ABSTRACT

Each semester, Penn students choose a small subset of courses from the thousands offered by the university. The process of course selection is often ambiguous, fragmented, and difficult. Penn Course Recommender aggregates a wide range of relevant factors and generates a list of course recommendations for personalized for each student. The system processes recommendations in a robust manner with minimal user input, aiming for simplicity and user-friendliness.

This paper outlines the development of Penn Course Recommender whose three main components are as follows: the Requirements Graph, the Recommendation System, and the User Interface. A discussion of the resulting quality of recommendations and related challenges are also included, along with suggestions for the future of Penn Course Recommender.

1. INTRODUCTION

Ever semester, students at the University of Pennsylvania all face the same unclear, difficult question: “what courses should I take next semester?” A variety of factors must be carefully balanced. Major and/or minor requirements must be fulfilled, but each planned future course may have unfulfilled prerequisites or corequisites. Students may also wish to take into consideration recommendations from major advisors, peer student suggestions, ratings of courses and professors from Penn Course Review, and how closely courses line up with their personal interests. Considering that Penn offers hundreds of courses each and every semester, course selection is a frequent and recurring area of confusion for students.

In addition, Penn students attend the university for a relatively short period of time (on average eight semesters) while paying a considerable sum of tuition to do so. Course selection not only influences what a student learns while in the university, but also a student’s employment or graduate opportunities. Thus, it is extremely important for students to take the most informed approach as possible to their course selection. Poor course planning could potentially stop a student from graduating in four years and cost their course selection. Poor course planning could potentially stop a student from graduating in four years and cost them additional tuition fees or even potential employment. Thus, it is extremely important for students to take the most informed approach as possible to their course selection.

Penn Course Recommender helps alleviate a key pain point. Most importantly it streamlines and simplifies the process of course selection in order of prerequisites. For example, an engineering student with a French minor should not have to juggle with the intricacies of cross-school or cross-departmental dependencies or realize his last semester that he cannot finish his minor. Penn Course Recommender creates a long term prerequisite-to-course mappings with majors that allow the recommender to suggest course decision in a transparent manner. This allows the student to plan the fulfilling of a major or minor with Penn Course Recommender with maximal visibility of fulfilling prerequisites for future courses.

Furthermore, Penn Course Recommender leverages similar peer student users’ course experiences to assist in maximizing the quality of the recommendations. Not only does the recommender take into consideration the reviews and ratings from Penn Course Review to suggest the highest rated courses, but it also examines the existing network of other student users. This allows for recommendations to be made based on the courses taken by other users with similar interests using adsorption technique (to be explained more in-depth below). More generally, Penn Course Recommender proposes courses that other students sharing majors or similar personal interests had enjoyed that the student may not have discovered otherwise without asking other students.

Overall, Penn Course Recommender has the potential to alleviate much of the ambiguity and uncertainty surrounding course discovery and selection, and can equip the students with the right tools to make more knowledgeable decisions regarding their academic path in University.

Penn Course Recommender provides personalized and targeted recommendations for each student. The system accomplishes this by with a user interface where the end user enters a list of previous courses taken, if any, intended field(s) of study, and areas of interest. The system then returns a list of the top recommended courses for the student. There are three major components that make this possible: the Requirements Graph, the Recommendation System, and the User Interface.

Advisor: Zack Ives (zives@cis.upenn.edu).
1.1 Requirements Graph

The Requirements Graph, at its core, contains all of the prerequisites relationships between courses. It holds nodes of logical operators connected to nodes of classes that allow for quick evaluation with a simple traversal to see if prerequisites have been fulfilled (to be explained in more detail below). It takes in the set of courses that a student has taken, and it returns the set of feasible courses - those courses for which the student has fulfilled all necessary prerequisites and has not yet taken.

1.2 Recommendation System

The Recommender System translates the Requirements Graph and a student's background and interests into a list of recommended courses. It takes in the set of feasible courses and uses a simple Multi-Attribute Model to combine normalized scores representing variables including fulfilling major/minor requirements, the ratings from Penn Course Review, a student's personal interests, and what other similar students took. This creates a final normalized score for each course. Ultimately, the highest scoring courses are passed on to the User Interface.

