Course Recommendation as Graphical Analysis

Connor Bridges, James Jared, Joshua Weissmann, Astrid Montanez-Garay,
Jonathan Spencer, Christopher G. Brinton
Princeton University, Princeton, New Jersey 08544
Email: {connorjb, jjared, weissj, am33, j.spencer, cbrinton} @princeton.edu

***Abstract*—This work proposes a method for course recommendation using grade and enrollment data. We analyze the per-semester sequence in which courses are taken in order to create a personalized course transition graph that balances the student's current grades, their expected improvement, and course popularity. Using a dataset of 6000 students and 1500 courses, we compare the recommended trajectories of top performing and low performing students to show that popularity alone is a strong heuristic for recommending successful trajectories.**

I. Introduction

Deciding which set of courses to take can be quite challenging for both veterans and newcomers in the higher education world. Despite helpful tools like course reviews and university specific scheduling applications, students may spend several hours sifting through the available options. Because of the large variety of course offerings, scheduling limitations, and worries about grades, a student may leave college having missed out on several courses that could have positively influenced their trajectory through school.

In this paper, we explore methods to recommend courses for students that will have a positive impact on their overall college experience. The system we introduce gathers historical grade and enrollment data to form a directed graph describing the temporal transition students make between different sets of courses. Using this graph and a student's past grade data, we make personalized suggestions about which course to take next by balancing course popularity and the grade we expect them to get in each course.

A. Related Work and Contributions

Although recommender systems began in the context of product recommendations, they have enjoyed wide use in education generally as well as course selection specifically [1], [2]. Broadly, recommender systems have been useful in personalized learning systems and Massive Open Online Courses (MOOCs) to craft content within a particular course to best fit the needs of the learner [3]. For course level recommendations, many techniques use survey data as a way to precisely label aspects of each course, such as difficulty, workload, or relevancy to a particular goal [4]–[6]. Some researchers use these ratings to estimate students ability levels, and make recommendations based on the best match for ability to difficulty [6], while others simply present all available meta-information to the user in an effort to inform their choice [5]. These supervised techniques are useful when explicit survey feedback data is available, but when survey data is unavailable, recommender systems must infer user preferences using enrollment history. Some approaches use student enrollment history to determine pairwise similarities between courses, then recommends users take the courses most similar to ones they had previously taken [7]–[9]. The most common algorithms used in course recommendation are collaborative filtering and association rules [10].

Our approach differs from the related work in a few key ways. In this work we operate using implicit feedback data rather than explicit feedback data, making use of both course enrollment and course performance grade data. Although this results in less straightforward evaluation metrics, this type of a system is immediately broadly applicable to nearly every course grade dataset. In addition, current approaches make recommendations with no regard to the order in which students take courses. This work differs by considering transitions between courses and recommending the course a student should take *next*, rather than recommending a course a student should take *sometime*.

II. Recommendation algorithm

A. Overview

The recommendation algorithm we propose takes a graphical approach, studying transitions between classes. We use student transcript data over the past few years to build a subsequency graph, from which the recommendation algorithm extracts the most optimal transition. The nodes of this directed graph are the various courses offered by the university, and the edges point from a class that any student took in one semester to a class that the same student took in the subsequent semester. The weight of each graph edge entering a node can be thought of as relating to the historical probability that a student selects that class, conditioned on them taking the originating class in the previous semester. However, given that empirical transition probability doesn't take into account grade performance, we develop a different weighting mechanism to take this into account.

B. Grade-Based Edge Weights

The equation for the weight of an edge in the subsequency graph begins with a simple equation representing the grade improvement experienced by students who transitioned from class i to class j. This score is an average difference between the grades of each student s who took both classes,

$$H_{ij} = \frac{\sum_{s=0}^{n_{ij}} (G_{sj} - G_{si})}{n_{ij}} \tag{1}$$

where G_{si} represents the grade of student s in class i, n_{ij} represents the number of students who transitioned from class i to class j, and H_{ij} is the improvement score. For a GPA range of $0 - 4$, this value ranges from -4 to 4, wherein the highest possible improvement is from a grade of 0.0 to 4.0, and a negative improvement indicates a grade drop between classes.

