Technical Report CS17-09

The Course Map

Dillan Smith

Submitted to the Faculty of
The Department of Computer Science

Project Director: Dr. Bonham-Carter
Second Reader: Professor Wenskovitch

Allegheny College
2017

I hereby recognize and pledge to fulfill my responsibilities as defined in the Honor Code, and to maintain the integrity of both myself and the college community as a whole.

Dillan Smith

DILLAN SMITH. The Course Map.
(Under the direction of Dr. Bonham-Carter.)

ABSTRACT

All college students have experienced the struggle of class registration. This is a time consuming and difficult process only made worse by the inefficiency of giant course catalogs. This work consists of a tool that can take a course catalog, mine it for necessary information, and then visualize that information in a compact and interactive interface. This visualization shows prerequisite relationships between classes and other information through color, arrows, and text. This tool concisely presents the information found in the course catalog for both students and faculty. 36 students, 9 from each class year were surveyed on their experience with the Course Map system v. the course catalog. Our results show that students have a much better overall experience with the Course Map compared to the course catalog. This shows that use of a similar system may result in a more efficient and less taxing registration process as well as an overall college experience for students.

Contents

List of Figures

Chapter 1

Introduction

It is common knowledge that the United States is not the foremost country when it comes to education. According to Pearson, the U.S. is ranked 14th out of 40 in ‘cognitive skills and educational attainment’ [2]. Even our higher education systems have glaring faults. Each year in college much time and money is wasted when students take classes they didn’t need or they did not find fulfilling. This among other mistakes lead to graduation rates as well as retention rates being not nearly as high as one might expect [1]. That is, how to effectively inform students about course structure and aid them in efficient exploration. It does this by , first, text mining information in the course catalog. It then takes that information and graphs it an interactive web application. This chapter begins by providing a motivation for the system presented, it briefly discusses current attempts to answer this problem, and closes with the goals of the work.

1.1 Motivation

Imagine for a moment that you are a college first year and first timer. You are entirely ignorant to the process and challenges that await you over the next couple years. Your first task is to sign up for your first semesters classes. Generally, this requires you to peruse the course catalog for information. For the sake of the example we will assume that you are undecided about your major. The average college has hundreds of courses or even hundreds of tracks or majors of study. Suddenly your task is much more daunting than you anticipated. Not knowing what exactly you want you jump to the section of courses your gut –or maybe that's just your parents– says you want. That is, the 'hard' sciences such as Biology, Chemistry, and even Math. Or maybe your parents were lawyers so you default to courses that would prepare you for that line of study. Regardless of what area, the situation our imaginary student has found themselves in is not nearly as harmless as it may sound. The odds of our student graduating at all may be as low as one in two.

The first couple semesters of a student's college career are the foundation that the rest of their studies –and even their life– rest on. This is a time of great potential, but it can also be immensely unforgiving of mistakes. It is a time of exploration but also a time of preparation. This is when core classes and valuable prerequisites must be fulfilled so as to best prepare one for graduation. It is also the only time that an undecided student has to find and declare what they wish to study. A single

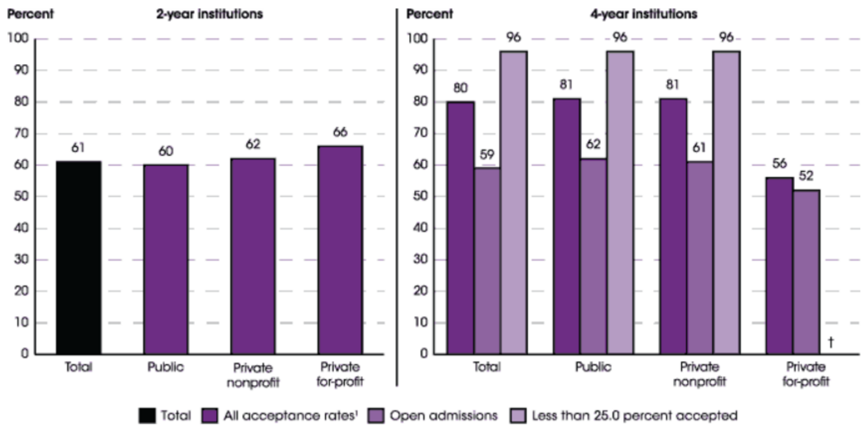


Figure 1.1: Retention Rates, NCES [1]

unnecessary class can be a massive mistake for a student with a packed schedule. A single unneeded/unwanted class at Allegheny for a student on a standard course load translates to ∼$5684 wasted away. This is found by dividing tuition per semester found at `http://sites.allegheny.edu/finserv/feescharges/` by the four classes contained in that semester. These mistakes, and many other factors, lead to only 60% of all first time undergraduate students graduating within six years [1].

Figure 1.1 depicts retention rates defined as the portion of students present in 2013 who returned the following fall of 2014. On the left are the retention rates of 2 year institutions and on the right are 4 year institutions. Rates drop as low as 52% for open admission private for-profit institutions. Keep in mind that this is only the number of students who returned and not the amount that actually completed their degrees. As we can see, a college education is not for everyone.

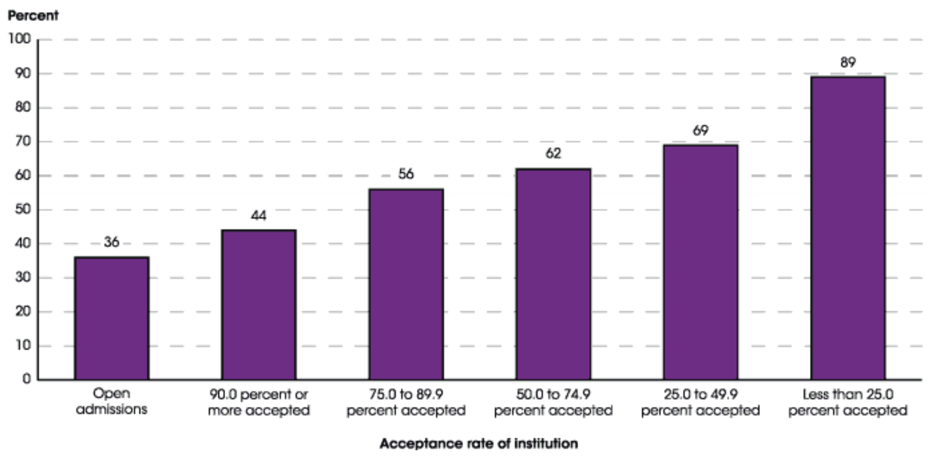


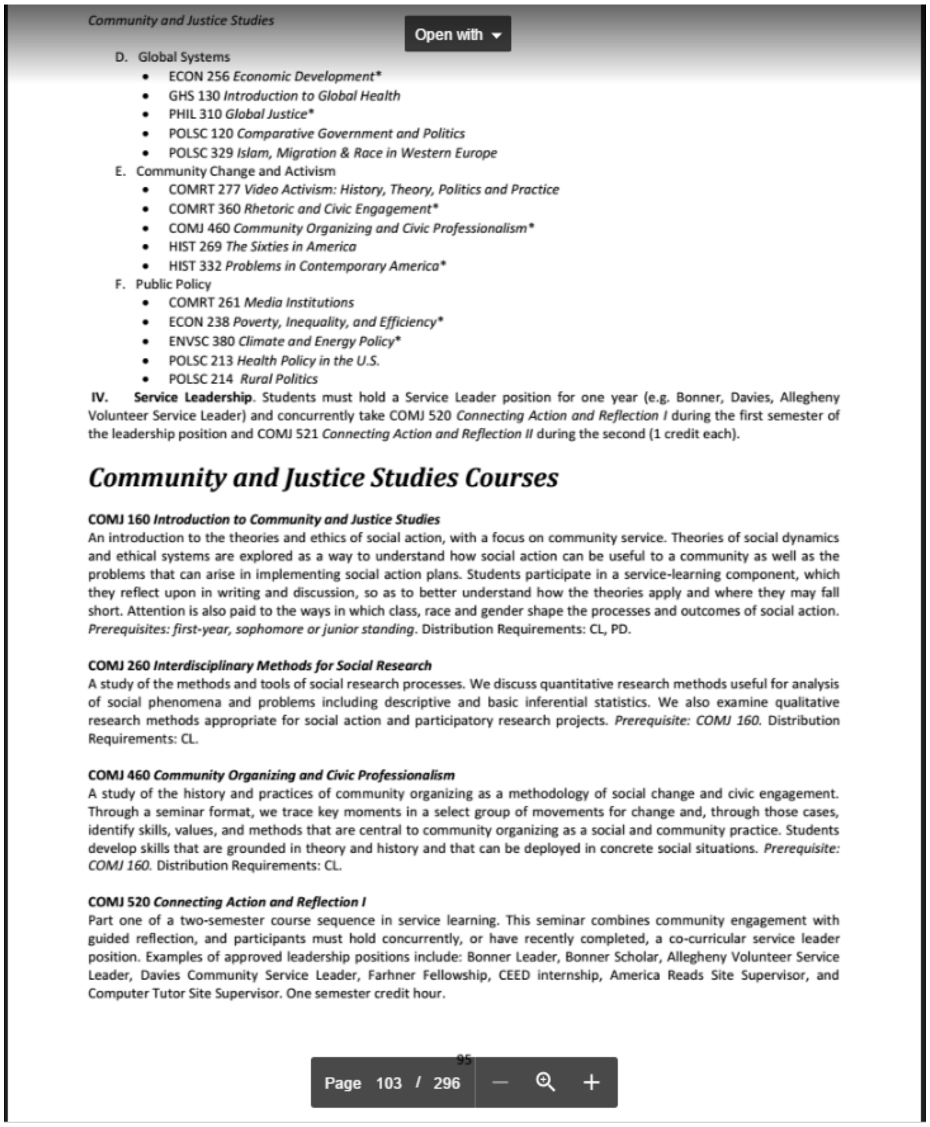
Figure 1.2: Graduation Rates Based on Acceptance Rates, NCES

In Figure 1.2 graduation rates based on institution acceptance rate are displayed. As is shown these rates reach as low as 36%. But surely more than 60%, and especially 36%, of our students are capable of graduating. Part of this may be attributed to the fact that it is estimated that up to 50% of all first time undergrad students enter school undecided on their major [8]. In addition, 70% of all undergrad students will change their major prior to graduation [8]. This implies that many college students do not yet know what they want. As a consequence of this, exploration is a necessity. But, exploration takes time. Something that many students are in short supply of.

Four years at a university sounds like a large amount of time at first, but requirements for any given track take up much of that time. A student at Allegheny College of the graduating class of 2017 with an average course load every semester will take 128 credits over the course of their career. At least half of these credits –and the time it takes– will be devoted to their major. In addition to this is, students take 20 credits for a minor and 8 for freshmen seminars. The challenges a modern day college student must face are numerous and often unknown to them. These include a large number of graduation requirements, a system that hinders exploration, a one in two chance of graduating at all, and many others.

Now, lets return to imagining an undergraduate first year. Unbeknownst to our student, there are many challenges ahead, and statistically the student may not graduate at all. If they do at all they may have crippling student debt to look forward to. The most important thing for them to begin and go on to join the 60% of graduating students is information. As the old adage says, knowledge is power. Our student needs to understand their schools course layout and infrastructure. He needs to know how to overload courses if needed and who to talk to to do so. He needs to know where to find financial aid. He needs to adjust to living away from home while also adjusting to a more academic life. The list of things the undergraduate student needs to know to be successful goes on and on. Luckily, most undergrad students are given an adviser who knows much of this. So, the most important thing is course information. Students need to know what courses they want to take so they go to their school's course catalog and are confronted with Figure 1.3.

Admittedly, the catalog does a very good job of containing a large amount of information. Just by looking at this page various pieces of information about Community



Community and Justice Studies

D. Global Systems
- ECON 256 *Economic Development**
- GHS 130 *Introduction to Global Health*
- PHIL 310 *Global Justice**
- POLSC 120 *Comparative Government and Politics*
- POLSC 329 *Islam, Migration & Race in Western Europe*

E. Community Change and Activism
- COMRT 277 *Video Activism: History, Theory, Politics and Practice*
- COMRT 360 *Rhetoric and Civic Engagement**
- COMJ 460 *Community Organizing and Civic Professionalism**
- HIST 269 *The Sixties in America*
- HIST 332 *Problems in Contemporary America**

F. Public Policy
- COMRT 261 *Media Institutions*
- ECON 238 *Poverty, Inequality, and Efficiency**
- ENVSC 380 *Climate and Energy Policy**
- POLSC 213 *Health Policy in the U.S.*
- POLSC 214 *Rural Politics*

IV. **Service Leadership**. Students must hold a Service Leader position for one year (e.g. Bonner, Davies, Allegheny Volunteer Service Leader) and concurrently take COMJ 520 *Connecting Action and Reflection I* during the first semester of the leadership position and COMJ 521 *Connecting Action and Reflection II* during the second (1 credit each).

Community and Justice Studies Courses

COMJ 160 *Introduction to Community and Justice Studies*
An introduction to the theories and ethics of social action, with a focus on community service. Theories of social dynamics and ethical systems are explored as a way to understand how social action can be useful to a community as well as the problems that can arise in implementing social action plans. Students participate in a service-learning component, which they reflect upon in writing and discussion, so as to better understand how the theories apply and where they may fall short. Attention is also paid to the ways in which class, race and gender shape the processes and outcomes of social action. *Prerequisites: first-year, sophomore or junior standing*. Distribution Requirements: CL, PD.

COMJ 260 *Interdisciplinary Methods for Social Research*
A study of the methods and tools of social research processes. We discuss quantitative research methods useful for analysis of social phenomena and problems including descriptive and basic inferential statistics. We also examine qualitative research methods appropriate for social action and participatory research projects. *Prerequisite: COMJ 160*. Distribution Requirements: CL.

COMJ 460 *Community Organizing and Civic Professionalism*
A study of the history and practices of community organizing as a methodology of social change and civic engagement. Through a seminar format, we trace key moments in a select group of movements for change and, through those cases, identify skills, values, and methods that are central to community organizing as a social and community practice. Students develop skills that are grounded in theory and history and that can be deployed in concrete social situations. *Prerequisite: COMJ 160*. Distribution Requirements: CL.

COMJ 520 *Connecting Action and Reflection I*
Part one of a two-semester course sequence in service learning. This seminar combines community engagement with guided reflection, and participants must hold concurrently, or have recently completed, a co-curricular service leader position. Examples of approved leadership positions include: Bonner Leader, Bonner Scholar, Allegheny Volunteer Service Leader, Davies Community Service Leader, Farhner Fellowship, CEED internship, America Reads Site Supervisor, and Computer Tutor Site Supervisor. One semester credit hour.

95

Figure 1.3: Example Page from Allegheny College Course Catalog

and Justice Studies can be gleaned. Courses and their descriptions can be seen as well as some requirements for a track. This is all well and good if Fred already knows that he is interested in Community and Justice. But as can be seen at the bottom of the figure, Fred is sifting through roughly 300 pages of information. Not only that but this catalog is from Allegheny College which only has roughly 2000 students and a correspondingly low number of courses. For students who still need to find what they want, infrastructure to aid in doing so is lacking. The collegiate system heavily favors students that seem to know what they want and does little to help those who do not.

This lack of ways for a student to gain information about their courses efficiently and quickly provides an impetus for a better system to be built. Some have tried to address this problem but none of the existing systems are entirely satisfactory. This work offers a better, more balanced solution. This is done by mining course information from the catalog and then visualizing that information in an interactive web application.