1.3 User Interface

The User Interface is the interface by which end users interact with the product. It allows for an input of the student user's completed coursework up to the current semester, major(s), and interested field(s) of study. The interface then returns to the user a list of the highest scoring recommended courses with their scores. All in all, the User Interface provides a simple, easy-to-use interface for students to retrieve only the necessary information about the courses and get the best recommendations.

Recommendations are personalized, and as such the evaluation of the results are subjective. The system generates a list of recommended courses personalized to each user, and this makes the user the sole person fit to evaluate. For example, highly advanced students may prefer to finish all degree requirements with the toughest and most rewarding courses with little regard for electives, while struggling students would balk at such recommendations.

While manual sanity checking was consistently carried out against small sets of sample data to find any glaring errors, evaluation of such subjective tasks ideally require evaluation by users themselves. Getting a large number of responses from student users is difficult due to the little incentive for a student to respond. Thus, the set of available peers for generating peer scores is limited and the recommendations in extension potentially could be affected.

2. RELATED WORK

There are two main areas of research related to Penn Course Recommender. Firstly, much research has been conducted studying the value of course and teacher rating systems and their various shortcomings. Secondly, the methods used by companies such as Amazon and Netflix to recommend products and movies, respectively, are a subject of recent research. More specifically, the research focuses on analyzing the added "network value" created by such systems and looking into the techniques used to make intelligent, personalized recommendations.

Much research has been done on the effectiveness of course review systems, and if the results alone are a good basis for choosing classes. Felton, Mitchell and Stinson question the very nature of rating professors and courses, noting that there is a strong correlation between the attractiveness of a professor (a variable that is polled on ratemyprosessors.com), the 'easiness' of the course, and the course quality rating they receive. Their research suggests that over half of course reviews are positively skewed to favor better looking professors teaching easier courses (source 3, source 9).

Engdahl, Keating and Perrachione write on the effect of grade feedback on teacher evaluations, concluding that there is a correlation with a student's success in the class and the resulting teacher evaluations, given that the evaluations are completed after final grade feedback (source 2). Gallagher takes the research a step further and highlights the importance of teacher evaluations not merely in rating them, but informing and improving teaching methods (source 4). Terman and Rankin-Ullock write on the effect of the instructor's gender in skewing evaluations. Expecting to find that evaluations of faculty would reflect bias toward faculty of the reviewer's own sex, what they found was surprising: Male students typically give lower ratings than female students do, but their ratings for female faculty are high regardless of the field. Female students show a bias against women faculty in traditional fields and in favor of women faculty in nontraditional fields. (source 8)

These confounding factors all indicate that relying solely on course review sources to inform course recommendation decisions is insufficient. More importantly, little to no research on course review methods covers the effectiveness of using reviews in curriculum planning.

With regards to the science of making recommendations, it is important to remember that Penn Course Recommender will be a consumer-facing product. By associating courses not only with reviews but with people, a course becomes a networked good. There are therefore many parallels with Penn Course Recommender and any online service that suggests products or services. The way in which Penn Course Recommender recommends courses is analogous to what Amazon does with retail, Netflix does with media and Facebook does with people. Assuming that these people are fairly like-minded to the student, Penn Course Recommender can leverage social networks in order to make more advanced recommendations.

Much research has been done on the subject of leveraging social networks to identify nodes with common traits. Oestreicher-Singer, Libai, Sivan, Carmi and Yassin write on the “network value” of goods and services, a recent phenomenon that has emerged via the development of online shopping, media, and social networks. Traditionally, the value of a product is determined solely by the marginal utility generated by its consumption. However, according to their research, products also influence one another's sale. With the emergence of highly frequented e-commerce environments, recommending products to consumers has become less about traditional, utilitarian value, but about their "network value". We can draw direct parallels of this with our project - although a certain course may be the most informative, it may be useless if that particular student’s interest lies in a different field (ie: it has a lower network value). Furthermore, the authors describe the systematic approach taken to building this network, separating the product’s value into its own intrinsic value and its value with regard to the entire
network. They highlight how the value of low sellers may be underestimated, whereas the value of best sellers may be overestimated. It is precisely this problem that Penn Course Recommender hope to solve for course selection. By calculating not only the intrinsic value, but also the network value of each course, Penn Course Recommender hopes to create more accurate and personalized recommendations for each student (source 6).