This metric, however, is not equally robust for every transition. While some classes were taken by hundreds of students who transitioned in and out quite often, many classes were much less frequented. This results in discrepancies in the robustness of each transition due to variable sample sizes, wherein some transitions occurred much more frequently than others. We therefore introduce two levels of Bayesian smoothing to this parameter. The first accounts for transition effects leaving a class, while the second accounts for transition effects entering a class. The first of these adjustments is governed by the equation

$$H'_{ij} = \frac{n_{ij}H_{ij} + n_i\bar{H}_i}{n_{ij} + n_i} \tag{2}$$

where n_i is the number of students who took class i and $\bar{H}_i$ is the average improvement score when leaving class i. The second adjustment is governed by a similar equation

$$H''_{ij} = \frac{n_{ij}H'_{ij} + n_j\bar{H}'_j}{n_{ij} + n_j} \tag{3}$$

where n_j is the number of students who took class j and $\bar{H}_j$ is the average improvement score when entering class j.

Finally, the algorithm converts this improvement score into an expected grade H^* by incorporating a user's input. This conversion introduces personalization for each user so that two users with different grades will not receive the same recommendations. The equation for the expected grade is

$$H^*_{ij} = \min(\max(G_i + H''_{ij}, 0), 4) \tag{4}$$

where G_i is the user's grade in class i. Because H'' ranges from -4 to 4, the expected grade potentially ranges from -4 to 8. Since grades outside the range of 0 to 4 are not possible, the expected grade is clipped to the allowable range. To those students, any improvement score above a certain range related to their current grade would be viewed equally, while H'' values below that range would also be viewed equally. We also note that since this statistic incorporates a students current classes and grades, it is only possible to calculate the expected grade for the subset of the graph that transitions out of their current schedule. This is precisely the subset of classes that the recommendation algorithm selects from.

C. Popularity-Based Edge Weights

The second factor that the graph's edge weight function takes into account is the overall proportion of students who made the transition between one class and its successor. Although a course consistently results in lower grades for students, it could still be popular for a variety of reasons. Whether it is part of a required course track or taught by a famous professor, there must be a reason if everyone takes a class despite expecting a bad grade. This proportionality metric is taken into account by the popularity score P, which is defined for the transition from class i to j as the ratio of the number of students who made that transition to the total number of students who took the outgoing class i. That is,

$$P_{ij} = \frac{n_{ij}}{n_i} \tag{5}$$

Because students averaged four classes in a semester and each of their previous classes was linked to all of their current courses, P ranged from 0 to roughly 0.25, the former end of the range indicating few transitions from one class as opposed to another, and the latter indicating close to 100% transition into a class.

D. Composite Recommendation Metric

Finally, the expected grade and popularity metrics are combined together into a single edge weight for use during recommendation. The goal in considering both P and H^* is to avoid hyper-fixation on grades and model more realistic considerations. First, both parameters are scaled to a range of 0 to 1 by multiplying H^* by a factor of $1/4$ and P by a factor of $1/P^*$, where P^* is equal to the inverse of the average semester size. We add parameterization to determine the importance of expected grade and class popularity when recommending courses by multiplying each metric by a new parameter (α and β, respectively) that can be determined experimentally based on a training dataset. The subsequency graph's final edge weights W_{ij} represent a personal utility of making the transition from class i to class j according to the user's expected grade and course popularity. The final equation for these weights is:

$$W_{ij} = \alpha\frac{H^*_{ij}}{4} + \beta\frac{P_{ij}}{P^*} \tag{6}$$

Using the subsequency graph as an underlying network, recommendation is as simple as choosing the edges with the highest weights among those available. For an input of a students current courses, the algorithm compiles a list of all the edges leaving the corresponding nodes in the subsequency graph, orders them by weight, and suggests the first 5 distinct classes that the edges point to.