1.2 Current State of the Art

Few scholarly works have been produced attempting to visualize college course layouts. Those few include Sommaruga *et al.* and Zucker & Ron's systems [17][22].

As will be discussed further, these systems have many strengths that this work attempts to incorporate. However, these systems are also riddled with weaknesses and, in some ways, do not apply very well to the plight of first time, undecided undergraduate students. In addition, there has been a large amount of work done on what makes a visualization effective. Among these are visual aesthetic, color, and the level of interactivity which all benefit information retention. This work is extremely interdisciplinary as it spans areas of psychology, computer science, and education. All of this pertains purely to methods for improving information retention and retrieval. This work focuses on combining the strengths of the two previously cited systems while taking into account the latest science surrounding what makes an effective visualization.

1.3 Goals of the Project

Discussed in Section 1.1, the modern undergraduate student has many challenges to face that result in a graduation rate of only 60% [1]. One of these is a lack of easily obtainable information on courses and disciplines offered at a given institution. Ultimately, the goal of this work is to create a more efficient and satisfying college experience. This is done by maximizing students' ability to gain, retain, and use course information. With information made more accessible to students, they will be more aware of areas of study that are and are not at their institution as well as the steps to take to complete them. With this comes a more efficient overall experience.

Ideally, this application will result in students being able to explore more, learn more, and become more well rounded and interdisciplinary. Obviously, this is something that is hard to quantify. Put simply, we hope that this results in maximizing the original goal of the liberal arts education. This will both improve student happiness and long term satisfaction as well as increase graduation rates. This work in particular focuses on Allegheny College specifically to increase focus and thereby feasibility of the implementation.

1.4 Thesis Outline

Chapter 2 reviews Sommaruga *et al.* as well as Zucker & Ron's systems. It goes over their strengths, weaknesses, and how those apply to the Course Map. The goal of this chapter is to situate our work within the context of the only related works we found as well as provide a baseline for traits that this system was designed with in mind. It also discusses the results of various studies done surrounding what makes an effective visualization. These focus mostly on the effects of interactivity, overall visual appeal or aesthetic, and color on the efficacy of the visualization. Generally, it is found that all of these have a positive correlation with information retention. As such, these traits are a focus of this work. Chapter 3 outlines the system this work presents as well as the survey used to measure its perceived efficacy by students and faculty. Chapter 4 discusses the results of the survey and what that means for the Course Map and Allegheny's student body. This work concludes in Chapter 5 where conclusions are drawn based upon survey results and challenges faced. This chapter also discusses future work for the implementation based on features we want to implement and student survey responses.

Chapter 2

Related Work

2.1 What Makes a 'Good' Visualization

This work centers around creating a stronger medium for communicating course information to a user. This work aims to improve user experience in their interaction with a given institutions course information. User experience is defined as the ease with which users gain, retain, and use information presented within the visualization or interface. This makes the efficacy of my visualization a focus. The question then arises, what makes a good visualization of data?

2.1.1 Interactivity

For many people it instinctively makes sense that being able to interact with material increases their ability to remember said material. In essence, interaction is simply kinesthetic/tactile learning, or learning by doing. This has also been proven empirically in various studies.

Many studies have linked interaction with information acquisition, retention, and recollection [16][21][7][11]. Sims and Rod wrote an article on the subject of interactivity as it applies to education and learning [16]. They state that 'Interaction is intrinsic to successful, effective instructional practice as well as individual discovery'. Improving individual discovery fits perfectly with the goals of this work since one of the main goals is to improve students ability to explore. De Jong *et al.* also state that interactivity is 'a necessary and fundamental mechanism for knowledge acquisition and the development of both cognitive and physical skills' [7]. Kettanurak *et al.* performed a study examining how interactivity affects attitude and how that, in turn, affects six different dimensions of learning [11]. The areas they examined included content, format, user-control, feedback, ease of use, and motivation. They found that interactivity improved a persons attitude and that then improves their perceptions of some of the dimensions mentioned. Their improved perception would then ideally increase the efficiency of their interactions with the system. More recently, Qian Xu and S. Shyam Sundar performed a study on e-commerce sites [21]. They found that interactivity increased the overall cognitive processing of information. Interestingly, they also found that high interactivity reduced the total time spent on a site while medium interactivity resulted in the best recollection of non-interactive content [21]. We made this system with a medium level of interactivity in mind.

The use of interactivity in a visualization can be explained more intuitively using an infant as an example. Babies learn much by watching the world around them, but that is not how they explore. In order to learn, babies touch, put things in their mouth, and generally experience the world directly both through touch and action. Interacting with and engaging with something is often the best way to gain an understanding of it.

2.1.2 Aesthetic, Visual Appeal, & Color

Cawthon *et al.* performed a study on how perceived aesthetic of visualizations affects their use [5]. They found a correlation between aesthetic and how quickly information is obtained and how well its retained. This reinforces how important visual appeal and cleanness is. Color choice and the visual mechanics of a visualization are also extremely important. Morey *et al.* found a correlation between consistent color themes and information gained and retained [15].

This means that when a visualization uses similar colors or even the same color for things that are similar it allows a user to more intuitively absorb those connections. For example, in Figure 2.1 is a graph of synthetic data. There is no key and one looking at this graph has no idea exactly what it's communicating. But, one intuitively understands that things that are green are similar in some way. This intuition and ease of making connections results in the viewer being able to more easily absorb and

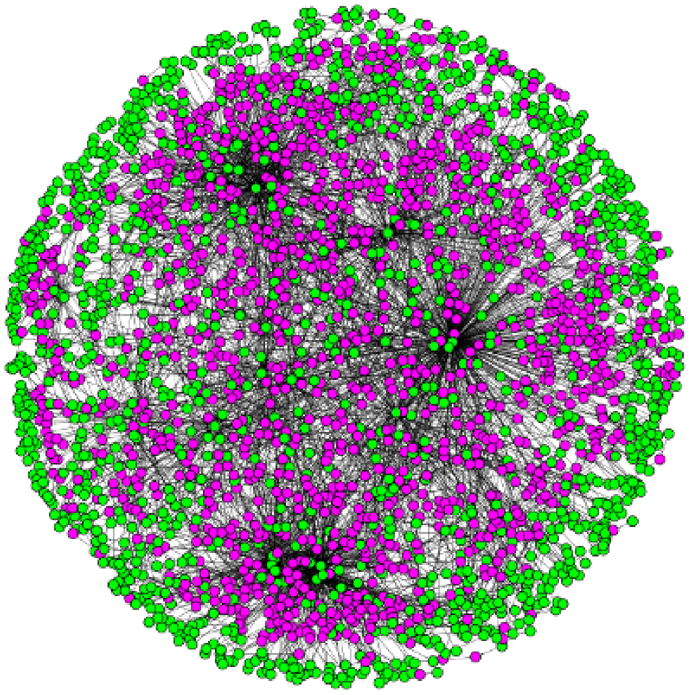


Figure 2.1: Example Graph Depicting the Power of Thematic Color

recall information contained in a visualization. For example, the viewer may intuitively understand that things that are green possess some property that drives them to the outside of the structure. This may seem simple or self explanatory, but this is only one example of the many connections that can be understood in an instant when intuitively visualized.

Just using color at all instead of gray scale has been shown to cause a 5-10% increase in memory performance [19]. This effect has also been shown to be magnified if the given color is matched with an emotion that is culturally identified with said color [12]. For example, green enhances memory for images associated with positive emotions whereas red enhances memory for images associated with negative emotions. Clearly the way information is communicated and displayed is important for retention.

In addition, the course catalog –being a black and white textbook– fails to take advantage of all these benefits. Based on this research, color, interactivity, and various dimensions of visual aesthetic are a focus of this work.

2.2 Current Systems

Some work has been done to visualize majors, departments, and even course catalogs. This section will discuss two such systems and analyze their strengths and weaknesses.

2.2.1 Sommaruga

Sommaruga et al. developed a 3D system for visualizing course curriculum [17]. Seen in Figure 2.2 this system visualizes a school's entire course catalog. Each department or discipline is broken up into its individual tracks with each of those tracks broken into modules represented as cubes.

Sommaruga's system also allows the user to hover or click on modules to see additional textual information [17]. It is also capable of visualizing an entire curriculum dynamically. It does this using an XML file and X3D to dynamically visualize a database containing the relevant information. A user is also able to navigate the visualized 'world' as well as limit the visualization to a given section of information as opposed to the entirety of it. The authors chose to use three dimensions to help them visualize the information but this decision gained them very little in terms of efficacy. They use the third dimension to visualize credit amount and length of a

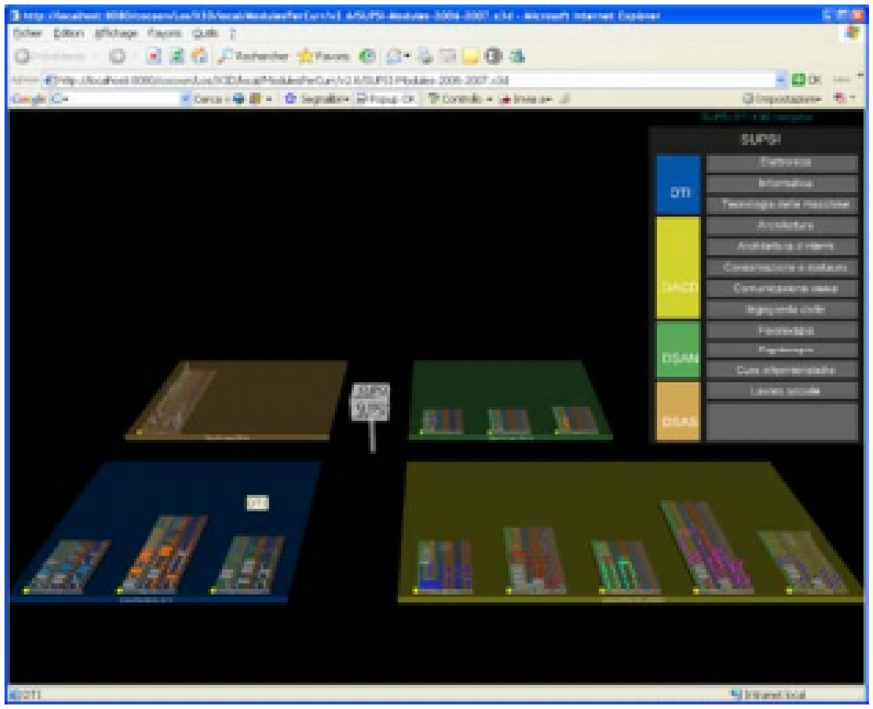


Figure 2.2: Sommaruga's 3D Visualization

module however the significance of differing sizes is not immediately apparent to the user without a scale for reference. This is also easily done in two dimensions using textual annotation. However, this decision does make sense in the overall context of the system given how much textual annotation there is already. The visualization also ends up looking extremely old and unappealing with a grey, orange, and blue color scheme. The color choices lack unity or thematic elements and end up cluttering the visualization rather than helping the user to intuitively gain information.

Sommaruga's system excels in terms of volume of information conveyed and interactivity. Since it is capable of displaying all of a given institutions course information,

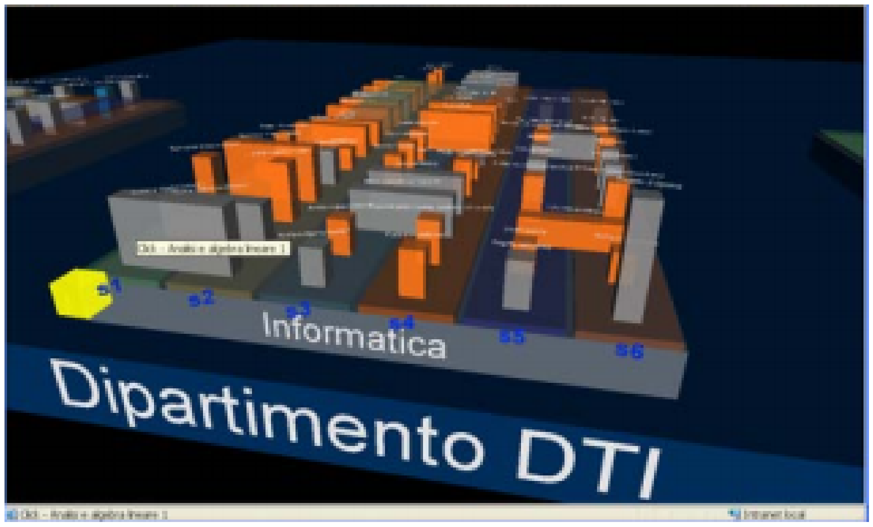


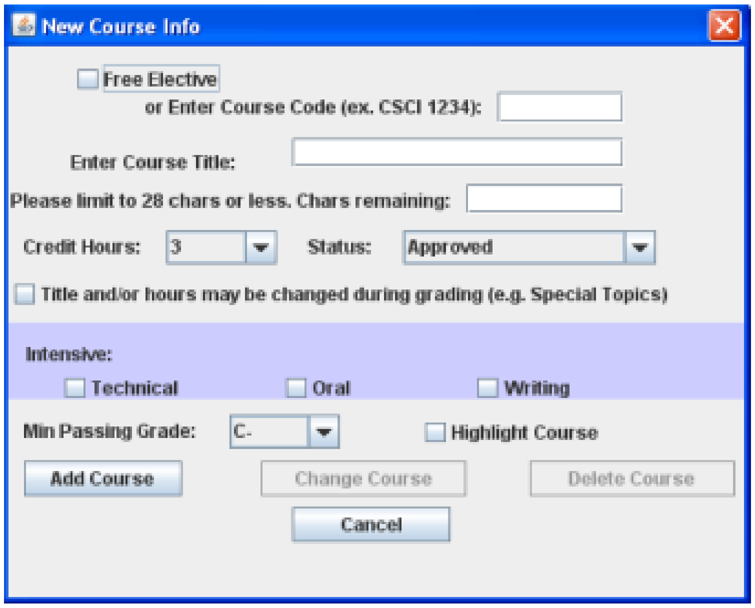
Figure 2.3: Sommaruga Enlarged View

this makes it useful to students prior to declaration; this allows students to explore information about any major prior to declaration. This is made more clear when put in contrast with the next system. Seen up close in Figure 2.3 Sommaruga is able to display a remarkably large amount of information in this visualization. Sommaruga's system exemplifies thoroughness of information conveyed and interactivity, but its weakness is a glaring lack of visual appeal and unity.

2.2.2 Zucker & Ron

Zucker & Ron have developed a small scale web chart tool used for representing a student's course of study [22].

Their system provides a framework for faculty to create graphs node by node. This customization allows the system a measure of flexibility in its application. One


New Course Info
Free Elective
or Enter Course Code (ex. CSCI 1234):
Enter Course Title:
Please limit to 28 chars or less. Chars remaining:
Title and/or hours may be changed during grading (e.g. Special Topics)
Intensive:
Technical
Oral
Writing
Highlight Course
Add Course
Change Course
Delete Course
Cancel

Credit Hours: 3

Status: Approved

Min Passing Grade: C-

Figure 2.4: Zucker Node Creation

can visualize an entire course flow at a school or simply a students course of study. Shown in Figure 2.4 the user of this system can specify details such as credit hours and even the minimum passing grade for the class. Once created these graphs can also be saved for later use. This allows the student or adviser to continue to view and use the visualization to help their student through the college process.