Stephen and Toubia agree with this research, taking a more pointed analysis by looking at e-commerce “social networks.” Their research examines the economic value implications of connecting various sellers on one centralized platform. They find that allowing sellers to be connected through this network creates considerable economic value, primarily by making sellers more accessible. Most importantly, they realize that the sellers that benefit most from the network are not the most popular ones, but the ones that are typically least accessible. Once again, this analysis is analogous with what we aim to do. By building a social network out of the course review data we currently have, we can build a more accurate picture of how each course should be valued by each student. (source 7)

As to the approach taken to make recommendations, Adomavicius and Tuzhilin provide an overview of the different methods used to suggest goods to consumers. These methods are typically classified in three categories: content based, collaborative, and hybrid. Drawing the parallel, one may be recommended a course because it aligns with that student’s interest, because his friends took it, or both. The authors also tackles the various limitations and shortcomings of these recommendation systems, highlighting the importance of strong understanding of contextual information surrounding the goods as well as well balance multi-criteria ratings (source 1). Focusing on the hybrid approach, Koren writes on using collaborative filtering in order to provide users with personalized suggestions for products and services. Using past transaction data, the collaborative filtering method allows one to build a network between people and products. Koren writes on the usage of neighborhood models, which group users and products together based on similarities. He also considers a new model not only based on heuristic similarities, but also with a global cost minimizing function. Once again, many parallels between this methodology and Penn Course Recommender’s course recommendation method can be drawn (source 5).

In conclusion, there has been much research done in the field of analyzing the impact of teacher and course reviews. Based on the above sources, one can surmise that there is much value to be gained, both by the student and the institution, with a transparent and well implemented teacher review system. However, it is frequently suggested that such systems are inherently biased and flawed, and cannot be solely relied on. Furthermore, a general and computable approach to course recommendation in the context of one’s entire curriculum has yet to have been founded. By considering courses in the context of the student’s degree goals and by complementing the intrinsic value of courses (their rating) with a network value, Penn Course Review is able to more effectively highlight undervalued courses or downstage overvalued courses. Furthermore, by applying collaborative filtering methods, targeted recommendations based on the student’s peer network can be made.

3. SYSTEM MODEL
For a block diagram of the System Model of Penn Course Recommender, see Appendix B.

3.1 Requirements Graph
The foundation of Penn Course Recommender is the Requirements Graph, a robust relational web of courses with prerequisites and corequisites to iterate over. The nodes of the Requirements Graph are populated with course information mined from the APIs of the Penn Course Registrar and Penn Course Review. For each course, the following information is stored in the Requirements Graph: prerequisites, ratings, description, the course(s) for which it is a prerequisite.

Iterating over this data structure yields useful information for course recommendation. The bidirectional prerequisite relationships allow the recommendation system to verify whether or not a course satisfies a prerequisite requirement for another course. Cross checking the set of courses the student has taken against the set of all courses yields a final set of feasible courses that the student has fulfilled the prerequisite requirements for.

3.2 Recommendation System
The Recommendation System is the engine that drives Penn Course Recommender. It outputs a list of recommended courses sorted in order of most to least recommended. The system uses a multi-attribute model with the following variables: Interests Score, Descendants Score, Peer Score, and Penn Course Review Score. Each variable is individually calculated and then normalized to a range of $[1, 2]$, and then all of the scores are multiplied together to produce the final recommendation score.

The individual scores are normalized within the range $[1, 2]$ because using a range $[0, 1]$ would potentially allow for individual scores to result in a zero; presence of even a single zero would render all other individual scores meaningless and produce a final score of zero due to using multiplication of the scores, and could potentially eliminate viable courses for recommendation from being recommended at all.