III. Dataset

The algorithm introduced in the previous section was implemented using a dataset from Princeton University, such that a user inputs the number of classes they took in the most recent semester, the numerical ID of each class according to the dataset, and the corresponding grade they received. We utilized a dataset from Princeton University comprising transcript data of 6058 students who together took 1568 unique classes. The classes were renamed and semesters were combined for anonymity, and schedules were incomplete in some cases, which resulted in records of fewer classes listed for a student than normal.

Total Number of Courses Taken	Sem. 1	Sem. 2	Sem. 3	Sem. 4
1	1	x	x	x
2	2	x	x	x
3	3	x	x	x
4	4	x	x	x
5	5	x	x	x
6	3	3	x	x
7	3	4	x	x
8	4	4	x	x
9	4	5	x	x
10	5	5	x	x
11	4	4	3	x
12	4	4	4	x
13	4	4	5	x
14	5	5	4	x
15	4	4	4	3
16	4	4	4	4
17	4	4	4	5
18	5	5	4	4
19	5	5	5	4
20	5	5	5	5
21	5	5	5	6

Fig. 1. Semester splitting rules for a single student. Because the dataset removed semester information to preserve anonymity, this was the mechanism whereby we reconstructed the semester divisions from an ordered set of courses.

While the size of this dataset is substantial, the anonymity limitations required a set of assumptions to be made in order to sufficiently analyze the recommendation algorithm. The first of these assumptions was that the classes listed for a student ID maintained the order in which they were taken. This assumption allowed us to split the classes into multiple semesters based on the total number of classes listed for a student. Schedules ranged from 1 to 23 classes, and were separated following the scheme shown in Table 1, which lists the number of classes delegated to each semester given the number of classes in a schedule. Schedules of 1-5 classes were left as one semester, 6-10 were split into two semesters, 11-14 were split into three semesters, and 15 or more were split into four semesters. Finally, we assumed that any two classes could be taken together in a single semester, although in reality certain classes are restricted from being taken simultaneously due to overlapping content.

Various other limitations should be noted of the dataset. As mentioned above, students and courses were anonymized and potentially randomly ordered; this presented us with a fundamental limitation in our ability to track transitions between courses, given that semesters might not be in any meaningful order. Additionally, it should be noted that a number of both courses and students were removed from the dataset, resulting in a few gaps in both sequences.