This tool uses color encoding to represent classes that are done, currently being taken, and scheduled for next semester. It places this information into a simple to read graph. The simplicity of the visualization makes gaining an understanding of the overall flow of the graph extremely easy. However, it is bare bones with little to no interactivity. The maps are also painstakingly created by hand instead of being generated dynamically based on a given store of information.

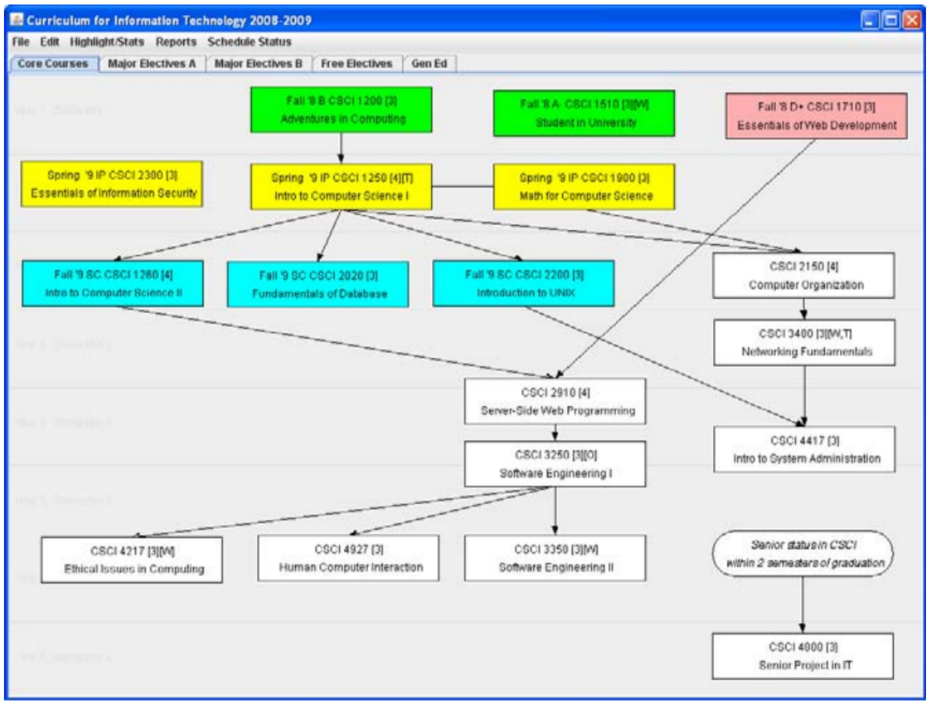

Curriculum for Information Technology 2008-2009
File Edit Highlight/Stats Reports Schedule Status
Core Courses
Major Electives A
Major Electives B
Free Electives
Gen Ed
Fall '8 B CSCI 1200 [3] Adventures in Computing
Fall '8 A- CSCI 1510 [3][W] Student in University
Fall '8 D+ CSCI 1710 [3] Essentials of Web Development
Spring '9 IP CSCI 2300 [3] Essentials of Information Security
Spring '9 IP CSCI 1250 [4][T] Intro to Computer Science I
Spring '9 IP CSCI 1900 [3] Math for Computer Science
Fall '9 SC CSCI 1260 [4] Intro to Computer Science II
Fall '9 SC CSCI 2020 [3] Fundamentals of Database
Fall '9 SC CSCI 2200 [3] Introduction to UNIX
CSCI 2150 [4] Computer Organization
CSCI 3400 [3][W,T] Networking Fundamentals
CSCI 2910 [4] Server-Side Web Programming
CSCI 3250 [3][O] Software Engineering I
CSCI 4417 [3] Intro to System Administration
CSCI 4217 [3][W] Ethical Issues in Computing
CSCI 4927 [3] Human Computer Interaction
CSCI 3350 [3][W] Software Engineering II
Senior status in CSCI within 2 semesters of graduation
CSCI 4800 [3] Senior Project in IT

Figure 2.5: Zucker's Visualization System

This systems most prominent strengths are conciseness of information, representation of a student's progress, and clear use of color to annotate information. The systems weaknesses come from the graphs having to be drawn by hand. The best option within the system for creating any large scale graph quickly is to import a formatted text file containing course information. But, the nodes are simply generated one after the other so the actual structure of the graph still has to be created by hand. So, outside of someone taking the time to manually determine the position of every course within a course catalog, this system won't serve our purpose. It is good at small scale personalized visualizations. Because of that it is mainly useful after a student determines their course of study.

As many students know, scheduling becomes relatively simple once their major and minor are declared. After this decision the process becomes focused on figuring out how to fill all the requirements. In the case of Allegheny College, a student also selects a personal adviser upon declaration that helps guide them through this process. Prior to declaring a student needs to get a hold of as much information as efficiently as possible. They also need to be able to do this independent of their adviser. This system does not help with that and is mainly useful only after a student has declared their course of study.

Our system attempts to combine and improve on the strengths of both these systems while eliminating weaknesses. We modeled this system after Sommaruga and it is capable of displaying all available course information for all areas of study while allowing the user to navigate or filter said information. It also learns from Zucker & Ron and places this information within a clean, simple, and intuitive graph. The combination of thoroughness of information representation and cleanliness of visualization combined with the latest research supporting interactivity and color choice places this work at the forefront of scholarly works attempting to visualize entire college course layouts.

Chapter 3

Method: The Course Map

This chapter walks through the process of text mining information out of the catalog found on Allegheny's website, visualizing it in R, and packaging it in a Shiny web application. It also covers the approach taken for evaluating user experience interacting with the system.

3.1 Text Mining in Python

While Allegheny College does make the course catalog publicly available on their website it is only published in a PDF format. Directly mining information from a PDF is extremely difficult so the first step was to convert the document to a plain text format. Originally, we used a python algorithm to accomplish this but the conversion created too much noise and too many artifacts within the text document for it to be easily mined. In addition, the python converter would lose new line characters within the document. As will be discussed later, the text mining process is centered on new line characters. After testing multiple publicly available converters, the website `http://www.zamzar.com/convert/pdf-to-txt/` was eventually settled on. This converter provided a plain text document with minimal noise and no missing new line characters.

Python was chosen as the language to implement the text mining component in because of its high level of abstraction, 'pythonic' properties, and many libraries for manipulating text. Python's white space formatting help make code more readable and cleanly. The fact that all of its data types are objects and there is extremely thorough documentation on how to use and manipulate lists and other data make development much easier. All of these make handling data and text more straight forward. Many modern text mining strategies employ python and its NLTK library [18][13][4][14]. It is for these reasons that kdnuggets.com found that 36% of all people use it for data mining. Currently, the python script is capable of mining the information found in both the course catalog and a spread sheet obtained from the registrar containing all courses offered at Allegheny College. As will be discussed later, it is unable to visualize the information found in the spread sheet.

While there has been massive amounts of work done with text mining in various contexts and languages, we have not found anything closely resembling the work done here. The strategy laid out will also be better understood given some context about the course catalog. While doing a very good job of containing a large amount of information, the catalog is not consistently formatted. It appears to have been written by hand and by many different people. The overall structure of the catalog is somewhat consistent, but there is enough variance to make text mining difficult. Because of this, the core approach was to examine the catalog by eye and find patterns general enough to allow acquisition of as much course data as possible. The first pattern that was noticed is that course titles are limited to –and entirely take up– a single line of text. The second pattern is that all of the course summary paragraphs are preceded and followed by a blank line. Because of this the script functions on Regular Expressions (RE) and new line characters. The overall flow of the algorithm can be seen in Figure 3.1

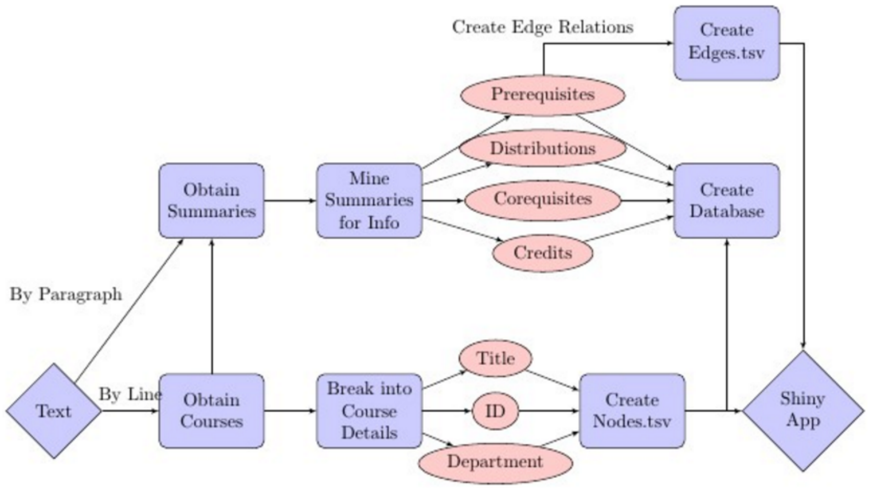


Figure 3.1: Flow Chart of Text Mining Operation

The text miner breaks the document up line by line based on the new line characters discussed before. It then scans those lines for RE matches to

```
acro+' [0-9]{3} .*\n'
```

This RE is scanning for any line that consists of an acronym, a three number code, and a title. Acro is any department acronym. For example, BIO or RELST. A given match may look like BIO 220 Organismal Physiology and Ecology. After this, various cleaning processes occur to weed out false positives from the RE scan. Next, the algorithm splits the original text into paragraphs based on pairs of new line characters found between each paragraph. Spoken plainly, there is a blank line between each paragraph in the catalog and that gap is what we search for. It then scans all the paragraphs seeing which ones begin with the courses found before and matches them up. Next, courses are tokenized and broken up into details such as department, course number, etc. This is followed by also tokenizing the summaries and searching them for other pertinent information. Information found within summaries includes prerequisites, co-requisites, credits, and distribution tags. At this point all the data has been obtained. From here, various processing steps occur. Prerequisite text is mined so as to establish edge relationships for the graph. The information is also organized into three different tsv files which are output into the R projects working directory and used for graph and database generation. The *database.tsv* file contains all course information and is used to populate a SQLite database. This database is used for debugging purposes and later becomes a part of the UI. Basic course information including id, department, and title is stored in the *nodes.tsv* file which is used for creating the nodes within the graph. The edge relationships that were previously established are stored in an *edges.tsv* file that's used to create the edges within the graph. The next step is graphing the data in R.

3.2 Visualizing in R

R is a well known language for statistical analysis and visualization. That fact in addition to its supremely clean and visually appealing graphs makes it the best option for graphing course information. The R script for graphing the information is extremely simple. It employs the visNetwork package which allows for the graphing of interactive force directed graphs. The script takes the previously mentioned nodes.tsv and edges.tsv files as input and outputs a graph similar to the one seen in Figure 3.2.

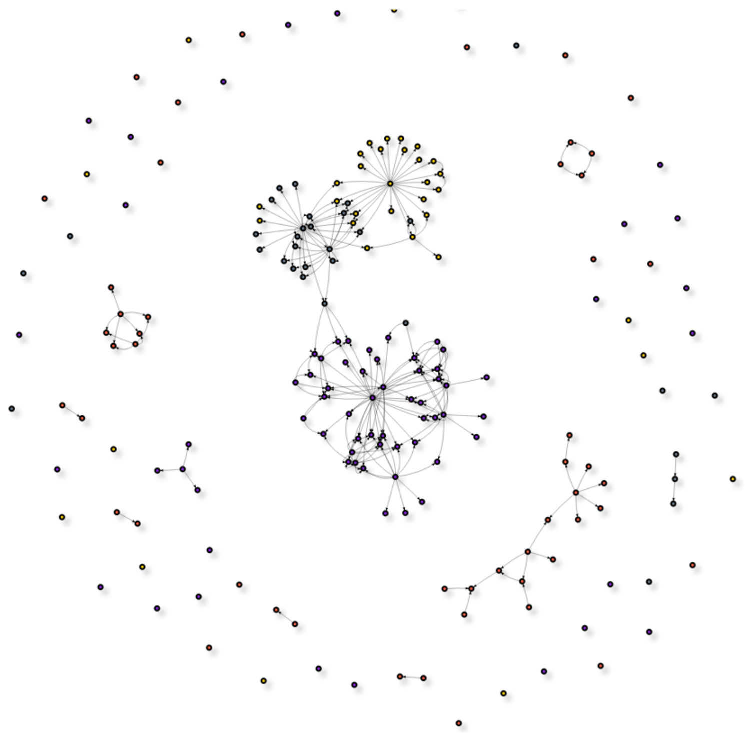


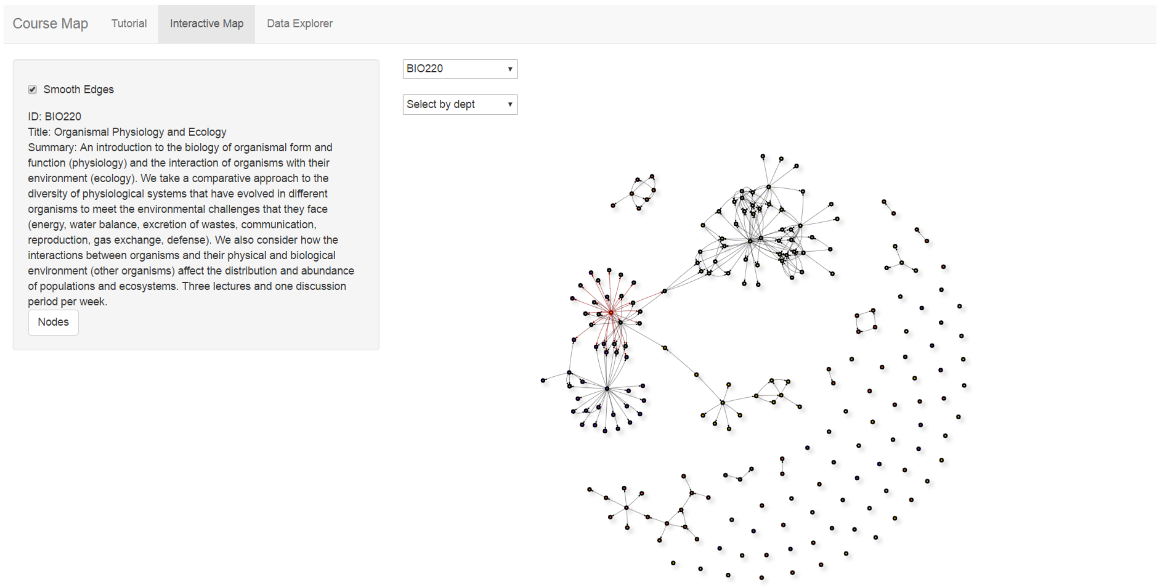
Figure 3.2: Example Graph

Classes are represented by nodes with directional edges between nodes showing prerequisite relationships. Just by looking at the graph one can see clusters of classes and understand various relationships. Clusters of classes are interrelated in that some are prerequisites of the others. Large clusters are usually entire departments or at least the majority of the department. Nodes that are unconnected have no prerequisites according to the text miner.

The visNetwork package also allows for some user interaction. Users can hover over courses to display information as well as search for classes by id (BIO 221) and highlight entire departments at once. However, all of this happens within the RStudio IDE. Clearly, this setup is not usable by the vast majority of people with no experience in software development. Some additional framework is required to package the system in such a way that any user can intuitively interact with it. The Shiny package was chosen to facilitate this.

