both high school students applying to college and undergraduate students applying to grad school. Typically, students meet with advisors to help make informed decisions. However, advisors are not always available and may not know the student well enough to make optimal suggestions. Because of this, students may feel discouraged from engaging in meaningful dialogue about their college aspirations. Furthermore, students may feel uncertain about meeting the acceptance standards of certain colleges, leading to an overwhelming fear of (potential) rejection. This RS simplifies matters by providing recommendations to students about which colleges from their preferences are the best matched for their qualifications.

The US College RS makes use of data obtained from a UCLA course[8]. This data was aggregated with university names for demonstration purposes only. It outputs college recommendations based on three (3) criteria: GPA, GRE score, and university ranking. GPA is defined between 0 – 4.0, with 4.0 being the maximum GPA. GRE scores are on a scale of 0 – 800, with 800 being the maximum score. University rankings are binned into four (4) categories, 1st tier through 4th tier, with 1st tier being the most "prestigious" universities (according to the dataset). Users can subsequently select all colleges they are interested in applying to. As will be with any RS integrated into the `Rec Room`, this criterion is displayed to the user in the Parameters component of the RS-specific page. The user can then specify the number of results they would like returned and run the recommendation. The US Colleges RS is a Nearest-Neighbor clustering model, which scatters each college in the dataset into a 3-dimensional space (1 dimension per evaluation criteria), and outputs the universities most similar (i.e., shortest distance) to the user input. For this example, the RS renders a bar graph in the Results component. The legend, again, color-codes each recommendation (college) and plots them in order, from most similar to least (left to right). The y-axis is simply the college similarity ranking, in terms of the number of which is the best choice

8 https://stats.idre.ucla.edu/r/dae/logit-regression/

(highest value) to worst (lowest value). Hovering over the bars will display a tooltip with the associated GPA, GRE, and Ranking of the college. Again, the information rendered in the Parameters component (i.e., sliders, input fields, and multi-select box) and the Results component (bar graph) is generated dynamically by the US College RS. Finally, the user can provide the RS feedback based on their satisfaction with the recommendation. Figure 3-b shows the output of a recommendation from the US Colleges RS.

5 SEMESTER ACCOMPLISHMENTS

This semester, I proposed to build the `Rec Room` as part the development track. I really enjoyed my time working on this project and the research associated with it. The milestones I set for myself, summarized below, were fully accomplished. Similarly, the number of hours it took (115 hours) was also inline with my projections. In all, this class was a very rewarding and enjoyable experience.

1. Functioning `Rec Room` prototype, including GUI and API.
2. Two RSs integrated into the `Rec Room` (including their datasets).
3. Thorough documentation for installation, use, and development.

6 CONCLUSION

Academic assistance provides students with additional support to advance their academic careers. Recommender systems have shown promise in delivering this support. These systems, however, are decentralized, operate as black-boxes, and may not be optimal to address the individual needs of each student. Thus, there is clear room for improvement. The `Rec Room` aims to fill this gap by providing a wide variety of academic assistance, tuned to their specific needs, all from a centralized location. The codebase and documentation for the `Rec Room` is available on github [9]. I invite the academic community and future Ed Tech students to continue building upon this prototype and bring `Rec Room` to its full potential and provide students with an automated, scalable, and on-demand method for receiving optimal academic assistance.

9 https://github.gatech.edu/jpena34/rec-room

7 REFERENCES

[1] Adomavicius, Gediminas, Bockstedt, Jesse, Curley, Shawn, and Zhang, Jingjing (2014). "De-biasing user preference ratings in recommender systems". In: *RecSys 2014 Workshop on Interfaces and Human Decision Making for Recommender Systems (IntRS 2014)*, pp. 2–9.

[2] Al-Badarenah, Amer and Alsakran, Jamal (2016). "An automated recommender system for course selection". In: *International Journal of Advanced Computer Science and Applications* 7.3, pp. 166–175.

[3] Brin, Sergey and Page, Lawrence (1998). "The anatomy of a large-scale hypertextual web search engine". In:

[4] Brockman, Greg, Cheung, Vicki, Pettersson, Ludwig, Schneider, Jonas, Schulman, John, Tang, Jie, and Zaremba, Wojciech (2016). "Openai gym". In: *arXiv preprint arXiv:1606.01540*.

[5] Bydžovská, Hana (2016). "Course Enrollment Recommender System." In: *International Educational Data Mining Society*.

[6] Casazza, Martha E and Silverman, Sharon L (2013). "Meaningful access and support: The path to college completion". In: *Retrieved from Council of Learning Assistance and Developmental Education Associations website: http://www. cladea. net/white_paper_meaningful_access. pdf*.

[7] Davidson, James, Liebald, Benjamin, Liu, Junning, Nandy, Palash, Van Vleet, Taylor, Gargi, Ullas, Gupta, Sujoy, He, Yu, Lambert, Mike, Livingston, Blake, et al. (2010). "The YouTube video recommendation system". In: *Proceedings of the fourth ACM conference on Recommender systems*, pp. 293–296.

[8] Engin, Gökhan, Aksoyer, Burak, Avdagic, Melike, Bozanlı, Damla, Hanay, Umutcan, Maden, Deniz, and Ertek, Gurdal (2014). "Rule-based expert systems for supporting university students". In: *Procedia Computer Science* 31, pp. 22–31.