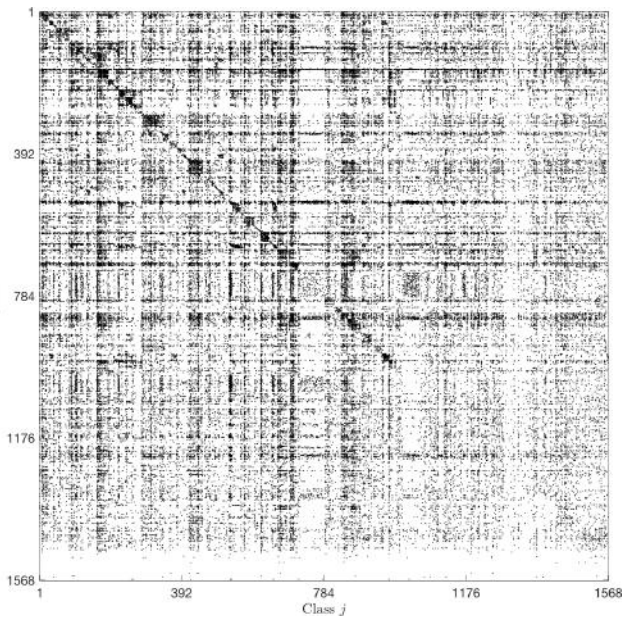


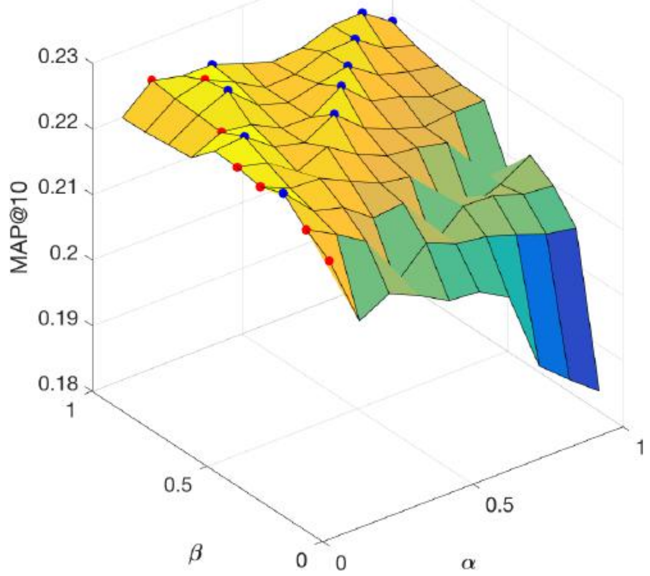
Fig. 2. Subsequency graph without edge weights. Note the strong density of points along specific columns, rows, and top left diagonal.

The unweighted subsequency graph for this dataset is visualized as a sparse matrix in Fig. 2. The strong density of points along a columns or rows of the graph represents a heavily populated course. The strong diagonal originating in the top left of the graph would seem to indicate heavy transitions from one course to itself, but in reality it simply reflects a common theme for a course to transition to a nearly consecutive course (such as C101 to C102). This diagonal suggests that the course IDs were not randomized, as subsequent courses are often labelled accordingly, and it supports the choice of semester delegation as we would expect to see this trend in actual course selection.

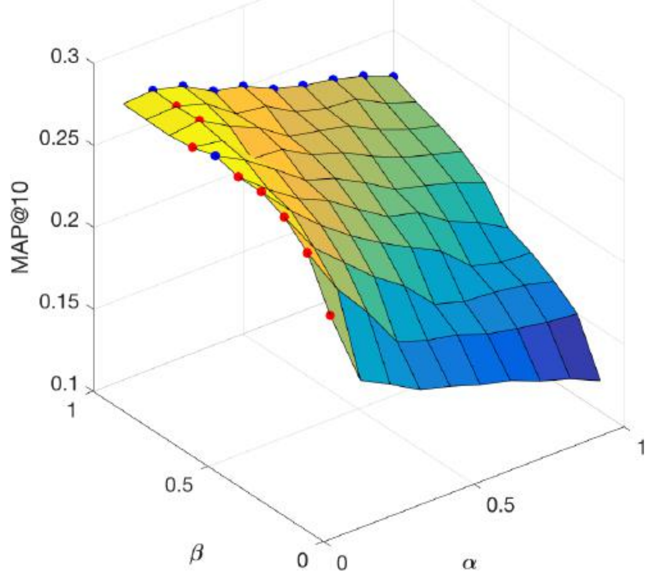
IV. Evaluation

A. Parameter Tuning

The first step in evaluating our algorithm lies in experimentally optimizing α and β in the edge weight equation (Equation 6). This is accomplished by separating our dataset into a training set to learn the entries of the H'' and P matrices, and two test sets of students against which to compare the relative performance of the recommendation. The first test set was extracted by sampling a random 10% of students with GPAs above the 90^{th} percentile (GPA ≥ 3.82), and the second comprised a random 10% sample of students with GPAs below the 10^{th} percentile (GPA ≤ 2.69). From these samples, students with only one recorded semester were removed, such that the final high GPA test set included $N_s = 45$ students and the final low GPA test set included $N_s = 52$ students. We hope that the recommender system models the behavior of students



(a) GPA ≥ 3.82



(b) GPA ≤ 2.69

Fig. 3. MAP@N values for various α and β for high (a) and low (b) GPA sets, with $N = 25$ class recommendations. Note smaller range for the high GPA test set. (a). Maximal precision values along α and β plotted in red and blue, respectively. Note overall trend upwards as α decreases and β increases, with an absolute maximum for $\alpha = 0.3$ and $\beta = 0.8$, and absolute minimum for $\alpha = 1.0$ and $\beta = 0.1$. (b). Minimal precision values along α and β plotted in red and blue respectively. Note similar trend to (a) with higher variation. The absolute maximum precision is achieved when $\alpha = 0.1$ and $\beta = 0.6$, and the minimum precision when $\alpha = 1.0$ and $\beta = 0.1$.

with high GPAs, while not necessarily modeling the behavior of students with low GPAs. As a result, we search for the choice of α and β such that our recommendation performance is maximized for the high GPA test set and minimized for the low GPA test set, indicating recommendations that would result in good grades for a user.