3.2.1 Shiny Web Application

The R Shiny package was determined to be a usable vehicle for packaging and delivering the visualization to users. Shiny is a package that allows for the creation of web applications inside the context of R. It also allows for reactive visualization of data and user interaction. Shiny takes specialized R code and converts it into HTML and CSS. The shiny framework consists of a *server.r* script and a *ui.r* script. The *server.r* script contains all the R code that graphs the data and controls all operations that occur behind the scenes. The *ui.r* file is where the actual interface is constructed. These two files interact and share information allowing for the graph to react to user input.



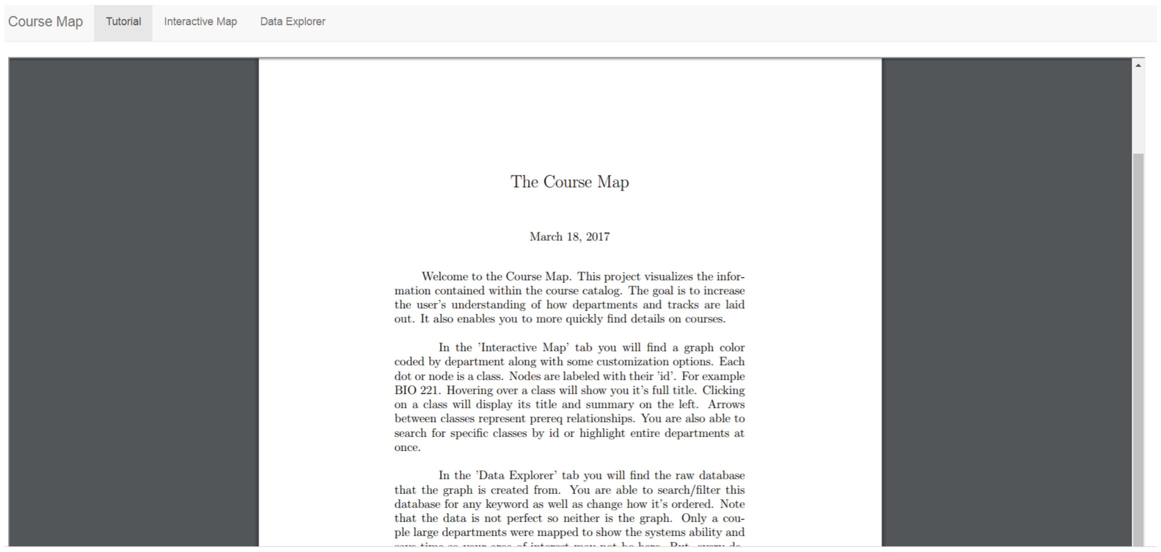
☑ Smooth Edges

ID: BIO220
Title: Organismal Physiology and Ecology
Summary: An introduction to the biology of organismal form and function (physiology) and the interaction of organisms with their environment (ecology). We take a comparative approach to the diversity of physiological systems that have evolved in different organisms to meet the environmental challenges that they face (energy, water balance, excretion of wastes, communication, reproduction, gas exchange, defense). We also consider how the interactions between organisms and their physical and biological environment (other organisms) affect the distribution and abundance of populations and ecosystems. Three lectures and one discussion period per week.

BIO220
Select by dept

Figure 3.3: Interactive Map Page

Inside this framework, we created a more usable interface seen in Figure 3.3. With this web interface comes added interactivity and usability. Being contained in a web application and having an intuitive multi-tab interface makes the system much more accessible. Users of the age 18-22 have spent large portions of their life using interfaces and applications similar to this one. Every popular web browser uses a tab based layout such as this one. It's also reasonable to assume that users are comfortable with using a system that allows for click/drag and zoom in/out functionality. The application is a three tab interface. Shown here is one of three tabs within the application. Labeled 'Interactive Map', it's the most important tab and contains the graph and various customization options. The first tab labeled 'Tutorial' can be seen in Figure 3.4. This tab is simple with an embedded pdf containing the tutorial. This pdf has a link in the bottom to a Google form which evaluates the perceived usefulness of the system. The tutorial provides extremely basic information about the Course Map so users do not feel completely lost. It briefly explains the three tab layout and features that are present while encouraging the user to take the survey which will be discussed later.



The Course Map

March 18, 2017

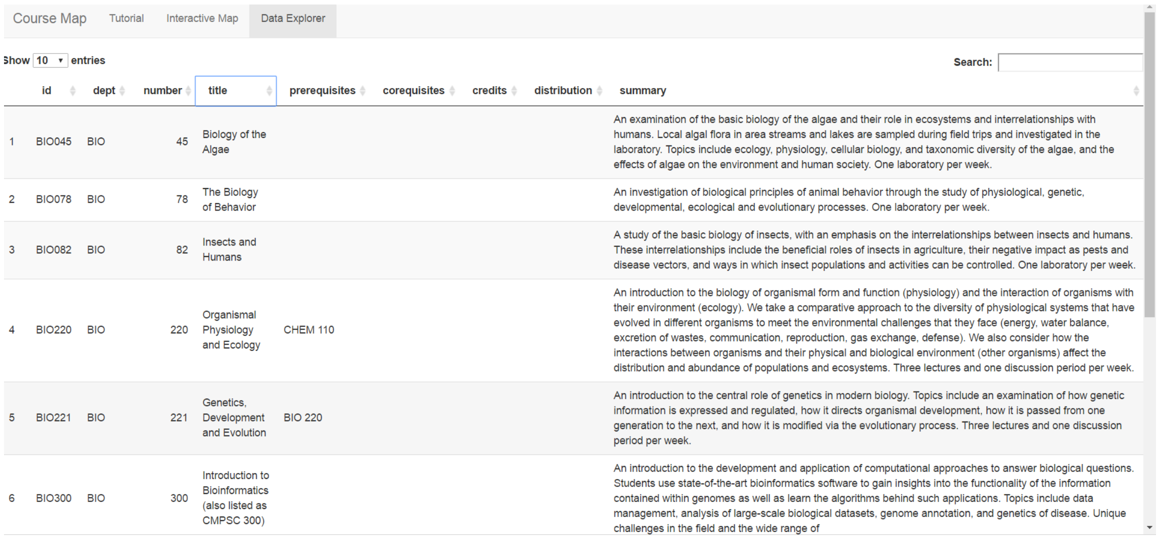
Welcome to the Course Map. This project visualizes the information contained within the course catalog. The goal is to increase the user's understanding of how departments and tracks are laid out. It also enables you to more quickly find details on courses.

In the 'Interactive Map' tab you will find a graph color coded by department along with some customization options. Each dot or node is a class. Nodes are labeled with their 'id'. For example BIO 221. Hovering over a class will show you it's full title. Clicking on a class will display its title and summary on the left. Arrows between classes represent prereq relationships. You are also able to search for specific classes by id or highlight entire departments at once.

In the 'Data Explorer' tab you will find the raw database that the graph is created from. You are able to search/filter this database for any keyword as well as change how it's ordered. Note that the data is not perfect so neither is the graph. Only a couple large departments were mapped to show the systems ability and

Figure 3.4: Course Map App: Embedded Tutorial PDF

The final tab titled 'Data Explorer' shown in Figure 3.5 allows the user to interact with the database the graph is created from. Headers include basic course information such as id and department as well as more complicated information such as the summary. Users are able to search for any keyword as well as reorder columns. The reordering of columns and search box are similar to shopping cites and other such interfaces all over the web. For example, when shopping on Amazon a customer is able to order search results based on price, relevance, and other categories. Because of this, users should adjust to and understand the use of the database very quickly. This applies even if they have no previous experience working with information organized in a relation database format.



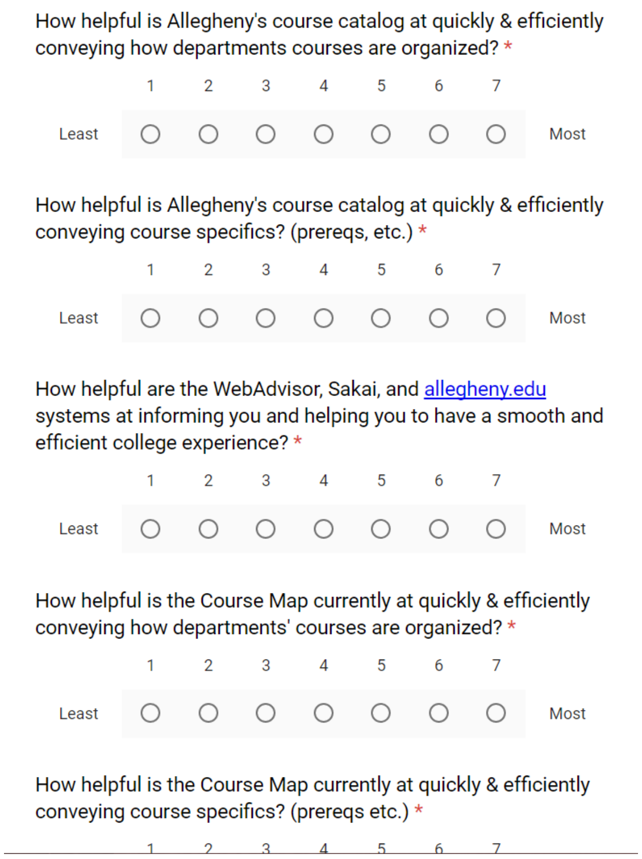
	id	dept	number	title	prerequisites	corequisites	credits	distribution	summary
1	BIO045	BIO	45	Biology of the Algae					An examination of the basic biology of the algae and their role in ecosystems and interrelationships with humans. Local algal flora in area streams and lakes are sampled during field trips and investigated in the laboratory. Topics include ecology, physiology, cellular biology, and taxonomic diversity of the algae, and the effects of algae on the environment and human society. One laboratory per week.
2	BIO078	BIO	78	The Biology of Behavior					An investigation of biological principles of animal behavior through the study of physiological, genetic, developmental, ecological and evolutionary processes. One laboratory per week.
3	BIO082	BIO	82	Insects and Humans					A study of the basic biology of insects, with an emphasis on the interrelationships between insects and humans. These interrelationships include the beneficial roles of insects in agriculture, their negative impact as pests and disease vectors, and ways in which insect populations and activities can be controlled. One laboratory per week.
4	BIO220	BIO	220	Organismal Physiology and Ecology	CHEM 110				An introduction to the biology of organismal form and function (physiology) and the interaction of organisms with their environment (ecology). We take a comparative approach to the diversity of physiological systems that have evolved in different organisms to meet the environmental challenges that they face (energy, water balance, excretion of wastes, communication, reproduction, gas exchange, defense). We also consider how the interactions between organisms and their physical and biological environment (other organisms) affect the distribution and abundance of populations and ecosystems. Three lectures and one discussion period per week.
5	BIO221	BIO	221	Genetics, Development and Evolution	BIO 220				An introduction to the central role of genetics in modern biology. Topics include an examination of how genetic information is expressed and regulated, how it directs organismal development, how it is passed from one generation to the next, and how it is modified via the evolutionary process. Three lectures and one discussion period per week.
6	BIO300	BIO	300	Introduction to Bioinformatics (also listed as CMPSC 300)					An introduction to the development and application of computational approaches to answer biological questions. Students use state-of-the-art bioinformatics software to gain insights into the functionality of the information contained within genomes as well as learn the algorithms behind such applications. Topics include data management, analysis of large-scale biological datasets, genome annotation, and genetics of disease. Unique challenges in the field and the wide range of

Figure 3.5: Course Map App: Database Tab

This entire interface is designed by Shiny defaults. The interface created is sleek in design as well as color choice. The web application's border and background are presented in an extremely clean gray scale. The lack of noise allows the user to more quickly identify the key points of the interface and learn how to use them. Overall, the system created is aesthetically pleasing, clean, and efficient. The next step, then, is evaluating users perceived usefulness of the Course Map versus their previous experience with the course catalog.

3.3 Evaluating Perceived Efficacy

There was not enough time to conduct a thorough survey regarding how the system affects information retention and recollection. This would have required a controlled study investigating how well students retain course information when using the catalog as compared to the Course Map application. However, users were given a brief survey evaluating their perceived usefulness of the course catalog, sakai, web advisor, and the Course Map System. Part of this survey can be seen in Figure 3.6.



How helpful is Allegheny's course catalog at quickly & efficiently conveying how departments courses are organized? *

	1	2	3	4	5	6	7	
Least	○	○	○	○	○	○	○	Most

How helpful is Allegheny's course catalog at quickly & efficiently conveying course specifics? (prereqs, etc.) *

	1	2	3	4	5	6	7	
Least	○	○	○	○	○	○	○	Most

How helpful are the WebAdvisor, Sakai, and allegheny.edu systems at informing you and helping you to have a smooth and efficient college experience? *

	1	2	3	4	5	6	7	
Least	○	○	○	○	○	○	○	Most

How helpful is the Course Map currently at quickly & efficiently conveying how departments' courses are organized? *

	1	2	3	4	5	6	7	
Least	○	○	○	○	○	○	○	Most

How helpful is the Course Map currently at quickly & efficiently conveying course specifics? (prereqs etc.) *

1 2 3 4 5 6 7

Figure 3.6: Course Map Survey

A Likert-scale was used for evaluating users perceived usefulness of the Course Map web application as well as the course catalog and other existing systems. These systems include WebAdvisor, sakai, and allegheny.edu. While commonly accepted as a robust evaluation tool, the Likert-scale seemed especially useful for this study. It maintains a range large enough to allow for variation while still having a middle point for users that are not entirely convinced either way. It should be noted that this system is not being compared in its current state to WebAdvisor etc. Rather, the system is being compared to the course catalog and the users perception of the Course Map's *potential* is being compared to WebAdvisor etc. It was hypothesized that students will find the Course Map system more intuitive, interactive, appealing, and useful than the course catalog.

The first two questions ask students to rate the catalogs ability to convey course specifics and how departments have their courses organized. For example, how easy is it to figure out that BIO220/221 are prerequisites for nearly everything? Providing this information is the purpose of the catalog and from these questions we gain an understanding of how useful students find it. However, that catalog is not the only tool available to students. So, they are then asked to rate how useful and informative other tools including Sakai and WebAdvisor are. Students are then asked two questions about the Course Map paralleling those regarding the catalog. These are also to evaluate perceived usefulness of the Course Map for comparative analysis with the catalog. Finally, students are asked to rate the potential that they see in the Course Map system –given further development– at informing and aiding their college experience. This is later compared to their ratings of existing systems. Students are given the opportunity to provide feedback on the strengths and weaknesses of the system but these are not represented in graphical form. The outcomes of the survey are discussed in Chapter 4.

3.4 Threats to Validity

The goal of this work is not to empirically prove that the Course Map system conveys information more effectively than the catalog. Rather, it's to show that users have a smoother and better experience with the Course Map, and at the very least believe that there is potential for a system like this one to improve their college experience. Therefore, one of the biggest threats to validity is variation within the survey data. Students who are underclassmen, and especially freshmen, are much more likely to be frustrated with the course catalog and find an alternative appealing. Whereas upperclassmen who have been exposed to the catalog for much longer will inevitably perceive it as more useful overall. The web application also works on mobile devices which is going to offer a much different and more tactile experience. This in turn will affect users responses to the systems usefulness. Finally, on such a small campus, personal bias should be taken into account. Many of the people who responded to the survey know me personally, or at least know of me, for better or for worse. Said bias could be strong enough to influence responders to rate our system based on their opinion of me instead of their objective view of the system. I tried to minimize this by removing the author from the web application and its url. That way when people received the link to the web application, unless names were mentioned, they would have no idea who the author is.

