Volume 10 Article 6

2010

An Automatic Dialog System for Student Advising

Brian McMahan
Minnesota State University, Mankato

An Automatic Dialog System for Student Advising

Brian McMahan (Department of Computer Science)
Dr. Rebecca Bates, Faculty Member, (Department of Computer Science)
Minnesota State University, Mankato

Abstract

Automatic dialog systems are an implementation of natural language processing theory with the goal of allowing the use of natural sentences to communicate with a computer system. The general purpose of this project was to design and implement an automatic dialog system for augmenting university student advising. Student advising is a relatively narrow domain of possible questions and responses. The automatic dialog system focused on prescriptive advising rather than developmental advising to further narrow the domain to scheduling and registration matters. A student advisor was interviewed and recorded during a mock advising session in order to model the interaction between students and their advisors. The phrases and advising information have been encoded using Artificial Intelligence Markup Language (AIML) and the dialog system has been implemented in the programming language Python. Future work includes expanding the database to include information directly from the Minnesota State University, Mankato student registration system as well as to implement a spoken language interface.

Introduction

Advising is one of the most important facets of a university's service to its students. Through this service, a student is aided in both a prescriptive and developmental manner. The developmental portion of advising deals with the academic, professional, and personal development of the students while the prescriptive deals with such things as course selection, requirements for graduation, and academic planning. The prescriptive portion is inherently the simpler of the two components of advising, but can sometimes take away from the development piece in a one-on-one advising session.

To allow advising sessions to be more centered on the development of the student, a supplemental advising dialog system (ADS) was developed. This project is concerned with the aspects of student advising that are rule-based, similar to the work in [6]. Further, prescriptive advising was approached through interactive dialog because it offered the student a user-friendly interface. The ADS enables a student to have most of their classes decided upon prior to an advising session which means more time for developmental advising, as seen in [2].

The goal of the ADS is to make advising easier and more efficient for both the student and the advisor. With the class schedule chosen for the semester, the student would then be able to print it and bring it to the advising session. Future modifications would allow the advisor to be able to log in and access the prospective schedule for approval. This would allow for quick, paperless access, streamlining the advising process.

This paper describes the background and purpose of an advising dialog system for student advising, and the methodology used to develop and implement it, including background information about the technical aspects of the project. It concludes with an analysis of the project at this point and discusses the future goals for the ADS as well as upcoming plans for implementation of new features.

Problem Background

Academic scheduling is complex because of the many permutations allowed for the typical undergraduate student, which is why the topic typically takes up more time than it should in an advising session. Students must be able to balance their necessary core requirements for their major while satisfying the general education requirements and choosing electives of interest to them. There is also a certain order in which the required classes should be taken due to pre-requisite requirements in upper-level classes. Poor academic planning can result in additional costly semesters in school. Further, as students enter the university, they must follow the academic bulletin of that year, which lists the requirements for general education and for their major. Students can choose to switch their bulletin in subsequent years if the requirements change and they wish to graduate under the new requirements. With students from at least four different entering years, an advisor can expect to work out of several different bulletins.

These factors play into a complex system that gives rise to the difficulties a student faces in selecting courses. Even more so now, with budget cuts affecting the number of times a class can be offered, students must plan out their schedules far in advance. If they do not plan to take classes ahead of time, their graduation date could potentially be pushed back due to core major requirements that come in sequences. This makes the motivation for any automated advising system, including [2] and [6], dual-focused: to supplement student advising, allowing for more developmental advising, and to make it easier for students to efficiently schedule their required classes over their academic career.

While these complexities can be handled by an expert system, the other challenge to this project is the task of interacting with the student through natural dialog. The field of natural language processing is a complicated one. In this field, there are two main components to language: the syntax and the semantics. Syntax is the structure of the sentence and how the words relate to each other. The semantics of language is the meaning of words and sentences. The complexity of the two is illustrated by the following sentences: “I want to sleep”, “I want to go to sleep”, “I want to sleep now”, “I would like to sleep”, “I would love to fall asleep”, “I really want to go to bed”, “I need to fall asleep”. For any normal person it is easy to understand that the sentences are relatively similar semantically. However, because of the syntactical differences between all of them, it is harder for a machine to make this distinction [5].

Purpose

The ADS system designed in this work addresses all of these complexities. The main goal of using a dialog system, despite the intricacies, is to interact with the student in a natural manner. Through conversation, the system will be able to ask questions and gain insights into the student's academic desires. These desires become known through mining conversational data and become the input for making the decisions about scheduling suggestions. This is important intuitively because a student does not mechanically choose classes based upon what is required. There is thought and desire put into the schedule selection process that cannot be ignored.

The ADS was designed so that any student of any academic class at Minnesota State University, Mankato can interact and gain something from it. For example, a senior with a well-defined plan for graduation can just use it to plan course times. In this case, the ADS will engage in very little conversation mining. However, if the student is a freshman or sophomore with a more variable schedule, the ADS will engage in a conversation to gain as much information about the student's academic scheduling interests as possible.