3.2.1 Interests Score
One of the inputs that a student is asked for is their academic interests, which they enter in a free-form text box. The logic behind including this is that students often have academic interests beyond just what is within their major requirements. These interests are cross-referenced with the official course descriptions that are taken from the Penn Registrar and stored in the Requirements Graph. Given a student’s input, each course is assigned an Interests Score based on how well the course matches with the student interest. For example, consider a student majoring in CIS who is highly interested in paleontology. The student would indicate “paleontology” as one of her interests, and the system would give a higher score to courses with “paleontology” in the course description. This happens for every word in the student’s input for interests. Overall, the courses with the descriptions that best match the student’s interests are assigned the highest score for the Interests Score section.

3.2.2 Descendant Score
In this context, let us define a descendant of a given course as any course for which that given course is part
of the prerequisite “chain.” As an example, consider that MATH104 is a prerequisite for MATH114, which is a prerequisite for MATH241. Descendants of MATH104 are MATH114, MATH240, and MATH241. A predecessor is essentially the opposite: for a given course, any course that is part of the prerequisite chain leading up to that given course is a predecessor. In this example, predecessors of MATH241 are MATH104, MATH114, and MATH240.

The Descendant Score effectively represents how important a course is to fulfilling a major. This is represented by the proportion of courses within the student’s major for which a given course is a predecessor. The Descendant Score helps to ensure that students are recommended courses relevant and essential for their studies and actually work on completing their major. This score is included because students often consider completion of prerequisite chains in their course planning.

As an example, CIS majors at Penn are required to take CIS120 before moving on to higher-level CIS courses. CIS120 has 24 descendants, many of which are requirements for the major. Accordingly, CIS120 gets a very high Descendant Score.

Conversely, consider a course such as Writing Seminar. While all Penn students are required to take this, writing seminar is not a prerequisite for any other course within the CIS major. Thus, Writing Seminar is assigned a very low Descendant Score.

Calculating the Descendant Scores for each student’s feasible courses is done using Major Requirements Tree. The Major Requirements Tree is a data structure constructed with internal nodes containing logical operators (Boolean operators “AND”, “OR” and “[X] OF”) and courses at the leaf nodes, allowing for complex representation of the different varying major requirements from section to section. For a visualization of a subsection of the Networked and Social Systems (NETS) Major Requirements Tree, see Appendix C.

The Major Requirements Tree calculates Descendant Scores based on the proportion of each requirement for which a given course is a predecessor. Consider CIS120. It is a predecessor for CIS121 (which is also a requirement for CIS majors), so when calculating the Descendant Score for CIS120, CIS121 contributes +1 point to the score. Now consider the CIS Depth Elective requirement, which requires CIS students to take one of seven courses. In this case, four of these courses are descendants of CIS120, so the CIS Depth Elective requirement contributes +4/7 points to CIS120’s Descendant Score.

In addition to “OR” requirements such as the one just described, the Major Requirements Tree allows for “X OF” (i.e., “2 OF”, “3 OF”, etc.) and “AND” logical operators. For “X OF” requirements, scores are calculated exactly the same way as just described and then multiplied by X. For “AND” requirements, the score is not divided by the total number of requirements in the set, so each descendant of a course contributes a full point to its score.

Additionally, the Major Requirements Tree allows for Descendant Scores to vary dynamically from student to student depending on the courses they have taken and the major requirements they have fulfilled. For example, suppose there are three CIS major students, Alice, Bob, and Cindy. Alice has fulfilled her requirement of taking two Engineering Tech Electives, Bob has taken one Engineering Tech Elective, and Cindy has not taken any. For Alice, the Engineering Tech Elective requirement is ignored entirely when calculating the Descendant Score. For Bob, the Engineering Tech Elective is treated as if only one course is required to be taken from the group, in the same manner as described above. For Cindy, the score for each course is doubled as compared to if only one course was required to be taken from the category.

Ultimately, Descendant Scores are calculated for every course that fulfills any requirement within the student’s major.

### 3.2.3 Peer Score

This score derives from a very common way that students select courses: asking their peers. In particular, students tend to ask similar students such as students with similar previous coursework for recommendations. The Peer Score is generated using a technique called Adsorption. This implementation of adsorption is run on a bipartite graph, specifically students and courses, with connections between each student and each of the courses that she has taken. The intuition behind adsorption is that, given a starting node, in this case student, what is the probability that they will land on a given node, in this case a course, after a random walk? The probability is then normalized in the range [1, 2].