Chapter 4

Results

This chapter discusses the results of the survey. The data is presented in graphical form and the Course Map is compared to the course catalog and other pre-existing systems.

The survey used to evaluate user experience with the Course Map system v. the Course Catalog was extremely simple. Despite this, the results are very informative. Shown in Figure 4.1 are the global averages for how well users thought the catalog and Course Map convey course specifics and overall department organization. The figure also shows students' outlook on the potential of the Course Map system v. current systems.

As one can see, the course map system was rated on average .66 points higher in specifics and .94 for organization . The difference is even more stark when comparing existing systems to the potential of the Course Map. Users were asked how well Sakai, WebAdvisor, and `allegheny.edu` inform them and aid their college experience. They were also asked how much potential they see in a system like the course map –given

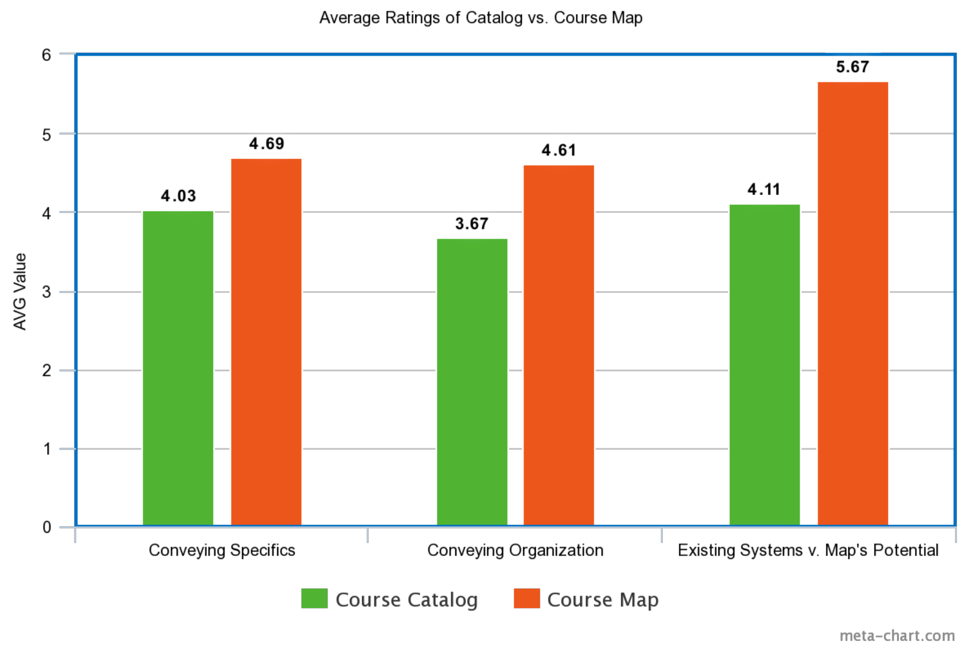


Figure 4.1: Survey Results: Global Averages of the Catalog and Course Map

further development– for doing the same. As we can see, users rated the potential of the course map a full 1.56 points above current systems.

When the results are grouped based on class year of the respondents, we note marked differences in behavior between classes. Shown in Figure 4.2 are the course specifics and department organization mentioned in the previous figure. First, as one would expect given Figure 4.1, the Course Map's overall ratings are higher than the catalog's. But if we examine how individual class years ratings change we notice something interesting: Sophomores, Juniors, and Seniors ratings remain somewhat constant with an average increase from the catalog to the Course Map. Freshman ratings of the course catalog are nearly a full point higher for specifics. They also have the lowest rating overall for how well the Course Map shows organization. Despite that, they maintain the highest rating for the Course Maps ability to show course specifics.

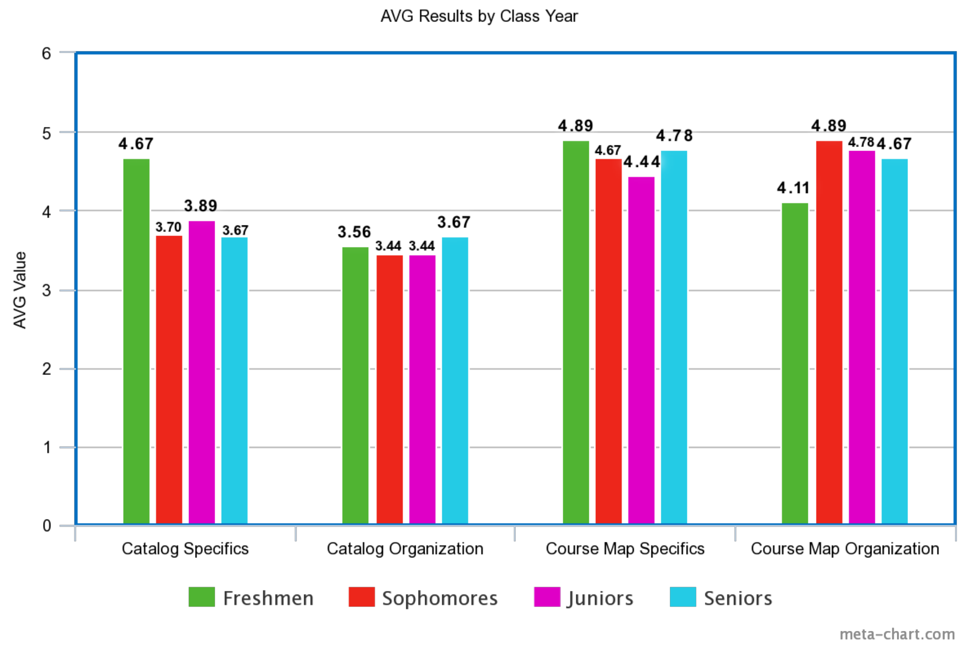


Figure 4.2: Survey Results: Specifics and Organization by Class Year

The three juxtaposed give us insight into a very important process that is occurring. When first given the course catalog, freshmen have little freedom compared to their more experienced counterparts. A freshman's first semester is locked in prior to reaching campus and both their first and second semesters consist mostly of freshmen seminars and baseline prerequisites. It is possible that because most freshmen schedules are so simple, they feel satisfied with how easy it is to obtain specifics because they already know what courses to look at. Once you know exactly what you want, the catalog is relatively effective. It isn't until sophomore year that many students realize how many options there are. This is also when students are confronted with having to declare their major by the end of the year. Because of this, they start to explore. So, this is when they first use the catalog in any large capacity. The attempt to explore —which the catalog does not facilitate well— leads to lower ratings. What's more difficult to interpret is that this pattern doesn't hold for students' opinions of how well the catalog communicates overall department organization. In this category, freshmen rate the catalog on par with the other class years. Like the other students, they find the catalog insufficient for explaining how departments organize their courses overall.

On the other hand, the different class years' opinions of the Course Map's potential were consistent. Potential is an important word here, and the word that was actually used in the question. Because the question is open to the responders imagination, interpreting the results is difficult. Shown in Figure 4.3 are these ratings broken up by class year.

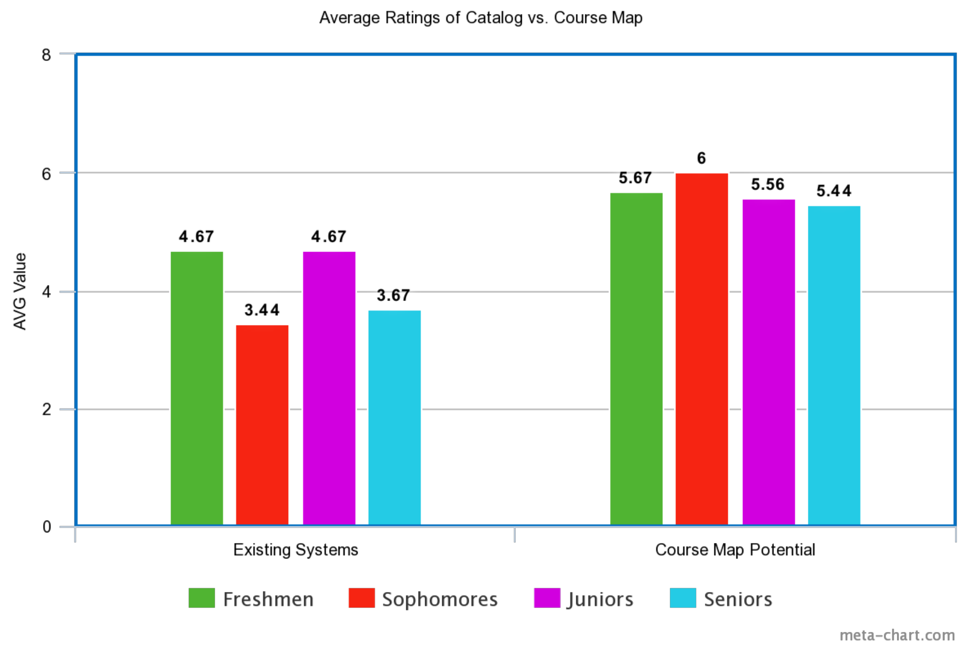


Figure 4.3: Survey Results: Efficacy of Current Systems v. Potential of Course Map

We do not have a very good way of knowing the rational behind students' potential ratings. What we can see is that they universally believe that a system like this one could serve them better than currently existing systems. While the grass is always greener on the other side, these margins are too large to be ignored. A few improvements were mentioned in multiple students' responses. These may be part of their belief in the system's potential, and will be more directly addressed in the future work section. What's more complicated are the students' opinions of Allegheny's current systems. Sophomores and seniors rate the current systems roughly a full point lower than freshmen and juniors do. Perhaps this is because these are the two most planning intensive years for students. Sophomore year is when many students have to map out their intended college career and senior year contains the senior composition.

Chapter 5

Discussion and Future Work

This chapter summarizes the results, discusses possible future work, and draws conclusions from the results.

5.1 Summary of Results

The Course Map and course catalog were rated from one to seven on their ability to quickly and efficiently convey course specifics and department layouts. The Course Map was given a 4.69 and 4.61 for specifics and department organization respectively. The course catalog was given a 4.03 and 3.67 . Freshmen especially favored the catalog's ability to convey course specifics rating it roughly a full point higher than the other class years did. Despite that belief, they rated the catalogs ability to convey department organization consistent with the other class years. As we can see, the Course Map received significantly higher ratings than the course catalog. The difference is even more stark when comparing satisfaction with current systems to a belief in the potential of the Course Map. Overall, the students surveyed are only moderately satisfied with existing systems giving sakai, webadvisor, and `allegheny.`

`edu` a score of 4.11. The margin between their view of existing systems and their belief in the potential of the Course Map is quite large with the Course Map's potential attaining an average rating of 5.67.

5.2 Future Work

The Course Map system is far from complete. Limited time and resources forced many decisions to be made regarding feasibility. We also didn't have any experience with R prior to this work which slowed the implementation process down considerably. One of the largest features of the system that we were unable to implement was user control of the visualization. That is, the ability for the user to control which departments were being visualized and have the system reactively respond to that. We were unable to implement this in R exclusively. This required turning to the python script. However, the package that allows one to call python scripts within R –rPython– doesn't work on windows. There was not enough time to work around this. This is possibly the most important feature we were unable to implement in time.

We also found research regarding color choice in visualizations only after sending out the working version of the system. There is recent research indicating that users are cognitively less efficient when the background of their tablet is white instead of colored [3]. So changing the background may be something to consider in the future. As was discussed earlier there is also research indicating that color has a powerful effect on memory when its combined with a pre-existing emotional attachment [12]. For example, green enhances memory of an object when said object has a positive emotional connotation and red enhances negative memories. Because the Course Map system displays various departments that are color coded we are unable to take full advantage of this. There is also no way of knowing what emotional connotations users have for certain departments. There may never be a way to take advantage of the boost from emotional connotation, but quite a bit of future work on the system will focus on the use of color.

Currently, the Course Map's exact layout changes every time the graph is reloaded. Reloads happen anytime the 'smooth edges' option is changed or the page is refreshed. Cognitive studies have been done showing that working memory is mainly important when someone is first interacting with an object. As time goes on, long term memory is employed to control where a person's attention goes [20]. Because of this work it has been conjectured that combining working and long term memory will result in optimizing attention and cognitive abilities [9]. This research combined with survey responses would suggest that making the layout consistent would be the most logical decision in the future.

Another feature that was considered and mentioned numerous times in survey responses is the ability of the system to display tracks. For example, a student intent on computer science may want to be able to see what the Applied Computing: Software Development track entails. We decided that this feature was too difficult to develop in time, but we are intent on this, and other features like it, being implemented in the future.

One final possibility is for personalization based on the user and integrating the Course Map with Allegheny College's pre-existing systems. For example, users would login to the Course Map and the system would know what classes they have taken and are taking. Based on this information, it could display classes that are the most relevant, show courses being offered next semester that they could take, and even offer classes as suggestions that the user may not have known about or considered. For example, a history minor interested in math may not know that there is a history of mathematics course. Given the ability to save previous user information or have access to Allegheny's systems, there are many useful features that can be implemented.

5.3 Consequences & Allegheny's Reaction

There are a number of consequences and effects that may take place in the event that a system like this one is deployed within Allegheny College. A potential consequence of students being able to more clearly see how departments are laid out is that they would no longer take classes that can be viewed as unnecessary. That is, classes not at the center of a department's progression. However, every department is designed differently. Biology is entirely centered upon its two intro courses. Art is a number of sub disciplines that build off of each other. The more diverse and broad a department is the more its courses are correspondingly spread out or unattached to each other. There would mostly likely be a change in how students enrolled in courses. But, that would be a consequence of how departments are already designed and would eventually lead to a better structure. Because students are able to see and adapt to department structure, the departments would consequently have to adjust their structure to accomplish whatever goal they set out for themselves. Biology likely will stay extremely centered and linear. Other departments may change and become more linear or less so.

Ultimately, I think that having a map of all the departments will enable students to more easily find what they want. It will also enable faculty to organize things so as to allow students to get to what they want more efficiently while providing a liberal arts education.

5.4 Conclusion

The Course Map system is far from complete. Despite this, students found it very useful —even more useful than the course catalog. They also believe immensely in the potential of a system like this one to inform and improve their college experience. Freshman specifically find the course catalogs ability to communicate course specifics helpful. This is contrasted with their average view of the catalogs communication of department organization and the highest class rating of the Course Map's specifics. There is much work to be done if this system – or one like it – is ever to be used on a grand scale. But, student responses and belief would indicate that said work is well worth the effort.