B. Evaluation Metric

We measure the performance of our recommendation algorithm using mean average precision for top-N recommendation (MAP@N) metric, which rewards algorithms that order the recommendations properly. Average precision for a set of N recommendations presented to student s is given by

$$AP_s@N = \sum_{n=1}^{N} \frac{P_s@n \cdot 1_{\text{s took class } c_s(n)}}{\min\{|R_s|, N\}}, \qquad (7)$$

where R_s denotes the set of classes that student s took during the subsequent semester, $c_s(n)$ denotes the n^{th} class recommended to the learner, and $P_s@n$ denotes the precision at n, i.e., the fraction of classes among the top n recommendations that the student actually took. Mean average precision (MAP@N) is the empirical mean of $AP_s@N$ for all students in the test set. We use $N = 10$ classes in our determination of α and β.

C. Results: Effect of Alpha and Beta

By sweeping both α and β over the range $[0.1, 1]$, we calculate the precision of the recommendations given by this algorithm for both the high and low GPA test sets. (Fig. 3). For context, a random recommendation system was also tested on the same training and test sets, wherein a random class out of those present in the training set was suggested with equal probability for all observed transitions from the input set of classes. With $N = 10$ classes, this random system resulted in a precision of MAP@N $= 0.0085$ for the high GPA test set and 0.0074 for the low GPA test set. The recommendation algorithm has a minimum precision of 0.1830 for the high GPA set and 0.1185 for the low GPA set, suggesting that even in the worst case the recommendations given by the algorithm are better than random. It also provides evidence that the algorithm does suggest useful courses to students, even when the input set of courses taken in a semester has not necessarily been observed before.

Within the high GPA test set (Fig. 3a), we see a rough curve for precision that decreases with α and increases with β. While we would expect that students with high GPAs would heavily consider their expected grades when choosing courses, this trend is not present in the precision of the algorithm's recommendations. This could be due to the fact that students don't necessarily have a good way to predict their own grades in a class and are therefore more likely to look to popular courses that their friends have taken in the past. It could also be that the main attraction of classes and course selection lies in course content and student interest, and that students somewhat ignore their potential grades in favor of taking a class they might find interesting. Interestingly, the overall maximum occurs when $\alpha = 0.3$ and $\beta = 0.8$, with MAP@N $= 0.2251$. At this point, the ratio between the two parameters is not maximized, suggesting that although popularity is the larger factor in course selection, expected grade is not ignored altogether.

We see a similar, but smoother trend in the low GPA test set (Fig. 3b), in which precision decreases with α and increases with β. The low GPA set had a larger range of precision and, interestingly, reached above the maximal precision of the high GPA test set. The maximum precision for this set (MAP@N $=$ 0.2732) occurs at $\alpha = 0.1$ and $\beta = 0.6$, which is close to the maximum precision point for the high GPAs.

Although both the high GPA set and the low GPA set favored lower values of α and higher values of β, the primary difference between the two is the shape of the variation. Whereas the low GPA set experienced a smooth decline in precision for increasing α, the high GPA experiences a plateau for pairs in which $\alpha < \beta$, followed by a sharp decrease for larger values of α. We can see that for both groups, expected grade is not a very strong predictor, and that popularity is a strong predictor for the low GPA set, but not necessarily so for the high GPA set. One possible explanation for why expected grade was not a strong predictor of courses is that students don't necessarily use expected grade in determining which courses to take, either because of major/general requirements or because they have a hard time predicting which courses will raise/lower their GPA. In the region where the popularity metric becomes the dominant factor, the discrepancy in performance between the high GPA and low GPA groups could shed light on behaviors of those two groups. Students with low GPAs may tend towards predictable courses and course trajectories followed by the majority of their peers, but students with high GPAs may tend towards more diverse courseloads.