As an example, imagine a student who has taken several computer science courses but has not taken Philosophy 101, along with a large cohort of students who have taken several of the same computer science classes as well as Philosophy 101. That student will have a high Peer Score for Philosophy 101.

### 3.2.4 Penn Course Review Score

At Penn, one of the most commonly used tools to aid in course selection is Penn Course Review, which has accumulated ratings of various categories from course quality, professor quality, workload, difficulty, etc. over many years. Typically, students desire highly rated courses and professors while being wary of courses rated high in workload or difficulty. The Penn Course Review Score takes this into consideration.

### 3.3 User Interface

Users interact with Penn Course Recommender via a web application. It is comprised of two main elements: profile builder and recommender.

#### 3.3.1 Profile Builder

The user’s first interaction with Penn Course Recommender is building his or her student profile. In the profile building process, the user will create her Profile by providing the following information:

1. Previous courses taken
2. Major(s) and Minor(s)
3. Academic interests

This information is then fed into the requirements graph and the recommender in order to produce course recommendations.

#### 3.3.2 Recommender

The recommender then displays a top set of recommended courses for the student in order of recommendation strength.
In addition to course name, statistics on the strength of the recommendation (and a breakdown of its components: interest score, peer score, etc) are displayed. Penn Course Review ratings, Professor name, and times offered are also all displayed.

4. SYSTEM IMPLEMENTATION

4.1 Requirements Graph

Course information is retrieved programmatically through the Penn Registrar and Penn Course Review APIs. While the Penn Registrar API only offers data on courses to be offered in an upcoming semester, the Penn Course Review API offers a wealth of this course information dating back over a decade, providing historical course information as well as how students rated each of the courses. Cronjobs running Python and bash scripts allow for automatic and periodical querying of the APIs for data, and the information then can be processed into JSON files that can then be loaded and cached with the said Python scripts. The list of courses offered in the upcoming semester retrieved from the Registrar API is labeled as such so that the recommender does not recommend courses that are not being offered.

The key data points are keyed to the course code that maps to a map of key:value paired information the rest of the relevant information: course name, department id, prerequisites, corequisites, section times, professor, Penn Course Review course ratings (course and professor quality, work load, difficulty, etc) and course description. Once the relevant data has been collected through the API calls, all data is thoroughly checked for error or outdated information.

For example, CSE and TCOM courses, alongside many others, are no longer offered at Penn. While Penn Course Review still retains such courses for historical purposes, it makes little sense to use the outdated historical information that is no longer relevant in Penn Course Recommender and such course information is discarded.

The course rating data from Penn Course Review show a clear relationship between instructors, courses, and the classes that are currently offered. These course ratings and information are incorporated directly into the Recommendation System.

In order to read in the prerequisites of a course, further parsing methods are used. This is because the prerequisites and corequisites are almost always presented in highly varying human-readable formats with no form of enforced consistency that makes translation to programmatically usable representation difficult. For example, commas are often used liberally to represent “and” or “or” while there are vague, unclear phrases stated as prerequisites such as “one semester of programming” or “one computer language” listed. The vague phrases unfortunately are often to interpretation of the department the course belongs to, and thus have to be ignored by the recommendation system. Regular expression checking is used to check for presence of usable course ids in the prerequisite statement per course, and then python library PyParsing is used to parse out the ids in the string into a list structure containing boolean operators and the course ids, which then can be iterated through to construct the course to prerequisite relationship in the prerequisites requirements graph. Constructing the relationship for each course as to what higher level courses each course is a prerequisite for requires simple traversal of the graph.

In order to model these often complicated prerequisite relationships, a requirement tree data structure is used. Each course is associated with a requirement tree with logical operators at the nodes and prerequisite courses at the leaves. By iterating through this data structure, the Requirements Graph can easily evaluate whether a student’s set of completed courses fulfills the prerequisite requirements for each course.