Appendix A

Python Code

```
#################################################################
# Text Mining Portion of the Course Map System
# Author: Dillan Smith
# All rights reserved
# Date: 4/4/2017
# Takes a text file containing Allegheny College's course catalog
# and breaks it up by line and by paragraph. It searches that
# corpus for courses, breaks up that information, and outputs into
# 3 tsv files which are used in the R portion of this system
#################################################################
import sys
import os
import pdfminer
import nltk
import re
import sqlite3 as lite
import string

# all the course acronyms that I know of
# 'ARAB', 'ART', 'BCHEM', 'BLKST', 'BIO', 'CHEM', 'CHIN', 'CLC', '
   CMPSC', 'COMJ', 'COMRT',
# 'DMS', 'ECON', 'EDUC', 'ENGL', 'ENVSC', 'EXL', 'FRNCH', 'GEO', '
   GERMN', 'GHS', 'HIST',
# 'INTDS', 'INTST', 'JOURN', 'LATIN', 'MATH', 'MUSIC', 'NEURO', '
   PHIL', 'PHYS', 'POLSC',
# 'PSYCH', 'RELST', 'SPAN', 'FS', 'FSART', 'FSBIO', 'FSCHE', 'FSCOM'
   , 'FSDMS', 'FSECO',
# 'FSENG', 'FSENV', 'FSFRE', 'FSGEO', 'FSGER', 'FSGHS', 'FSHIS', '
   FSMAT', 'FSMUS',
# 'FSNEU', 'FSPHI', 'FSPHY', 'FSPOL', 'FSPSY', 'FSREL', 'FSPA'
# note: FSMLG should be in here but its throwing an error
dept_acronyms = {"BIO", "PSYCH", "COMRT", "ENVSC", "CMPSC"}

def clean_text_file():

    corp = open('2016-17coursePDF.txt', 'r')
    line_txt = []
    # parse the original txt file by line
    while True:
        line = corp.readline()
        if line == "":
            break
        else:
            line_txt.append(line)
    corp.close()

    # remove spaces prior to \n
    line_txt = remove_ghost_spaces(line_txt)

    # write the cleaned lines back into a new working version of the
        text
    corp = open('workingVersion.txt', 'w')
    for line in line_txt:
        corp.write(line)
    corp.close()


def remove_course_lists(chunkList, checkList):
    for paragraph in chunkList:
        courseInPara = 0 # counter for course lines in the paragraph
        paragraph = paragraph.split('\n') # parse by line
        for course in checkList:
            for line in paragraph:
                if course == line+'\n':
                    courseInPara += 1
        if courseInPara > 1:   # remove all paragraphs with more
            than one course
            chunkList.remove('\n'.join(paragraph))

    return chunkList

#clean for course names that appear more than once
def remove_duplicates(values):
    output = []
    seen = set()
    for value in values:
        if any(word in ['recommended','required'] for word in value.
            split(' ')):
            values.remove(value)
    values = remove_ghost_duplicates(clean_special_characters(values
        ))
    for value in values:
        # If value has not been encountered yet,
        # ... add it to both list and set.
        if value not in seen:
            output.append(value)
            seen.add(value)
    return output
```

```
def clean_special_characters(values):
    invalidChars = set(string.punctuation.replace(":", ""))
    invalidChars2 = set(string.punctuation.replace(",", ""))
    check1 = []
    check2 = []
    output = []
    for value in values:
        splits = value.split(' ')
        if len(splits) >= 3:
            if any((char in invalidChars for char in splits[2]) and
                (char in invalidChars2 for char in splits[2])):
                continue
            output.append(value)
    return output


def remove_ghost_duplicates(values):
    output = []
    for x in range(len(values)):
        sample = values[x]
        if sample[len(sample)-1] == '@':
            continue
        sampleSplit = sample.split(' ')
        real_index = x
        for y in range(x+1,len(values)):
            sample2 = values[y]
            if sample2[len(sample2)-1] == '@':
                continue
            sampleSplit2 = sample2.split(' ')
            if sampleSplit2[:2] == sampleSplit[:2]:
                real_index = y
                values[y] += '@'
        values[x] += '@'
        output.append(values[real_index][:-1])

    #for value in output:
        #print value
    print "CLEANED FOR DUPLICATES"
    return output


def remove_ghost_spaces(temp_txt):
    # remove all spaces immediately prior to newlines in the lined
       text
    for x in range(len(temp_txt)):
        while True:
            if temp_txt[x][len(temp_txt[x])-2]+temp_txt[x][len(
                temp_txt[x])-1] == ' \n':
                newline = temp_txt[x][:len(temp_txt[x])-2] +
                    temp_txt[x][len(temp_txt[x])-1]
                temp_txt[x] = newline
            else:
                break
    return temp_txt


def get_courses(temp_txt, acronyms):
    temp_courses = []
    for acro in acronyms:
        for line in temp_txt:
            check = tuple(re.finditer(acro+' .{3}', line, flags=0))
            if len(check) > 1:  # skip lines that contain more than
                one course acronym combo
                continue
            matchObj = re.match(acro+' [0-9]{3} .*\n', line, flags
                =0) # look for match with CMPSC ### . . .
            if matchObj is not None:
                if matchObj.group():
                    temp_courses.append(matchObj.group())
    print "COURSES OBTAINED"
    return temp_courses


def get_summaries(temp_paragraphs, temp_courses):
    summaries, new_courses = [], []
    for para in temp_paragraphs:
        line_para = para.split('\n')  # break each paragraph up by
            line

        if len(para) < 3:  # dont try to process garbage paragraphs
            continue
        for course in temp_courses:  # search for course that
            matches start of para
            if line_para[0] == course[:len(course)-1]:  # courses
                have \n whereas .split strips them off line_para
                summaries.append('\n'.join(line_para[1:]))  # append
                    rejoined paragraph to working summaries
                new_courses.append(course)  # append working course,
                    index matches the above summary
    print "RETURNING SUMMARIES WITH MATCHING COURSES"
    return summaries, new_courses


def parse_summaries(summaries, prereq, coreq, cred, dist):
     for y in range(len(summaries)):
        prereq.append('')
        cred.append('')
        dist.append('')
        coreq.append('')
        tokens = nltk.word_tokenize(summaries[y])
        for x in range(len(tokens)):
            # check for prereq trigger
```

```
if tokens[x] == 'Prerequisites' or tokens[x] == '
   Prerequisite':
    x += 1  # compensate for colon, avoid trying to
       reference out of range index
    temp = x
    temp += 1
    while True:

        # if it's the last word in the sentence (theres
           a period) cut out the garbage space in
        # the prereq and quit
        if tokens[temp][len(tokens[temp])-1] == '.':
            prereq[y] = prereq[y][:-1]
            break

        prereq[y] += tokens[temp] + ' '
        temp += 1
    x = temp
# look for coreq info
elif tokens[x] == 'Corequisite':
    x += 1  # compensate for colon, avoid trying to
       reference out of range index
    temp = x
    temp += 1
    while True:

        # if it's the last word in the sentence cut out
           garbage space and quit
        if tokens[temp][len(tokens[temp])-1] == '.':
            coreq[y] = coreq[y][:-1]
            break

        coreq[y] += tokens[temp] + ' '
        temp += 1
    x = temp
# grab credit information
elif tokens[x] == 'Credit':
    x += 1  # compensate for colon, avoid trying to
       reference out of range index
    temp = x
    temp += 1
    while tokens[temp] != '.':
        cred[y] += tokens[temp] + ' '
        temp += 1
    x = temp\
# look for new dist. tags
elif tokens[x] == 'Distribution Requirements':
    x += 1  # compensate for colon, avoid trying to
       reference out of range index
    temp = x
    temp += 1
    while True:
```