The difference in the shapes of the two plots could also explain why the high GPA test set experienced a lower maximum precision overall. If these students are more likely to take a diverse courseload, then they would be more likely to choose transitions that have not been observed before within the dataset. While any course transition is theoretically possible, bar certain university specific restrictions such as prerequisites and major requirements, this algorithm is restricted to suggesting courses that have been taken subsequently in the past. The discrepancy in maximum precision and general decrease in precision with α could also indicate that GPA is less a product of course selection as it is of other factors. A student's grade may reflect their work ethic, course interest, and instructor preferences as much as course selection, and these factors may play a larger role for students with high GPAs than others. These factors are difficult to disentangle, and could cause disruptions in the expectations set for this algorithm with respect to GPA.

D. Results: Precision and GPA

Fig. 3 begins to show a surprising relationship between GPA and precision, in which the classes recommended to a student match those taken by students with overall lower GPAs more than those with higher GPAs. This relationship is quantified in Fig. 4. For these tests, 100 students were randomly sampled from the entire data set and removed for training. Once again, students with only one recorded semester were removed, leaving a sample size of $N_s = 74$. The plotted GPAs were calculated using the courses from the second semester onward, as recommendations could be made for these semesters but not for the first. In regression calculation, outliers outside 2 standard deviations around the mean were removed. None of the regressions show statistically significant correlation, so it is difficult to make any conclusions about the direction of the relationship between precision and GPA.

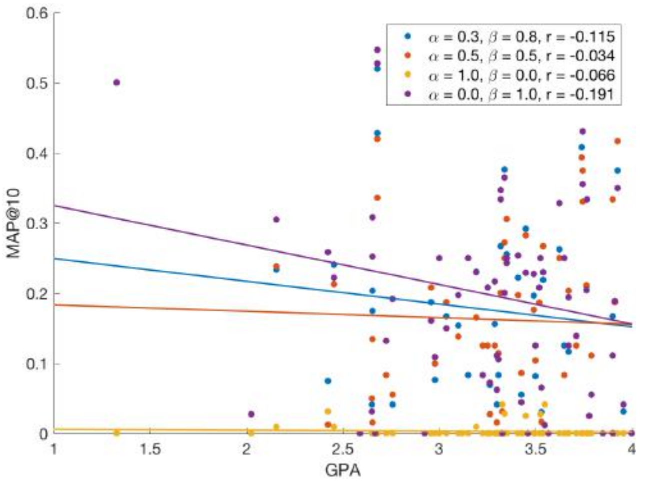


Fig. 4. Precision vs GPA for various α and β pairs, including optimized values $\alpha = 0.3$ and $\beta = 0.8$. Regression lines are plotted for each pair, with correlation coefficients r given. GPA is calculated using only courses from semesters in which the algorithm was able to make recommendations, meaning that first semester courses are excluded. Note general negative correlation except for $\alpha = \beta = 0.5$.

The general lack of correlation between precision and GPA may suggest that students who receive good grades do not choose classes differently to those with lower grades. It is likely that GPA is not determined only by course selection, and that it is rather a combination of specific students' work ethic and general abilities. While the effect of the ratio between α and β seen in Fig. 3 points to the inclusion of of expected grades in course selection to maintain precision, it is possible that the grade a student expects to receive does not necessarily match the grade they do receive. It could also be that students are a poor judge of grade improvement, as they are not privy to the information contained in the model presented here.

We also note that the precision values for the third set ($\alpha = 1.0, \beta = 0.0$) are almost zero, whereas the precision values for the fourth set ($\alpha = 0.0, \beta = 1.0$) are among the highest. This could suggest that expected grade is a much weaker factor than popularity when it comes to choosing classes, and it is perhaps the case that this represents the relative proportion of course slots that must be filled with required classes rather than a completely free choice.