4.2 Recommendation System

4.2.1 Interest Score

Interest Scores are calculated using a technique called tf-idf (text frequency-inverse document frequency). Tf-idf reflects how important a word is to a document in a collection. In this case, the “word” is each of the student’s interests, and the documents are the descriptions of each of the courses. The tf-idf score is proportional to the number of times that the word appears in the document and inversely proportional to the number of times that it appears across all documents. In practice, this means that a very specific interest, such as “palaeontology”, will give a large boost to courses that have that word in its description. On the other hand, if a student includes a very common word in her interests, such as “science”, then courses with that word will only get a very small boost.

Tf-idf is calculated using a user’s stated interests, their feasible courses, and the descriptions of the feasible courses. The first step is to generate a map of feasible courses to their descriptions, which are stored as an array as the original text has been split on whitespace. After this is completed, the system iterates over the courses and looks at each word. If a word in the description matches a stated interest a mapping from the interest to the course and occurrence count is either made or updated. After this is complete, the system iterates over the interests and computes the score for each associated course using the following formula:

$$0.5 + (0.5 \times \frac{\text{freq of interest}}{\text{max freq of interest}}) \times \log \left( \frac{\text{num courses}}{\text{num courses containing interest}} \right)$$

The mapping is then reversed and the score for each class is calculated by adding up all of the associated interest scores. This is then normalized in the range $[1, 2]$.

4.2.2 Descendant Score

As explained in the System Model section, the Major Requirements Tree is used to calculate the Descendant Score for each course in a student’s major. It is implemented recursively with an “AND” node at the root and each of the requirements as its children. For a visualization of a sub-section of the Networked and Social Systems (NETS) Major Requirements Tree, see Appendix C.

Calculating Descendant Scores is an $O(n^2)$ problem. It is done by traversing the tree using Depth First Search (which takes time $O(V + E) = O(n)$ because it is a tree) for each course (of which there are $O(n)$, giving us the $O(n^2)$ runtime). When a node (i.e. single course) is reached, it returns a score of 1 if the course is a descendant of the course being examined and a 0 otherwise. When a parent node (i.e. a logical operator) receives all of its scores from its children, it sums them up and, depending on the type of node:

- “OR” nodes: divides by its number of children and returns that value to its parent

- “AND” nodes: divides by its number of children and returns that value to its parent
at and, for every label at that node, the label is propagated
given weight 1. During the iterator step, each node is looked
initialization a unique label is generated for each user and
This is then used to start the iterative MapReduce job. On
that encompases the adjacency list of the user/course graph.
The first step is to generate a MapReduce readable text file
ition. This implementation of Adsorption uses an averaging
Descendant Score is being calculated.
contain a course that is a descendant of the course whose
Similarly to before, this allows for quick
the courses that are part of each of its unsatisfied children.
This works by looking at each node and recursively adding
the courses that are part of each of its unsatisfied children.
Similarly to before, this allows for quick O(1) lookup as to
whether or not a subtree contains a course. This means
that rather than having to scan each and every node within
a tree to try to find if it contains a course’s descendants,
the algorithm can easily skip over the subtrees that do not
contain a course that is a descendant of the course whose
Descendant Score is being calculated.

4.2.3 Peer Score
Peer Scores are generated using a technique called Adsorption.
This implementation of Adsorption uses an averaging
algorithm that is lends itself to the MapReduce framework.
The first step is to generate a MapReduce readable text file
that encompasses the adjacency list of the user/course graph.
This is then used to start the iterative MapReduce job. On
initialization a unique label is generated for each user and
given weight 1. During the iterator step, each node is looked
at and, for every label at that node, the label is propagated
along the edges with weight \( \frac{\text{number of edges}}{\text{number of edges}} \). The iterator runs
twice and then compares the max change for each group of
labels. Adsorption keeps iterating until the max change is
less than the threshold. When the max change for all labels
is less than the threshold, adsorption halts and outputs a
text file that maps a user to a list of classes with associated
probabilities of the user getting to that class. 1 is added to
this probability to give it the desired weight.