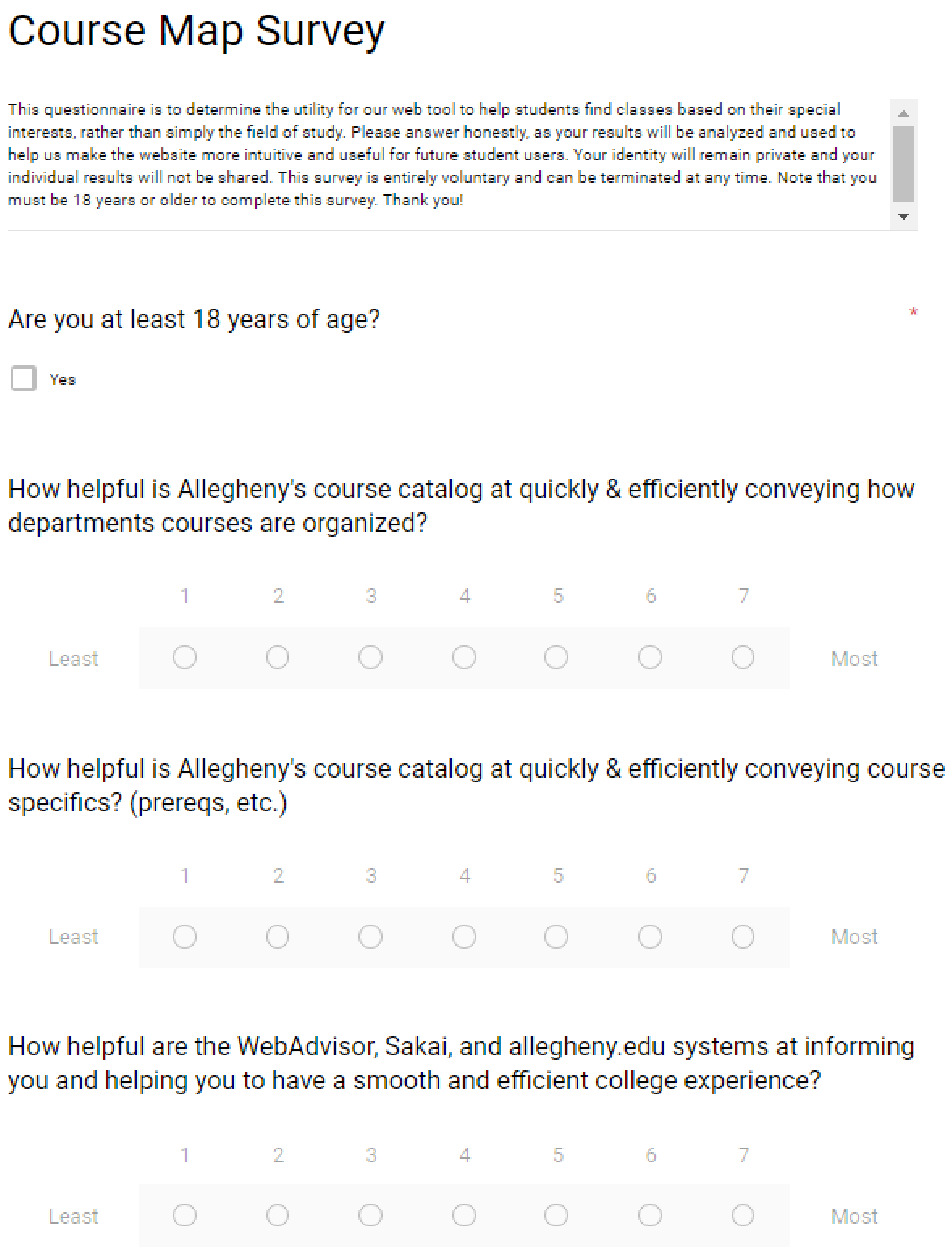
```
                    # if it's the last word in the sentence cut out
                       of the period and break
                    if tokens[temp][len(tokens[temp])-1] == '.':
                        dist[y] = dist[y][:-1]
                        break

                    dist[y] += tokens[temp] + ' '
                    temp += 1
                x = temp

     print "SUMMARIES PARSED"
     return prereq, coreq, cred, dist


def populate_db(idTemp,deptTemp, numberTemp, titleTemp,
   prerequisitesTemp,corequisiteTemp,creditsTemp,distributionTemp):
    conn = lite.connect('testDB.db')
    cur = conn.cursor()
    cur.execute("DROP TABLE Nodes;")
    cur.execute("CREATE TABLE Nodes"
                "(id TEXT PRIMARY KEY NOT NULL,"
                " dept TEXT,"
                " number INT,"
                " title TEXT,"
                " prerequisites TEXT,"
                " corequisites TEXT,"
                " credits TEXT,"
                " distribution TEXT);")
    for x in range(len(idTemp)):
        record = "\"" + idTemp[x] + "\"," + "\"" + deptTemp[x] + "
           \"," + "\"" + numberTemp[x] + "\"," + "\"" + \
                 titleTemp[x] + "\"," + "\"" + prerequisitesTemp[x]
                    + "\"," + "\"" + \
                 corequisiteTemp[x] + "\"," + "\"" + creditsTemp[x]
                    + "\"," + "\"" + distributionTemp[x] + "\""
        #print record
        cur.execute("INSERT INTO Nodes VALUES(" + record + ');')
    conn.commit()
    conn.close()

#breaks courses up into specific details
def getCourseDetails(tempCourses):
    course_department, course_number, course_id, course_title = [],
       [], [], []
    for x in range(len(tempCourses)):
        temp_course = tempCourses[x]
        split_course = temp_course.split(' ')
        course_department.append(split_course[0])
        course_number.append(split_course[1])
        course_id.append(split_course[0] + str(split_course[1]))
        course_title.append(' '.join(split_course[2:len(split_course
            )]))
    return course_id, course_department, course_number, course_title


def textMine():
    # uncomment if you need to clean up converted text document
       still, creates 'workingVersion.txt'
    # clean_text_file()

    corp = open('workingVersion.txt')
    line_txt = []
    while True:
        line = corp.readline()
        if line == "":
            break
        else:
            line_txt.append(line)
    corp.close()
    # course names
    courses = []



    # grab all the course titles, NOTE: this also grabs course
       titles that are not followed by summaries
    courses = get_courses(line_txt, dept_acronyms)
    courses = remove_duplicates(courses)  # remove duplicate courses
        and calls other things
    courses = sorted(courses)  # sort the courses, this results in
       department clusters sorted by course number

    # rejoin and cluster text by paragraph
    corpus = ''.join(line_txt)
    paragraphs = corpus.split('\n\n')
    paragraphs = remove_course_lists(paragraphs, courses)  # remove
       paragraphs with more than one course in them
    summaries, courses = get_summaries(paragraphs, courses)



    # get the seperate department code names and class numbers and
       create course id

    course_id, course_department, course_number, course_title =
       getCourseDetails(courses)


    for x in range(len(course_title)):
        course_title[x] = course_title[x][:-1]
    prerequisites , corequisite , credits , distribution = [] , [] , [] ,
       []
    prerequisites , corequisite , credits , distribution =
       parse_summaries(summaries , prerequisites , corequisite ,
       credits ,
                                                                          distribu
                                                                             )

    # shorten and clean summaries for information I already pulled
       out and for \t values
    for x in range(len(summaries)):
        # only keep what comes before all the information I already
           mined
        summaries[x] = summaries[x].split("Prerequisite:", 1)[0]
        summaries[x] = summaries[x].split("Prerequisites:", 1)[0]
        summaries[x] = summaries[x].split("Corequisite:", 1)[0]
        summaries[x] = summaries[x].split("Credit:", 1)[0]
        summaries[x] = summaries[x].split("Distribution Requirements
           :", 1)[0]
        # make sure they dont screw up my tsv format
        summaries[x] = summaries[x].replace('\t', ' ')

    # I dont remember why this is here
    for x in range(len(prerequisites)):
        prerequisites[x] = prerequisites[x].translate(None, string.
           punctuation)

    # attaching department IDs for color coding within R
    dept_id = []
    count = 1
    dept_id.append(count)
    for x in range(1, len(course_id)):
        if course_department[x] != course_department[x - 1]:  # if
           its a new department
            count += 1
        dept_id.append(count)
    for x in range(len(summaries)):
        summaries[x] = ' '.join(summaries[x].split('\n'))

    return courses, course_id, course_department, dept_id,
       course_number, course_title, prerequisites, corequisite, \
           credits, distribution, summaries

# populates the database.tsv file with course information
def popDBTSV(t_course_id, t_course_department, t_course_number,
   t_course_title, t_prerequisites,
                 t_corequisite, t_credits, t_distribution,
                    t_summaries):
    db_tsv = open('R/RWork/RPractice/database.tsv', 'w')
    db_tsv.write('id\tdept\tnumber\ttitle\tprerequisites\
       tcorequisites\tcredits\tdistribution\tsummary\n')
    for x in range(len(t_course_id)):
        if t_course_id[x] == '':
            t_course_id[x] = 'NA'
        if t_course_department[x] == '':
            t_course_department[x] = 'NA'
        if t_course_number[x] == '':
            t_course_number[x] = 0
        if t_course_title[x] == '':
            t_course_title[x] = 'NA'
        if t_prerequisites[x] == '':
            t_prerequisites[x] = 'NA'
        if t_corequisite[x] == '':
            t_corequisite[x] = 'NA'
        if t_credits[x] == '':
            t_credits[x] = 'Assumed 4'
        if t_distribution[x] == '':
            t_distribution[x] = 'NA'
        if t_summaries[x] == '':
            t_summaries[x] = 'NA'
        if x != len(t_course_id) - 1:
            db_tsv.write(
                t_course_id[x] + '\t' + t_course_department[x] + '\t
                    ' + t_course_number[x] + '\t' + t_course_title[x]
                + '\t' + t_prerequisites[x] + '\t' + t_corequisite[x
                    ] + '\t' + t_credits[x] + '\t' + t_distribution[x
                    ]
                + '\t' + t_summaries[x] + '\n')
        else:
            db_tsv.write(
                t_course_id[x] + '\t' + t_course_department[x] + '\t
                    ' + t_course_number[x] + '\t' + t_course_title[x]
                + '\t' + t_prerequisites[x] + '\t' + t_corequisite[x
                    ] + '\t' + t_credits[x] + '\t' + t_distribution[x
                    ]
                + '\t' + t_summaries[x] + '\n')

    db_tsv.close()

# populate nodes.tsv file with some course information
def popNodesTSV(t_course_id, t_course_department, t_dept_id,
   t_course_number, t_course_title):
    node_csv = open('R/RWork/RPractice/nodes.tsv', 'w')
    node_csv.write("id\tdept\tdeptNum\tnumber\ttitle\n")
    for x in range(len(t_course_id)):
        if x != len(t_course_id) - 1:
            node_csv.write(
                t_course_id[x] + '\t' + t_course_department[x] + '\t
                    ' + str(t_dept_id[x]) + '\t' + t_course_number[x]
                + '\t' + t_course_title[x] + '\n')
        else:
            node_csv.write(
                t_course_id[x] + '\t' + t_course_department[x] + '\t
                    ' + str(t_dept_id[x]) + '\t' + t_course_number[x]
                + '\t' + t_course_title[x])

    node_csv.close()

# find edge relationships by mining prerequisite information
def getEdges(t_prerequisites, t_course_id):
    t_edges = [[]]
    # need to figure out a less ugly way to do this but it works
    for x in range(len(t_prerequisites)):
        split_prereq = t_prerequisites[x].split(' ')
        for y in range(len(split_prereq) - 1):
            sample = (split_prereq[y] + split_prereq[y + 1]).upper()
                 # stick two together, looking for id matches
            for z in range(len(t_course_id)):
                if sample == t_course_id[z]:
                    t_edges.append([sample, t_course_id[x]])

    t_edges = t_edges[1:]  # remove empty zeroth pair, don't know
       why it's there
    return t_edges

#populate the edges.tsv file with relationships found in previous
   method
def popEdgesTSV(t_edges):
    csv = open('R/RWork/RPractice/edges.tsv', 'w')
    csv.write('from\tto\tweight\n')
    for x in range(len(t_edges)):
        if x == len(t_edges)-1:
            csv.write(t_edges[x][0] + "\t" + t_edges[x][1] + "\t" +
                '5')
        else:
            csv.write(t_edges[x][0] + "\t" + t_edges[x][1] + "\t" +
                '5\n')
    csv.close()

#very similar to text mine, information is more available except for
    prereq stuff
def sheetMine():
    import csv
    courses, course_id, course_department, dept_id, course_number,
       course_title, prerequisites, corequisite, credits, \
    distribution, summaries = [], [], [], [], [], [], [], [], [],
       [], []
    with open('corpus.tsv', 'r') as tsvfile:
        reader = csv.reader(tsvfile, delimiter='\t')
        firstLine = True
        for row in reader:
            if firstLine:
                firstLine = False
                continue
            courseNum = row[0].split('*')[1]
            courseDept = row[0].split('*')[0]
            if courseDept+courseNum in course_id:
                continue
            courses.append(courseDept+courseNum+' '+row[1])
            course_id.append(courseDept+courseNum)
            course_department.append(courseDept)
            course_number.append(courseNum)
            course_title.append(row[1])
            summaries.append(row[4])
            credits.append(row[2])
            distribution.append(row[5]+','+row[6])

    for y in range(len(summaries)):
        prerequisites.append('')
        corequisite.append('')
        if summaries[y] == '':
            summaries[y] = 'NA'
        if summaries[y][len(summaries[y])-1] != '.':
            summaries[y] = summaries[y] + '.'
        tokens = nltk.word_tokenize(summaries[y])
        for x in range(len(tokens)):
            # check for prereq trigger
            if tokens[x] == 'Prerequisites' or tokens[x] == '
               Prerequisite':
                x += 1  # compensate for colon, avoid trying to
                   reference out of range index
                temp = x
                temp += 1
                while True:

                    # if it's the last word in the sentence (theres
                       a period) cut out the garbage space in the
                    # prereq and quit
                    if tokens[temp][len(tokens[temp]) - 1] == '.':
                        prerequisites[y] = prerequisites[y][:-1]
                        break

                    prerequisites[y] += tokens[temp] + ' '
                    temp += 1
                x = temp
            # look for coreq info
            elif tokens[x] == 'Corequisite':
                x += 1  # compensate for colon, avoid trying to
                   reference out of range index
                temp = x
                temp += 1
                while True:

                    # if it's the last word in the sentence cut out
                       garbage space and quit
                    if tokens[temp][len(tokens[temp]) - 1] == '.':
                        corequisite[y] = corequisite[y][:-1]
                        break
                    corequisite[y] += tokens[temp] + ' '
                    temp += 1
                x = temp
    count = 1
    dept_id.append(count)
    for x in range(1, len(course_id)):
        if course_department[x] != course_department[x - 1]:  # if
            its a new department
            count += 1
        dept_id.append(count)
    for x in range(len(summaries)):
        summaries[x] = ' '.join(summaries[x].split('\n'))

    return courses, course_id, course_department, dept_id,
       course_number, course_title, prerequisites, corequisite, \
           credits, distribution, summaries



def main():

    temp = raw_input("Would you like to mine course info from the
       text or from the spread sheet? (text,sheet):  ")
    if temp == "text":
        courses, course_id, course_department, dept_id,
            course_number, course_title, prerequisites, corequisite,
            \
        credits, distribution, summaries = textMine()
    elif temp == "sheet":
        courses, course_id, course_department, dept_id,
            course_number, course_title, prerequisites, corequisite,
            \
        credits, distribution, summaries = sheetMine()
    # debugging output to make sure everything matches up
    print len(course_id)
    print len(course_department)
    print len(dept_id)
    print len(course_number)
    print len(course_title)
    print len(prerequisites)
    print len(corequisite)
    print len(credits)
    print len(distribution)
    print len(summaries)

    popDBTSV(course_id, course_department, course_number,
       course_title, prerequisites, corequisite, credits,
             distribution, summaries)  # create database.tsv
    popNodesTSV(course_id, course_department, dept_id, course_number
       , course_title)  # create nodes.tsv
```

```
edges = getEdges(prerequisites, course_id)  # mine prerequisites
    for relationships and create edges

count = 0
unattachedCoursesPerDepartment = [0]
courseCount = []

popEdgesTSV(edges)  # create edges.tsv

# uncomment if you need to repopulate db with mined information
populate_db(course_id, course_department, course_number,
   course_title, prerequisites, corequisite,
           credits, distribution)




# FINDING NUMBER OF UNLINKED COURSES PER DEPARTMENT
vertices = open('R/RWork/RPractice/nodes.tsv', 'r')
links = open('R/RWork/RPractice/edges.tsv', 'r')
count = 0
vertices.readline()
vList = vertices.read()
vList = vList.split('\t')
links.readline()
lList = links.read()
lList = lList.split('\t')
for x in range(len(lList)):
    lList[x] = lList[x].replace('5\n', '')
print lList
for acro in sorted(dept_acronyms):
    courseCount.append(course_department.count(acro))
for x in range(0, len(course_id)):
    if x > 0 and course_department[x] != course_department[x -
        1]:  # if its a new department
         count += 1
         unattachedCoursesPerDepartment.append(0)
    if course_id[x] not in lList:
         print course_id[x] + " is not in the lList"
         unattachedCoursesPerDepartment[count] += 1


print "Number of Unlinked Courses: "+str(count)
print sorted(dept_acronyms)
print "# of Courses: " + str(courseCount)
print "# of unnattached courses: " + str(
   unattachedCoursesPerDepartment)
proportionUnattached = []
for x in range(len(courseCount)):
    proportionUnattached.append(unattachedCoursesPerDepartment[x
       ]/float(courseCount[x]))
print proportionUnattached
```

```
#################################################################

if __name__ == "__main__":
    main()
```

```
# Title: Script for evaluating survey results from Course Map system
# Author: Dillan Smith
# Date: 4/4/2017
# All rights reserved
import csv
import sqlite3 as lite
with open('data.csv', 'rb') as f:

    data = [[], [], [], [], [], [], [], []]
    reader = csv.reader(f)
    row_num = 0
    for row in reader:
        if row_num == 0:
            header = row

        else:
            for x in range(len(row)):
                if x < 6:
                    data[x].append(int(row[x]))
                else:
                    data[x].append(row[x])
        row_num += 1
    for x in range(len(data)):
        print header[x]
        print data[x]

    # using SQL
    conn = lite.connect('surveyData.db')
    cur = conn.cursor()
    #get global averages
    cur.execute("select AVG( allegheny_specifics), AVG(
       courseMap_specifics), AVG(allegheny_organization), "
                "AVG(courseMap_organization),  AVG(systems), AVG(
                   courseMap_potential) from data;")
    globalAVG = list(cur.fetchall()[0])
    print "Global Averages:
       ##################################################"
    print "Average Course Catalog Specifics Rating: " + str(
       globalAVG[0])
    print "Average Course Map Specifics Rating: " + str(globalAVG
       [1])
    print "Average Course Catalog Organization Rating: " + str(
       globalAVG[2])
    print "Average Course Map Organization Rating: " + str(globalAVG
       [3])
    print "Average Allegheny Systems Rating: " + str(globalAVG[4])
    print "Average Course Map Potential Rating: " + str(globalAVG
       [5])
    print "
       ############################################################"
    # get AVG values grouped by school year
    cur.execute("select AVG(allegheny_specifics), AVG(
       allegheny_organization), "
                "AVG(courseMap_specifics), AVG(
                   courseMap_organization), AVG(systems), AVG(
                   courseMap_potential) "
                "from data where classYear = 'Freshman';")
    freshAVG = list(cur.fetchall()[0])
    cur.execute(
        "select AVG(allegheny_specifics), AVG(allegheny_organization
           ), AVG(courseMap_specifics), "
        "AVG(courseMap_organization), AVG(systems), AVG(
           courseMap_potential) from data where classYear = '
           Sophomore';")
    sophAVG = list(cur.fetchall()[0])
    cur.execute(
        "select AVG(allegheny_specifics), AVG(allegheny_organization
           ), AVG(courseMap_specifics), "
        "AVG(courseMap_organization), AVG(systems), AVG(
           courseMap_potential) from data where classYear = 'Junior
           ';")
    juniorAVG = list(cur.fetchall()[0])
    cur.execute(
        "select AVG(allegheny_specifics), AVG(allegheny_organization
           ), AVG(courseMap_specifics), "
        "AVG(courseMap_organization), AVG(systems), AVG(
           courseMap_potential) from data where classYear = 'Senior
           ';")
    seniorAVG = list(cur.fetchall()[0])
    print "AVG of allegheny_specifics, organization,
       courseMap_specifics, organization, systems, " \
          "and potential by Class Year"
    print freshAVG
    print "Freshmen: "+'         '+str(freshAVG[0])[:5]+'         '+
       str(freshAVG[1])[:5]+'              '+\
          str(freshAVG[2])[:5]+ '             '+str(freshAVG[3])
             [:5]+'           '+str(freshAVG[4])[:5]+\
          '           '+str(freshAVG[5])[:5]
    print "Sophomores: " +'       '+str(sophAVG[0])[:5]+'         '+
       str(sophAVG[1])[:5]+'             '+\
          str(sophAVG[2])[:5]+'             '+str(sophAVG[3])[:5]+'
                      '+str(sophAVG[4])[:5]+\
          '           '+str(sophAVG[5])[:5]
    print "Juniors: " +'         '+ str(juniorAVG[0])[:5]+'
       '+str(juniorAVG[1])[:5]+'              '+\
          str(juniorAVG[2])[:5]+'             '+str(juniorAVG[3])
             [:5]+'           '+str(juniorAVG[4])[:5]+\
          '           '+str(juniorAVG[5])[:5]
    print "Seniors: " +'         '+ str(seniorAVG[0])[:5]+'
```

```
       '+str(seniorAVG[1])[:5]+'              '+\
          str(seniorAVG[2])[:5]+'             '+str(seniorAVG[3])
             [:5]+'          '+str(seniorAVG[4])[:5]+\
          '            '+str(seniorAVG[5])[:5]
    print "
       ########################################################################
       "

    print "Number of Respondents from Each Grade"
    cur.execute("SELECT classYear FROM data WHERE classYear = '
       Freshman';")
    classYear = cur.fetchall()
    print "Number of Freshmen: "+str(len(classYear))
    cur.execute("SELECT classYear FROM data WHERE classYear = '
       Sophomore';")
    classYear = cur.fetchall()
    print "Number of Sophomore: " + str(len(classYear))
    cur.execute("SELECT classYear FROM data WHERE classYear = '
       Junior';")
    classYear = cur.fetchall()
    print "Number of Junior: " + str(len(classYear))
    cur.execute("SELECT classYear FROM data WHERE classYear = '
       Senior';")
    classYear = cur.fetchall()
    print "Number of Senior: " + str(len(classYear))
    print "
       ########################################################################
       "
    print "Averages Based on Gender"
    cur.execute("select AVG(allegheny_specifics), AVG(
       allegheny_organization), AVG(courseMap_specifics), "
                "AVG(courseMap_organization), AVG(systems), AVG(
                   courseMap_potential) FROM data WHERE gender = '
                   Male';")
    maleAVGs = cur.fetchall()
    cur.execute("select AVG(allegheny_specifics), AVG(
       allegheny_organization), AVG(courseMap_specifics), "
                "AVG(courseMap_organization), AVG(systems), AVG(
                   courseMap_potential) FROM data WHERE gender = '
                   Female';")
    femaleAVGs = cur.fetchall()

    print maleAVGs
    print femaleAVGs
```



Course Map Survey

This questionnaire is to determine the utility for our web tool to help students find classes based on their special interests, rather than simply the field of study. Please answer honestly, as your results will be analyzed and used to help us make the website more intuitive and useful for future student users. Your identity will remain private and your individual results will not be shared. This survey is entirely voluntary and can be terminated at any time. Note that you must be 18 years or older to complete this survey. Thank you!

