Project Proposal

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, a RS is a software, application, or technique, which 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 recommend courses based on the user's preferences. 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 propose the `Recommender Room`, or `Rec Room`[1], for short, a framework for providing students with a wide variety of academic assistance from a unified collection of recommender systems. To do this, the `Rec Room` will make available two interfaces: one for the user and one for RSs. The former will allow students to select which RSs they want to use and let them tune their output based on their personal preferences. The latter will provide standard requirements for RSs to interact with the `Rec Room` and allow them to share datasets. The modular framework 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

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).

well as their benefits and limitations. Section 3 presents my proposed work for this semester. The project deliverables are described with Section 4, with the scheduled task list in Section 5. Lastly, Section 6 concludes.

2 RELATED WORK

The objective of a recommender system is pretty 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, or online 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 frameworks, of notable success, which that are most similar to my proposed work is 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 Stanford students. My proposed work differs from these frameworks 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 PROPOSED WORK

My proposed work for this semester is building a framework, the `Rec Room`, that provides students with a wide variety academic assistance from a unified collection of recommender systems. The main objective of this project is to provide users valuable, on-demand access to same academic support services traditionally offered by in-person. The `Rec Room` will provide useful, high-quality, and relevant recommendations to assist students in navigating their academic careers.

My proposed framework 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 a standardized research 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` would present users, specifically, students, with a variety of recommender systems that provide can academic assistance. The RSs intend to provide targeted support to address the user's needs. Furthermore, the framework would be transparent about how a RS comes to a suggestion, by allowing users to view the data the being used and granting them the ability to tune the RS scoring criteria. Users would be more inclined (or not) to follow the recommendation of a RS if they could see what factors contributed to their decision making. Like Gym, the `Rec Room` would be 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. The framework consists of five major components: (1) the User, User Preferences, and the GUI; (2) the Datasets and RS Catalog; (3) the Recommendation Engine; (4) Output and User Feedback; and (5) User-defined Models. Each will be discussed further in the following subsections.

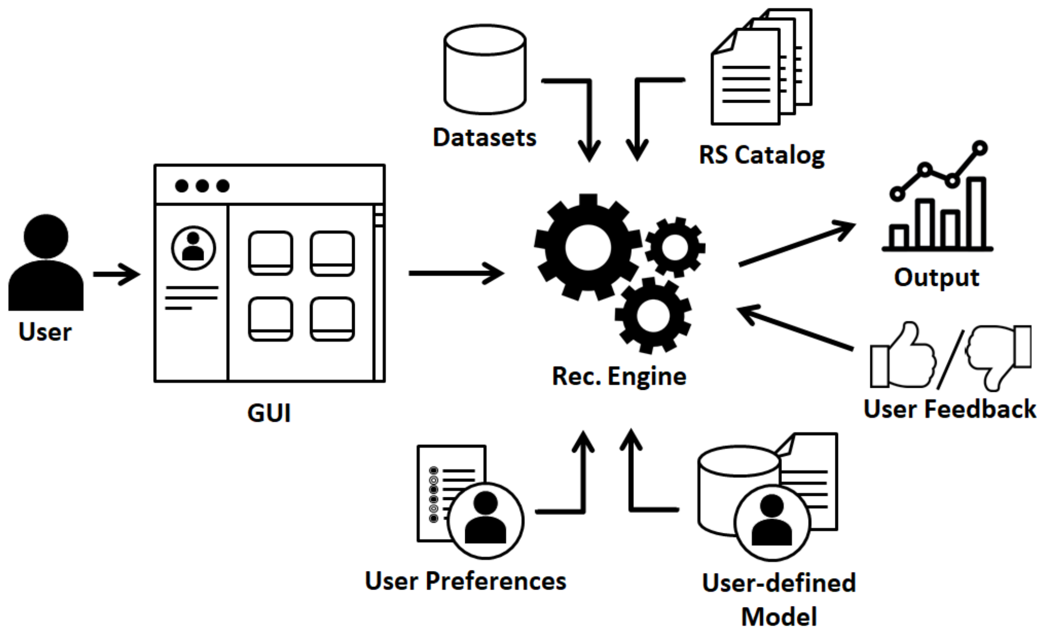

Figure 1—"The `Rec Room` architecture. The general workflow of this architecture enables users to set their preferences, select the RS that meets their needs, receive interpretable recommendations, and provide feedback to the system."

3.1 Example Use-Case

To provide context to the proposed framework, let's imagine that a student, Alice, is choosing which OMSCS[2] courses to enroll in for the next semester. Her academic advisor has not responded to her emails requesting assistance and her time-ticket shows her registration window will begin shortly. Rather than waiting for a response and signing up for the courses that she needs to take to avoid being waitlisted in the meantime, Alice seeks for guidance from the `Rec Room`.