4.2.4 Penn Course Review Score
Penn Course Review has an API that grants access to all
of its aggregated course ratings data. This data is already
associated with each course in the Requirements Graph. The
variables we look at are Course Quality, Instructor Quality,
Difficulty, and Amount of Work. Because students typically
want high-quality courses that are not too difficult, these
ratings are combined using the formula:

\[
\text{PCR Score} = \frac{\text{Course Quality} \times \text{Instructor Quality}}{\text{Difficulty} \times \text{Amount of Work}}
\]

For courses that have been previously offered and thus
have ratings available, these scores are normalized over the
range of \([1, 2]\). If no ratings are available for a course, then it
is assigned the mean of the scores of all other courses’ Penn
Course Review Scores.

Finally, the system will multiply the four scores together
to arrive at a final score in the range \([1, 16]\), ranking the
scores in order of weight. The system then passes these
recommendations along to the User Interface.

4.3 User Interface
The User Interface is a simple web application built using
Apache Tomcat and Java on the backend and Java Servlet
Pages (JSP) with Bootstrap framework on the frontend. Us-
ing Java as the server side language allows for simple, easy
communication between the web server and the Recommenda-
tion System, as the potentially complex systems of the
recommender such as adsorption is built with Java.
Simple bootstrap template based pages are served with
limited allowed user inputs: student user’s completed course-
work up to the current semester, major(s), and interested
field(s) of study. The system returns a simple list of sug-
gested courses ordered by recommendation priority which
then can be displayed to the user.

5. RESULTS
Given the nature of course recommendation, the evalua-
tion of results are subjective. The recommender generates a
personalized set of classes for each user based on the inputs
and only the user has the ability to say whether or not the
recommendation was “good”.
Some students may only care for getting the easiest sched-
ule with the smallest workload possible regardless of course
quality, while students in dual degree programs, for example,
may prioritize courses that satisfy more than one require-
ment above all other factors. Thus, it is next to impossible
to provide a “perfect” recommendation satisfying everyone.
Manual sanity checking was done throughout the devel-
opment process as a way to ensure that the various parts
of the recommender were working properly. Courses were
checked against assumptions that were based on the course
selection experience of the authors, who have collectively
selected courses for 32 semesters. This was done to ensure
quality of recommendations and that courses that clearly
should not be recommended were not.
We utilize sanity checking in the evaluation of Descendant
Scores. By viewing the ranked scores of each courses for each
major, we can verify that the top-ranked courses make sense
for the major.
Below is a table of the top Descendant Scores amongst
the feasible courses for two scenarios for CIS majors: a first
6. FUTURE WORK

There are four additions that could be useful to the further development of Penn Course Recommender. First, would be an easy way for users to import all of the classes they have taken at Penn. While it may be simple for someone who has only taken a few courses at Penn, it is cumbersome for others. This issue raises the barrier to entry and could turn away students who would otherwise benefit from Penn Course Recommender. The addition of a simple way for users to import their courses would also be beneficial to the quality of results. As adsorption relies on the underlying bipartite user class graph, more data yields a strong result with more similar classes being recommended as well as a wider variety.

The second addition is a richer web application. The ability to search for classes or specific students would create a richer user experience. It would give students a way to help themselves, whether they are looking to talk to a person who has already taken a class they are interested in or to see what a complete degree looks like for a person who has completed a degree in an area they are considering. It would also be beneficial to have links as students may not necessarily have other methods of contacting users of Penn Course Recommender. Any use of a user’s name or email should be strictly opt-in.

The third addition is an improved feedback loop. Based on the time frame it was not useful to create a post-recommendation feedback loop. If Penn Course Recommender was up and running over the course of multiple semesters, giving users a way to give feedback on suggested courses would be very useful for refining the recommender. Feedback such as “I loved this course” could help that course be recommended more often and feedback such as “I hated this course” could help an elective’s score decrease so it is recommended less often.

Finally, the last addition that could be made is the addition of majors outside of NETS, CIS, and BE. This would increase the user pool for Penn Course Recommender which would be improve the quality of recommendations but would also increase the number of students Penn Course Recommender is able to help.