Are you at least 18 years of age? *

☐ Yes

How helpful is Allegheny's course catalog at quickly & efficiently conveying how departments courses are organized?

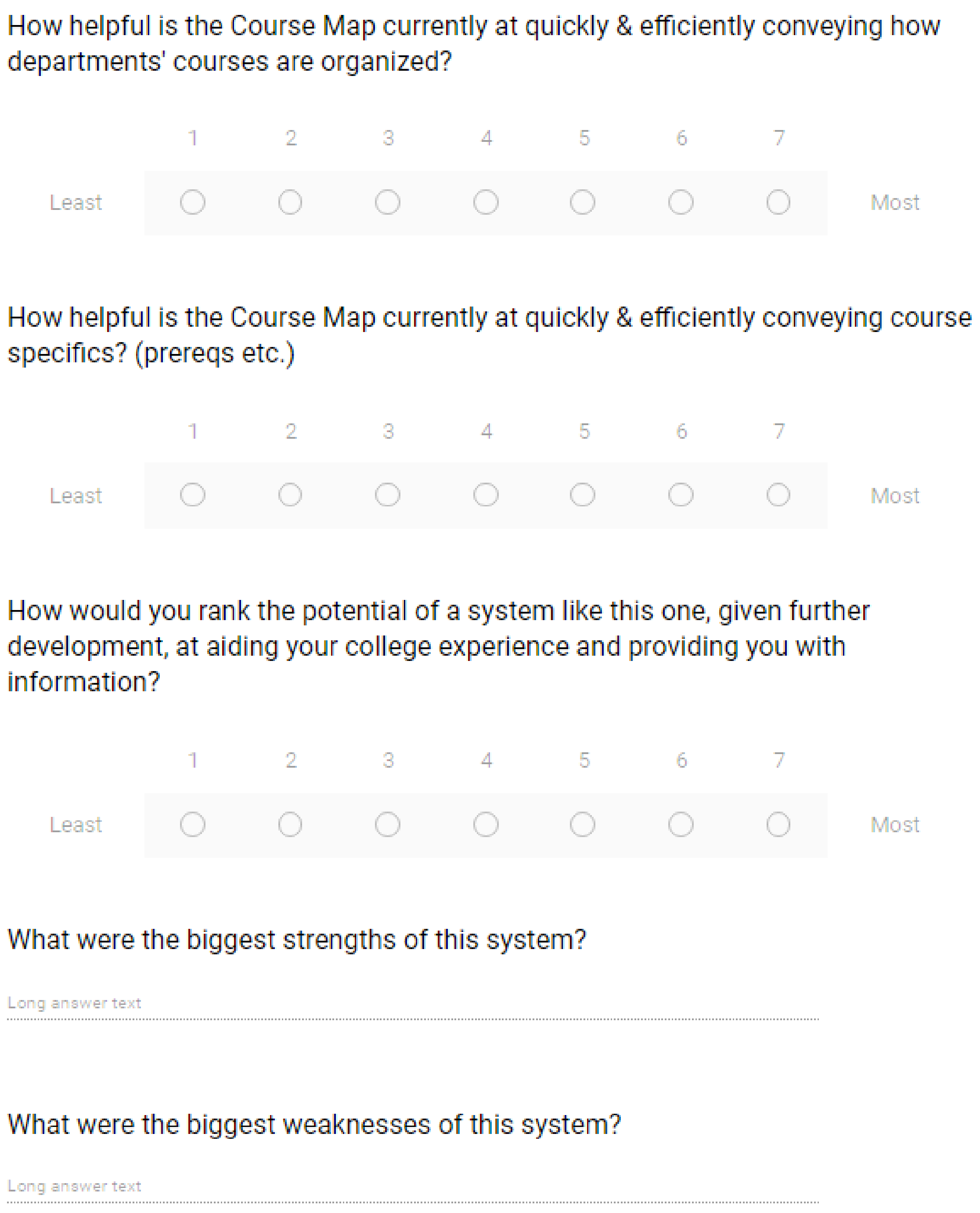
	1	2	3	4	5	6	7	
Least	○	○	○	○	○	○	○	Most

How helpful is Allegheny's course catalog at quickly & efficiently conveying course specifics? (prereqs, etc.)

	1	2	3	4	5	6	7	
Least	○	○	○	○	○	○	○	Most

How helpful are the WebAdvisor, Sakai, and allegheny.edu systems at informing you and helping you to have a smooth and efficient college experience?

	1	2	3	4	5	6	7	
Least	○	○	○	○	○	○	○	Most



How helpful is the Course Map currently at quickly & efficiently conveying how departments' courses are organized?

	1	2	3	4	5	6	7	
Least	◯	◯	◯	◯	◯	◯	◯	Most

How helpful is the Course Map currently at quickly & efficiently conveying course specifics? (prereqs etc.)

	1	2	3	4	5	6	7	
Least	◯	◯	◯	◯	◯	◯	◯	Most

How would you rank the potential of a system like this one, given further development, at aiding your college experience and providing you with information?

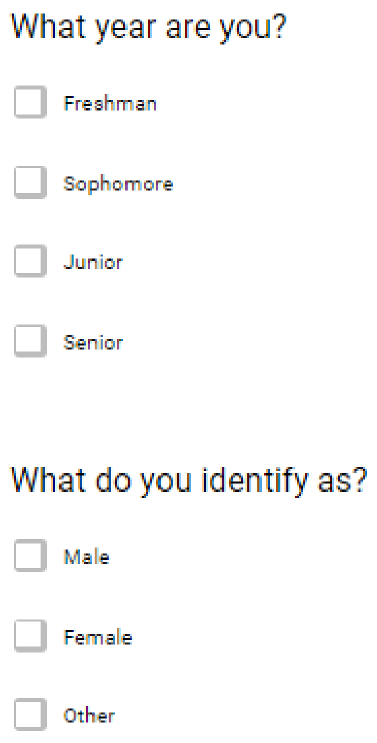
	1	2	3	4	5	6	7	
Least	◯	◯	◯	◯	◯	◯	◯	Most

What were the biggest strengths of this system?

Long answer text

What were the biggest weaknesses of this system?

Long answer text


What year are you?

☐ Freshman

☐ Sophomore

☐ Junior

☐ Senior

What do you identify as?

☐ Male

☐ Female

☐ Other

Appendix B

R Code

```
#Title: server.R portion of Shiny App Course Map system
#Author: Dillan Smith
#Date: 4/4/2017
#All rights reserved

require(visNetwork)
require(shiny)
require(igraph)
function(input, output){
  #import nodes and edges file create in textMiner
  nodes <- read.table("nodes.tsv", sep = "\t", header=T, as.is=T)
  links <- read.table("edges.tsv", sep = "\t", header=T, as.is=T)
  db <- read.delim("database.tsv", sep = "\t", header=T, as.is = T)

  #create interactive graph
  output$network <- renderVisNetwork({


    nodes$shape <- "dot"
    nodes$shadow <- TRUE # Nodes will drop shadow
    nodes$title <- nodes$title # Text on click
    nodes$label <- nodes$id # Node label
    nodes$size <- 12 # Node size
    nodes$borderWidth <- 2 # Node border width
    nodes$color.background <- c("slategrey", "tomato", "gold", "
       blueviolet", "burlywood", "burlywood4",
                                "black", "coral2", "chocolate", "
                                   cadetblue1", "brown", "
                                   aquamarine1",
                               "deepskyblue4", "deeppink4", "
                                  darkslategray4", "darkseagreen",
                                  "darksalmon",
                               "deeppink", "darkorchid4", "khaki1",
                                   "hotpink3", "limegreen", "
                                  maroon4", "orangered1",
                               "orange1", "seagreen", "royalblue",
                                  "royalblue4", "red", "red4", "
                                  purple",
                                "slateblue4", "springgreen", "
                                   springgreen4", "peru")[
                                   nodes$deptNum]
  nodes$color.border <- "black"
  nodes$color.highlight.background <- "orange"
  nodes$color.highlight.border <- "darkred"
  links$arrows <- "to"
  visNetwork(nodes,links)%>%
    visOptions(selectedBy = "dept", nodesIdSelection = TRUE)%>%
    #get the id of the node being clicked on when clicked
    visEvents(select = "function(nodes) {
              Shiny.onInputChange('current_node_id', nodes.nodes);
              ;}")%>%
    #visPhysics(stabilization = FALSE)
    #visIgraphLayout()
    #prevent users from dragging nodes
    visInteraction(dragNodes = FALSE)%>%
    #visEdges(smooth = FALSE)%>%
    #option based on check box in UI
    visIgraphLayout(smooth = input$smooth)#%>%
    #visLayout(hierarchical = input$hierarchy)
    #visLayout(randomSeed = 123)

})

#gets id of node that clicked on
observeEvent(input$current_node_id, {
  visNetworkProxy("network") %>%
    visGetNodes()
  visFocus(visNetworkProxy("network"), input$current_node_id,
     scale = 1,  offset =  list(x = 0, y = -150))
})

#the database created the tsv file from the textMiner
output$table <- DT::renderDataTable(db)

#Output summary of clicked on node
output$nodes_summary <- renderText( {

    paste("Summary:",db[match(input$current_node_id, nodes[,"id"])
       [1], "summary"])

})

#output title of clicked on node
output$nodes_title <- renderText({
  paste("Title: ",db[match(input$current_node_id, nodes[, "id"])
     [1], "title"])
})

#output id of clicked on node
output$nodes_id <- renderText({
    paste("ID:",db[match(input$current_node_id, nodes[, "id"])[1], "
        id"])
  })

  #output$nodes_name <- renderText(paste(db[match(
      input$current_node_id, nodes[, "id"])[1], "id"],
                                              #db[match(
                                                  input$current_node_id,
                                                  nodes[, "id"])[1], "title
                                                  "]
                                              #))








}
```

```
#Title: ui.R of Shiny App Course Map System
#Author: Dillan Smith
#Date 4/4/2017
#All rights reserved
require(shiny)
require(visNetwork)
require(igraph)
#full page
navbarPage("Course Map", id="nav",

  #tutorial tab with embedded pdf
  tabPanel("Tutorial",

           tabPanel("Reference",
                    tags$iframe(style="height:1000px; width:100%;
                        scrolling=yes",
                    src="map.pdf"))
  ),
  #second tab with graph
  tabPanel("Interactive Map",
    sidebarLayout(
      sidebarPanel(
        #checkboxInput("hierarchy", label = "Hierarchichal Layout",
            value = FALSE),
        checkboxInput("smooth", label = "Smooth Edges", value = TRUE
            ),
        textOutput("nodes_id"),
        textOutput('nodes_title'),
        textOutput("nodes_summary"),
        actionButton("getNodes", "Nodes")
      ),
      #the graph
      mainPanel(
        visNetworkOutput("network", height = "1000px")
      )
    )



  ),
  #final tab with the database in it
  tabPanel("Data Explorer",
      fluidRow(
        DT::dataTableOutput("table")
      )



  )



)
```

Bibliography

[1] The condition of education - postsecondary education - programs, courses, and completions - undergraduate retention and graduation rates - indicator may (2016).

[2] Index - 2014 vs 2012.

[3] Muhammad Nur Adilin Mohd Anuardi, Hideyuki Shinohara, and Atsuko K Yamazaki. A pre-study of background color effects on the working memory area of the brain. *Procedia Computer Science*, 96:1172–1178, 2016.

[4] Neeraj Bhanot, Neeraj Bhanot, P Venkateswara Rao, P Venkateswara Rao, SG Deshmukh, and SG Deshmukh. Identifying the perspectives for sustainability enhancement: A text mining approach for a machining process. *Journal of Advances in Management Research*, 13(3):244–270, 2016.

[5] Nick Cawthon and Andrew Vande Moere. The effect of aesthetic on the usability of data visualization. In *Information Visualization, 2007. IV'07. 11th International Conference*, pages 637–648. IEEE, 2007.

[6] Hwiman Chung and Xinshu Zhao. Effects of perceived interactivity on web site preference and memory: Role of personal motivation. *Journal of Computer-Mediated Communication*, 10(1):00–00, 2004.

[7] Ton De Jong, Luigi Sarti, and EG-Programm DELTA. *Design and production of multimedia and simulation-based learning material.* Springer, 1994.

[8] Virginia N Gordon and George E Steele. *The undecided college student: An academic and career advising challenge.* Charles C Thomas Publisher, 2015.

[9] Anna Grubert, Nancy B Carlisle, and Martin Eimer. The control of single-color and multiple-color visual search by attentional templates in working memory and in long-term memory. *Journal of Cognitive Neuroscience*, 2016.

[10] Emma Henderson, Gaëlle Vallée-Tourangeau, and Frédéric Vallée-Tourangeau. Touchy thinking: interactivity improves planning. 2016.

[11] Vichuda Nui Kettanurak, K Ramamurthy, and William D Haseman. User attitude as a mediator of learning performance improvement in an interactive multimedia environment: an empirical investigation of the degree of interactivity and

learning styles. *International Journal of Human-Computer Studies*, 54(4):541–583, 2001.

[12] Christof Kuhbandner and Reinhard Pekrun. Joint effects of emotion and color on memory. *Emotion*, 13(3):375, 2013.

[13] Filipe R Lucini, Flavio S Fogliatto, Giovani JC da Silveira, Jeruza Neyeloff, Michel J Anzanello, Ricardo de S Kuchenbecker, and Beatriz D Schaan. Text mining approach to predict hospital admissions using early medical records from the emergency department. *International Journal of Medical Informatics*, 2017.

[14] S Monisha, R Saranya, and K Vijay. Gauzy knowledge sharing in conspiring environment using text mining. *Indian Journal of Science and Technology*, 9(27), 2016.

[15] Candice C Morey, Yongqi Cong, Yixia Zheng, Mindi Price, and Richard D Morey. The color-sharing bonus: Roles of perceptual organization and attentive processes in visual working memory. *Archives of Scientific Psychology*, 3(1):18, 2015.

[16] Rod Sims. Interactivity: A forgotten art? *Computers in Human Behavior*, 13(2):157–180, 1997.

[17] Lorenzo Sommaruga and Nadia Catenazzi. Curriculum visualization in 3d. In *Proceedings of the twelfth international conference on 3D web technology*, pages 177–180. ACM, 2007.

[18] Chih-Hsuan Wei, Robert Leaman, and Zhiyong Lu. Beyond accuracy: creating interoperable and scalable text-mining web services. *Bioinformatics*, 32(12):1907–1910, 2016.

[19] Felix A Wichmann, Lindsay T Sharpe, and Karl R Gegenfurtner. The contributions of color to recognition memory for natural scenes. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 28(3):509, 2002.

[20] Geoffrey F Woodman, Nancy B Carlisle, and Robert MG Reinhart. Where do we store the memory representations that guide attention? *Journal of Vision*, 13(3):1–1, 2013.

[21] Qian Xu and S Shyam Sundar. Interactivity and memory: Information processing of interactive versus non-interactive content. *Computers in Human Behavior*, 63:620–629, 2016.

[22] Ron Zucker. Vicurrias: a curriculum visualization tool for faculty, advisors, and students. *Journal of Computing Sciences in Colleges*, 25(2):138–145, 2009.