3.2 The User, User Preferences, and the GUI

The web-based, graphical user interface will be the primary interaction method between users and the `Rec Room`. Here, users will be able to create an account and answer a general student survey. The survey will provide the `Rec Room` with background information on the user, such as name, demographic, current GPA, major, their hobbies, etc. These user preferences will be shared amongst the available RSs. Once the user has created their profile, they can proceed to choose any

2 Georgia Tech's Online Master of Science in Computer Science program.

of the academic support services offered via the `Rec Room` RS catalog. After selecting a particular RS to use for their particular needs, the GUI will present the user with the following: (1) descriptions of the RS, the data it uses, and its scoring criteria; (2) additional information needed by the RS (e.g., another survey specific to this RS); and (3) tunable parameters (knobs), that allow the user to define the weight, or importance, of several factors that influence the RS scoring criteria.

In Alice's case, she would first load the `Rec Room` from her browser. She will then create an account and complete the academic background survey. From this, the `Rec Room` learns that Alice is a software engineer, finds cybersecurity research interesting, and is three courses away from graduating from OMSCS with a specialization in machine learning. Now Alice can browse through the available RSs that can assist her in choosing which course to take.

3.3 Datasets and RS Catalog

The `Rec Room` will store a unified collection of recommender systems along with their accompanying datasets to provide academic assistance on a variety of different tasks. The RS Catalog, similar to a library, will be the collection of RSs that are currently integrated into the `Rec Room`. These RSs can use any recommendation technique they would like. To ensure compatibility between RSs and the `Rec Room`, the RS Catalog will expose an API from which RSs can use. This structure guarantees that RSs abide by the `Rec Room`'s implementation requirements, such as proper method signatures and variable definitions. These methods will be called at runtime and send information to the GUI to be displayed. The datasets that each RS uses will be also be stored. The `Rec Room` will grant shared access to its datasets so that multiple RSs can make use of them.

Alice has finally found the RS she needs from the RS Catalog. The RS description states that it can provide assistance in selecting course recommendations for OMSCS students. Upon selecting the RS, a new page is loaded. The new page presents Alice the dataset that the RS will use, the tunable parameters available, and a request for more information. In this case, the data the RS uses is from OMS Central, a website that provides course difficulty and workload rankings to OMS students. Alice notices that the additional information requested specific to the machine learning specialization, since she's already filled that out in the profile page. She proceeds to fill out the additional information. The RS now learns that Alice only needs three courses to graduate: Machine Learning, Graduate Algorithms, and an elective. Lastly, Alice modifies the two knobs that the RS uses for weighing importance. She sets the "time required" parameter to "moderate", since she does not want to take multiple time consuming courses, and sets the "combined difficulty threshold" to 3.75 (on a scale of 1-5, with 5 being the most difficult). Alice is now ready to run the RS.

3.4 The Recommendation Engine

The central piece of the `Rec Room` is its recommendation engine, where information regarding the user, the recommender system selected, the data to be used, and how scoring criteria should be weighted is combined. The engine runs the RS, by calling the proper methods, and provides users with visual feedback as to the state current process (e.g., estimated wait times.). Once finished, the output of the RS is properly formatted and sent to the GUI to be displayed.

After Alice clicked the "run" button, the `Rec Room` quickly provided a recommendation: take Network Security and Machine Learning. Alice wondered how the RS came up with the course suggestions so she scrolled through the OMS Central dataset and found that Network Security has a an average difficulty rating of 3.07 and a workload rating of 12.84 (hours/week) while Machine Learning and Graduate Algorithms have an average difficulty rating of 4.18 and 4.36 and workload rating of 20.88, and 20.57, respectively. Curious to see what would happen if she changed the weightings of the scoring criteria, Alice moved the weighting knobs to set a preference of "high" and 5.0 corresponding to "time required" and "combined difficulty threshold" and re-ran the RS. To her delight, the RS now recommended she take Graduate Algorithms and Machine Learning. This "thought experiment" demonstrated to Alice how the internals of the RS worked. Further, she was confident that the original suggestion that RS had made was a good one.

3.5 Output and User Feedback

The output of each RS will vary depending on the task at hand. Since the `Rec Room` provides a standard API that RSs in its library interact with, it is easy to define a common set of methods that visualize output in a user-friendly manner. For example, the `Rec Room` can provide a "plotting" interface, where RS output can be visualized in the GUI. This enables RSs to provide output data that is formatted to fit a pre-defined set of plotting capabilities, such as spider charts, bar graphs, or heat maps. Moreover, if a user does not agree with the recommendation that is being output by the RS, they can provide feedback to the `Rec Room` so the RS in question can be tuned. In some cases, feedback as simple as "thumbs-up" or "thumbs-down" may suffice in improving the RS for future use.

Alice was content with the recommendation she received. A simple plot showing course rankings allowed her to quickly interpret the recommendations that were output by the RS. Alice provides positive feedback about her experience using the RS. The `Rec Room` can use this information to strengthen the relationship between the output recommendations, the ratings in the dataset, and other users with similar preferences to Alice.