[9] Fong, Simon and Biuk-Aghai, Robert P (2009). "An automated university admission recommender system for secondary school students". In: *The 6th International Conference on Information Technology and Applications*. Citeseer, p. 42.

[10] Gantner, Zeno, Rendle, Steffen, Freudenthaler, Christoph, and Schmidt-Thieme, Lars (2011). "MyMediaLite: A free recommender system library". In: *Proceedings of the fifth ACM conference on Recommender systems*, pp. 305–308.

[11] Gomez-Uribe, Carlos A and Hunt, Neil (2015). "The netflix recommender system: Algorithms, business value, and innovation". In: *ACM Transactions on Management Information Systems (TMIS)* 6.4, pp. 1–19.

[12] Grewal, DS and Kaur, K (2016). "Developing an intelligent recommendation system for course selection by students for graduate courses". In: *Business and Economics Journal* 7.2.

[13] Huang, Ling, Wang, Chang-Dong, Chao, Hong-Yang, Lai, Jian-Huang, and Philip, S Yu (2019). "A Score Prediction Approach for Optional Course Recommendation via Cross-User-Domain Collaborative Filtering". In: *IEEE Access* 7, pp. 19550–19563.

[14] Jacobson, Kurt, Murali, Vidhya, Newett, Edward, Whitman, Brian, and Yon, Romain (2016). "Music personalization at Spotify". In: *Proceedings of the 10th ACM Conference on Recommender Systems*, pp. 373–373.

[15] Klašnja-Milićević, Aleksandra, Ivanović, Mirjana, and Nanopoulos, Alexandros (2015). "Recommender systems in e-learning environments: a survey of the state-of-the-art and possible extensions". In: *Artificial Intelligence Review* 44.4, pp. 571–604.

[16] Koutrika, Georgia, Bercovitz, Benjamin, Kaliszan, Filip, Liou, Henry, and Garcia-Molina, Hector (2009). "Courserank: A closed-community social system through the magnifying glass". In: *Third International AAAI Conference on Weblogs and Social Media*.

[17] Lin, Hanhui, Xie, Shaoqun, Xiao, Zhiguo, Deng, Xinxin, Yue, Hongwei, and Cai, Ken (2019). "Adaptive Recommender System for an Intelligent Classroom Teaching Model". In: *International Journal of Emerging Technologies in Learning (iJET)* 14.05, pp. 51–63.

[18] Linden, Greg, Smith, Brent, and York, Jeremy (2003). "Amazon. com recommendations: Item-to-item collaborative filtering". In: *IEEE Internet computing* 7.1, pp. 76–80.

[19] Lü, Linyuan, Medo, Matúš, Yeung, Chi Ho, Zhang, Yi-Cheng, Zhang, Zi-Ke, and Zhou, Tao (2012). "Recommender systems". In: *Physics reports* 519.1, pp. 1–49.

[20] Pan, Chenguang and Li, Wenxin (2010). "Research paper recommendation with topic analysis". In: *2010 International Conference On Computer Design and Applications*. Vol. 4. IEEE, pp. V4–264.

[21] Park, Young (2017). "A Recommender System for Personalized Exploration of Majors, Minors, and Concentrations." In: *RecSys Posters*.

[22] Ricci, Francesco, Rokach, Lior, and Shapira, Bracha (2011). "Introduction to recommender systems handbook". In: *Recommender systems handbook*. Springer, pp. 1–35.

[23] Sharma, Richa and Singh, Rahul (2016). "Evolution of recommender systems from ancient times to modern era: a survey". In: *Indian Journal of Science and Technology* 9.20, pp. 1–12.

[24] Sikka, Reema, Dhankhar, Amita, and Rana, Chaavi (2012). "A survey paper on e-learning recommender system". In: *International Journal of Computer Applications* 47.9, pp. 27–30.

[25] Siting, Zheng, Wenxing, Hong, Ning, Zhang, and Fan, Yang (2012). "Job recommender systems: a survey". In: *2012 7th International Conference on Computer Science & Education (ICCSE)*. IEEE, pp. 920–924.

[26] Tarus, John K, Niu, Zhendong, and Mustafa, Ghulam (2018). "Knowledge-based recommendation: a review of ontology-based recommender systems for e-learning". In: *Artificial intelligence review* 50.1, pp. 21–48.

[27] Todd, Adam G (2002). "Academic support programs: effective support through a systemic approach". In: *Gonz. L. Rev.* 38, p. 187.

[28] Toven-Lindsey, Brit, Rhoads, Robert A, and Lozano, Jennifer Berdan (2015). "Virtually unlimited classrooms: Pedagogical practices in massive open online courses". In: *The internet and higher education* 24, pp. 1–12.

[29] Wikipedia (2020). *Recreation room — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/w/index.php?title=Recreation%20room&oldid=939928605. [Online; accessed 23-February-2020].

[30] Wormald, R and Guimond, C (2012). "Creating a more efficient course schedule at WPI using linear optimization". In: *Major Qualifying Rep, Worcester Polytechnic Institute, Worcester*.