The fact that no weight ratio is able to better predict classes taken by students with high GPAs than those with low GPAs also suggests that there may be a relationship between α and β that is not accounted for here. The popularity of a course, for example, could be related to the fact that many students expect to receive a good grade in the class, which would point to an interdependence of the two parameters that is not reflected by this model.

V. Conclusion and future work

With this course recommendation model and algorithm, we are not only able to explore trends among students, but suggest courses similar to those chosen by actual university students.

As it stands, we can use our algorithm to recommend a set of courses. However, this model does not take into account the information that could be gleaned from relationships between classes taken concurrently. A possible solution to this problem would be to create a concurrency graph, similar to the subsequency graph above, but with undirected links between classes that were taken simultaneously during one semester. We believe that the information encoded in this graph would hold tremendous insight. For instance, the graph would be able to provide information about two classes that were never taken in conjunction. With this information, we could then assume that these classes could not be taken concurrently, such as Princeton classes PHY 103 (a physics course on mechanics designed for engineers) and PHY 105 (a physics course on mechanics for physics majors), which cannot be taken together due to their overlapping content. Furthermore, we could also use tools like the graphical lasso to see the relationships between groups of factors within this concurrency graph and discover clusters of courses that work well together. It is also possible to calculate course and student biases and to utilize a latent factor model similar to those used in the Netflix recommendation system. These parameters have the potential to improve the quality of our recommended courses by creating a more accurate model of both student and course relationships.

While this model takes into account two key elements when determining which courses to recommend, it also overlooks several other key components. For example, this model has no consideration for intended majors and minors, nor does it explicitly consider prerequisite course relationships. General core requirements are also unaccounted for, and more subjective measures such as course reviews and general academic interest currently have no place in this model. Further practical issues are also ignored, including course scheduling and the semesters in which classes are offered. The limitations of our data prevented us from including the above features, but it is possible that our model could be updated to include them with a more complete dataset.

Finally, the underlying University data that is used to populate the subsequency graph is continually updated with each semester and each round of classes taken by Princeton students. The algorithm also requires the input of at least one semester of classes and grades from each user, which could easily be used to update the underlying networks. At some point, it would be prudent to implement a system that restructured the graphs along with these sources of new information so as to maintain the accuracy and quality of recommendations.

References

[1] Linden, G., Smith, B., and York, J., "Amazon. com recommendations: Item-to-item collaborative filtering," *IEEE Internet computing*, Vol. 7, No. 1, 2003, pp. 76–80.

[2] Burke, R., "Hybrid recommender systems: Survey and experiments," *User modeling and user-adapted interaction*, Vol. 12, No. 4, 2002, pp. 331–370.

[3] Brinton, C. G. and Chiang, M., "Social learning networks: A brief survey," *Information Sciences and Systems (CISS), 2014 48th Annual Conference on*, IEEE, 2014, pp. 1–6.

[4] Farzan, R. and Brusilovsky, P., "Social navigation support in a course recommendation system," *International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, Springer, 2006, pp. 91–100.

[5] Farzan, R. and Brusilovsky, P., "Encouraging user participation in a course recommender system: An impact on user behavior," *Computers in Human Behavior*, Vol. 27, No. 1, 2011, pp. 276–284.

[6] Chen, C.-M., Lee, H.-M., and Chen, Y.-H., "Personalized e-learning system using item response theory," *Computers & Education*, Vol. 44, No. 3, 2005, pp. 237–255.

[7] O'Mahony, M. P. and Smyth, B., "A recommender system for on-line course enrolment: an initial study," *Proceedings of the 2007 ACM conference on Recommender systems*, ACM, 2007, pp. 133–136.

[8] Sandvig, J. and Burke, R., "Aacorn: A CBR recommender for academic advising," *Technical Report, DePaul University*, 2005.

[9] Bendakir, N., "Using association rules for course recommendation," *Proceedings of the AAAI Workshop on Educational Data Mining*, Vol. 3, 2006.

[10] Lin, W., Alvarez, S. A., and Ruiz, C., "Efficient adaptive-support association rule mining for recommender systems," *Data mining and knowledge discovery*, Vol. 6, No. 1, 2002, pp. 83–105.