The main goal of the ADS is to get the necessary information in a simple but comprehensive manner that is satisfying to the student and advisor in regards to time, ease of use, and accuracy. Our approach can be broken down into two sub-areas that are feasibly addressed: natural language processing and an expert system. The expert system is the easier of the two, requiring only domain knowledge of student advising and a framework for representing it, while the first area requires much more work to maintain ease of use within the natural dialog.

Technical Background & Methodology

The implementation methodology for the ADS for student advising concentrates on the expert system and the natural language interface. There is also a passive knowledge bank in which the natural language interface stores information for the expert system to use to make decisions. A graphical representation of the conceptual understanding of this approach can be seen in Figure 1.

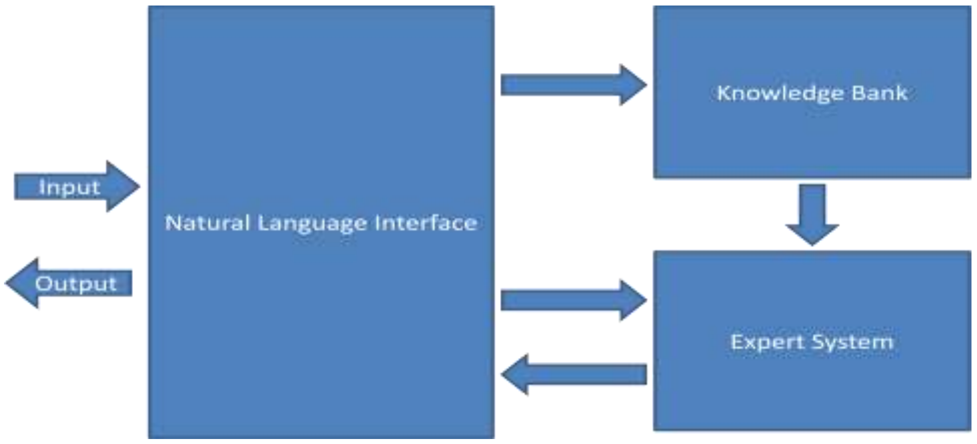


Figure 1. Spoken language dialog system approach: Data flows between the three typical components of a system: a natural language interface, expert system, and knowledge bank.

The expert system was approached as a rule-based decision engine, i.e., static and fast rules that decide what courses to suggest based on knowledge bank information. Most of the rules are inherited from the academic bulletin the student is using, but we chose to include more rules based upon possible information that can be derived from the conversation with the student. Other work in automated student advising systems has the same approach [6].

The natural language interface has several major steps involved in it: dialog management, a user interface, and enhancing the dialog content. In order for the project to supplement student advising effectively, the dialog needs to be natural and accurate. In the field of natural language processing, this is a difficult task because sentences can vary syntactically while staying the same semantically. As shown earlier, there may be many sentences that mean the same thing. Many methods approach this problem by using statistical data and probabilities of words and sentence structure to predicting semantic content. However, this requires a large data corpus to formulate accurate statistics.

Another approach to the syntax versus meaning problem is through rule-based pattern matching. This approach matches phrases or sentences and explicitly states responses for the phrases or sentences. This approach has quick results because patterns and responses can be created during system development and easily modified if other patterns become obvious with system use.

A commonly used pattern matching implementation is the Artificial Intelligence Markup Language (AIML) [8, 9, 10]. AIML was developed by Dr. Richard Wallace and is the basis for ALICE, which won the Loebner prize in 2000, 2001, and 2004. The language stores information in an XML-style document as input patterns, a topic category, and the appropriate response. However, AIML is just a passive language. It is a way to store possible input and output matches but it requires an interpreter to load the AIML files and store the patterns for quick access. The most stable and basic version of the AIML interpreter [7] was implemented in the Python programming language [4], which is why it was chosen for developing this project.

Although AIML has its benefits, there is a downfall to the phrase matching in that it requires the possible input sentences and their responses to be explicitly coded. Further, because the possible input sentences are explicitly coded, it takes an exact match to trigger it. If a user makes a spelling error or uses a different syntax than what is expected, then it would fail because of this need for exact matching [5].

This inherent weakness in AIML was addressed in this work by designing a state manager to control the conversation flow. The state manager maintained a conversation state at all times. For any topic of conversation, there are only one or two sentences that could be relevant during it. Also, the ADS was designed to ask only yes/no type questions in order to minimize the amount of indeterminacy that could arise from possible answers. To ensure that the student could still opt out of any part of the conversation, there are also phrases that are not dependent on the state. For the most part, all conversation is controlled by the state manager to insure that enough information is obtained for the expert system to make decisions about course recommendations.