7. ETHICS

Ethics are not a significant concern for Penn Course Recommender. As it currently functions, Penn Course Recommender does not deal with sensitive information, it does not ask for users’ grades or other information such as their name or email. As Penn Course Recommender provides no way to login with a password there is also no worry that the passwords will be leaked.

The only part of Penn Course Recommender that needs to be dealt with carefully are the courses that are recommended to a user. It could be a worry that if a user is given a bad set of courses, enrolls in the courses, and then finds that they are no longer on track to graduate or some other equally bad consequence. This issue is mitigated by the fact that Penn Course Recommender is not meant to be the end-all-be-all of course recommendation; it is merely one of many factors that a student can consider. For example, in order to enroll in classes, students are required to get signoff from their advisor or some other staff member in the department who should be able to catch to issue before the user enrolls.

<table>
<thead>
<tr>
<th>Rank</th>
<th>First Semester</th>
<th>Second Semester</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MATH104 (2.00)</td>
<td>CIS120 (2.00)</td>
</tr>
<tr>
<td>2</td>
<td>CIS120 (1.85)</td>
<td>BIOL121 (1.78)</td>
</tr>
<tr>
<td>3</td>
<td>CIS160 (1.83)</td>
<td>CIS240 (1.73)</td>
</tr>
<tr>
<td>4</td>
<td>CIS110 (1.80)</td>
<td>BIOL101 (1.72)</td>
</tr>
<tr>
<td>5</td>
<td>PHYS150 (1.78)</td>
<td>MATH114 (1.71)</td>
</tr>
</tbody>
</table>

While Descendants Score is just one of the four scores that contribute to the ultimate recommendations, it is very important for indicating which courses are important to a major. For the first semester top courses, they are all spot-on; those are exactly the kinds of courses that a first semester freshman would take. For the second semester top courses, it is surprising to see two Biology courses with such high scores. Even with this slight shortcoming, the other three courses again are exactly the courses that a second semester CIS major should take in order to fulfill necessary prerequisite chains for higher-level CIS courses.

Getting data from a large set of students proved to be very difficult and the data collection surveys were designed accordingly. Students were asked to list the courses they took their first and second semesters and how satisfied they were with the respective schedules on a scale of 1 - 5 where 5 is most satisfied and 1 is least satisfied. There is little incentive for students to take the time to provide their past coursework and evaluate the recommendation. As such, there were only 19 good responses. This affected not only the evaluation of the results, but the quality of the recommendations as well. Given that a large part of the recommender, the peer scores, relies on rich user data, the peer score results were not as rich as they would have been given a larger dataset.

Using the survey results, the recommended generated second semester schedules for all of the good responses. For every response, the generated schedule was compared to the actual schedule and was given a similarity score based on the number of courses in common of the form:

$$\sum \frac{\text{number of courses respondent took recommended course}}{\text{number of courses respondent took}}$$

The average similarity score is .500 with minimum 1/2 and maximum 5/5. The similarity score was then multiplied with the stated satisfaction score to get the quality score. The average quality is 1.708 with minimum .500 and maximum 4.5. Schedules were then generated for third semester classes but the few responses that had associated emails either had not completed their third semester of classes and could not rate the classes or did not give feedback.

One of the issues with gathering results for a course recommender is the long feedback cycle. Courses for the following semester are selected in the middle of the current semester. This mean that within the scope of the project it was not possible to generate a recommendation and get feedback after a student took the recommended courses.

It is difficult to compare Penn Course Recommender to previous work. It is a novel system making recommendations for a subject matter that has not been addressed from an academic perspective before. Therefore, we are unable to compare our results to those of any related previous work.
8. CONCLUSION

Penn Course Recommender provides an easy to use way for students to get course recommendations for the coming semester. It does this by using a recommender that considers how interested a student may be in the course, how helpful the course will be in completing their degree, whether or not similar students have taken the course, and aggregated reviews of the quality of the course. Students are able to access their recommended courses through an easy-to-use web application.

APPENDIX

A. REFERENCES


B. BLOCK DIAGRAM

Figure 1: Penn Course Recommender Overview
C. MAJOR REQUIREMENTS TREE

Figure 2: Subsection of the NETS Major Requirements Tree