3.6 User-defined Model

Lastly, the `Rec Room` is designed to be modular, extendable framework. This means that new RSs can be developed which can integrate with the `Rec Room` easily and efficiently, as long as they adhere to the required API. This is a very important part of the `Rec Room` because it allows for continuous updates to RSs already available and development of new and different RSs. A user, now being a developer, can bring their own datasets to use and build and test a RS around it. Alternatively, the user can leverage existing datasets to build their own RS. The resulting RS can be integrated into the `Rec Room` framework.

Alice is currently enrolled in CS 6460, Education Technology. She notices that her peers in the class could benefit from an RS that provides project recommendations based on their interests and academic experiences. Thus, she begins to gather historical data on past projects and begins to develop an RS that can be integrated into the `Rec Room`.

4 DELIVERABLES

The deliverables I intend to have at the end of the semester are four-fold:

1. A functioning `Rec Room` prototype. This includes the frontend graphical user interface (GUI) and backend application programmable interface (API). The GUI will provide users the ability to create a user profile, select RSs, and tune the output according to their preferences. The API will allow for modular integration of various RSs and shared datasets. The `Rec Room` prototype will be command-line executable and available as an importable library. The framework will primarily be constructed using Python.

2. Two recommender systems to demonstrate the `Rec Room`'s functionality: One RS for selecting university courses and another for selecting colleges to apply to. Each RS will leverage its own recommendation mechanism, such as clustering techniques and heuristics methods, further demonstrating the `Rec Room`'s flexibility.
3. Datasets, generated or publicly available, which will be included in the `Rec Room`. Their inclusion will allow users to view the data that was used to provide recommendations, as well as provide developers access to the data so they can develop their own RSs.
4. Thorough documentation for installation, use, and development of the `Rec Room` will be provided. This will ease the learning curve for users and developers alike.

These deliverables are in addition to the two intermediate video milestones, final presentation, and final paper. I will use the intermediate milestones to gather feedback from mentor and peers based on my progress and adjust my prototype as needed. My final presentation will be a walkthrough of my completed work. Lastly, the final paper will report on my experience, potential shortcomings, and accomplishments.

For Milestone 1, my deliverables will be:

1. Functioning GUI with the following webpages/endpoints
   - Dashboard
   - Recommender
   - Account
2. Stub examples of an RS and recommendation visualization
3. Account logins/sessions
4. Video demo

For Milestone 2, my deliverables will be:

1. Two functioning recommender systems
   - RS for selecting university courses
   - RS for selecting colleges to apply to
2. Add the ability to view the datasets via new webpage/endpoint
3. Video demo

5 TASK LIST

My proposed work is divided into three (3) tasks: (1) Building the `Rec Room` framework; (2) Developing the RSs, integrating them into them, and testing them; (3) Preparation for the final deliverables and last modifications to the prototype. I am the only contributor to this proposed work, so I excluded the "Member Responsible" column. Note that my schedule is for a total of 115 hours.

Week #	Task #	Task Description	Estimated Time (Hours)
8	1	Begin structuring code Begin scoping GUI requirements Begin development of the backend API	15
9	1	Begin creating GUI wireframe and templates Begin instantiating flat-file caching Continue development of the backend API	15
10	1	Flesh out GUI and visualization components Enable user sessions Integrate frontend GUI and backend API Prepare Milestone 1 deliverable	15
INTERMEDIATE MILESTONE 1 DUE			
11	2	Refine prototype based on Milestone 1 feedback Finding or generate dataset(s) for the RS#1 Begin developing RS#1	15
12	2	Integrate RS#1 to backend API Integrate RS#1 to frontend GUI Test RS#1	10
13	2	Finding or generate dataset(s) for RS#2 Begin developing RS#2	10
14	2	Integrate RS#2 to backend API Integrate RS#2 to frontend GUI Test RS#2 Prepare Milestone 2 Deliverable	15
INTERMEDIATE MILESTONE 2 DUE			
15	3	Refine prototype based on Milestone 2 feedback Refine RSs based on Milestone 2 feedback Fix any outstanding, incomplete, broken features Prepare the project presentation Prepare the project paper Prepare the final project Prepare the project documentation	10
16	3	\tabitem Fix any outstanding, incomplete, broken features Finalize the project presentation Finalize the project paper Finalize the final project Finalize the project documentation	10

6 CONCLUSION

Academic assistance provides students additional support to advance their academic careers. Recommender systems have shown promise in providing academic support to students. These systems, however, are decentralized, operate as black-boxes, and may not be optimal to address an individual student's needs. Thus, there is clear room for improvement. The `Rec Room` aims to fill this gap by providing students the ability to receive a wide variety of academic assistance, tuned to their preferences, all from a centralized location. A clear outline for the development of the `Rec Room` framework was presented, demonstrating its feasibility as a solution. The `Rec Room` has the potential to provide students with an automated, scalable, and on-demand method for receiving optimal academic assistance.

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*.