After preparing the dialog management system, the next step in development was to choose a method for interfacing with a user. While there are many avenues for Python graphical user interfaces (GUIs), the Pyjamas Python web tool-kit was chosen because it offered a cross-platform GUI that could be accessed from any browser [3]. Pyjamas allows a GUI to be programmed in Python but then it converts it to JavaScript, which is a basic but powerful web scripting language. Pyjamas GUIs are able to change the structure of the user interface if a situation arises during a conversation where it needs to allow more or fewer user options. This user interface also has the capability to communicate over a protocol called JSON-RPC [1] to a back-end server. This enabled a light-weight client side application with the dialog management and expert system computation on the back-end. The two communicated over the JSON-RPC protocol, as can be seen in Figure 2. Once the data is input to the back end, it flows to the state manager that interacts with the AIML. This is shown in Figure 3. The result was a simple and easy to use dialog system that could be used from any browser. A screen shot of the system's GUI is shown in Figure 4.

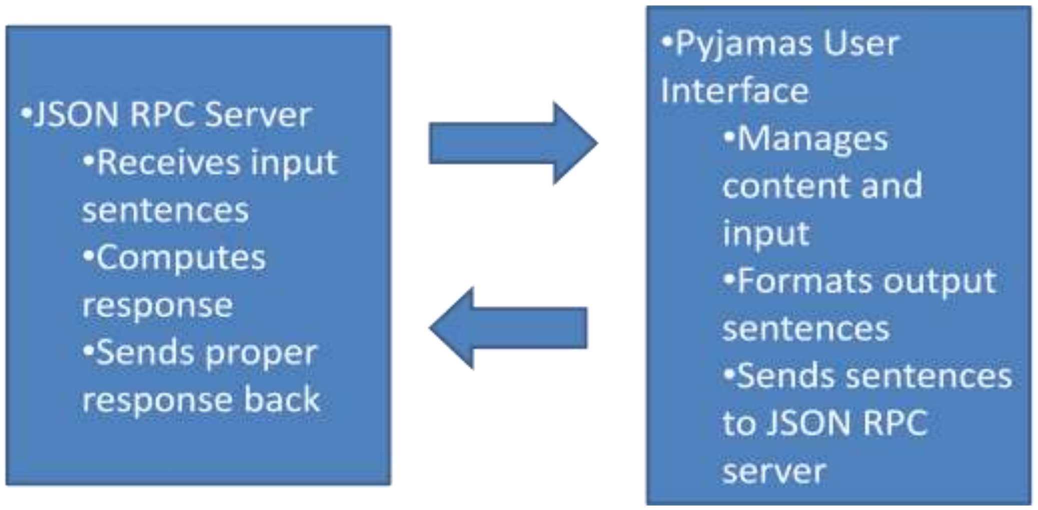

•JSON RPC Server
•Receives input sentences
•Computes response
•Sends proper response back
•Pyjamas User Interface
•Manages content and input
•Formats output sentences
•Sends sentences to JSON RPC server

Figure 2. A graphical representation of the front-end Pyjamas user interface communicating with the back-end server.

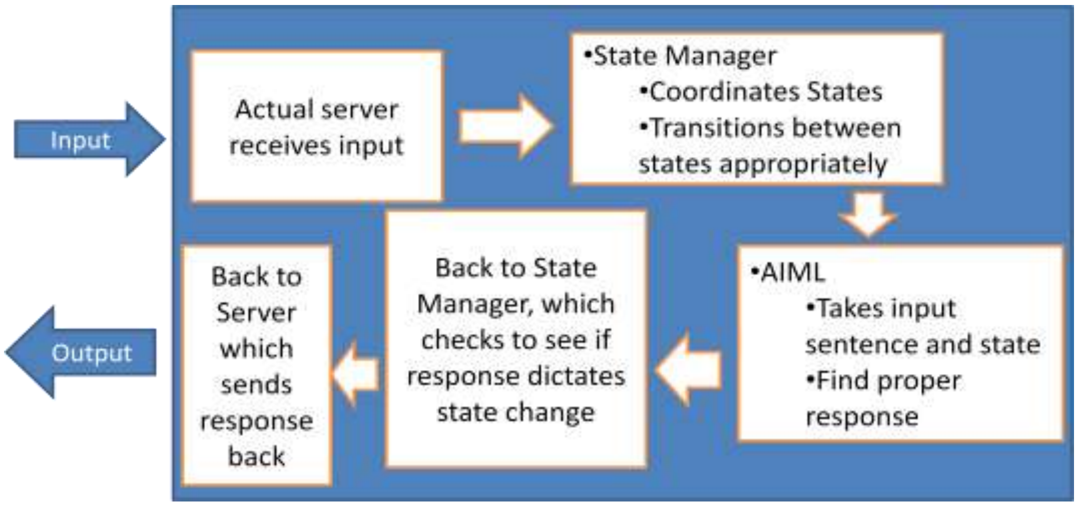
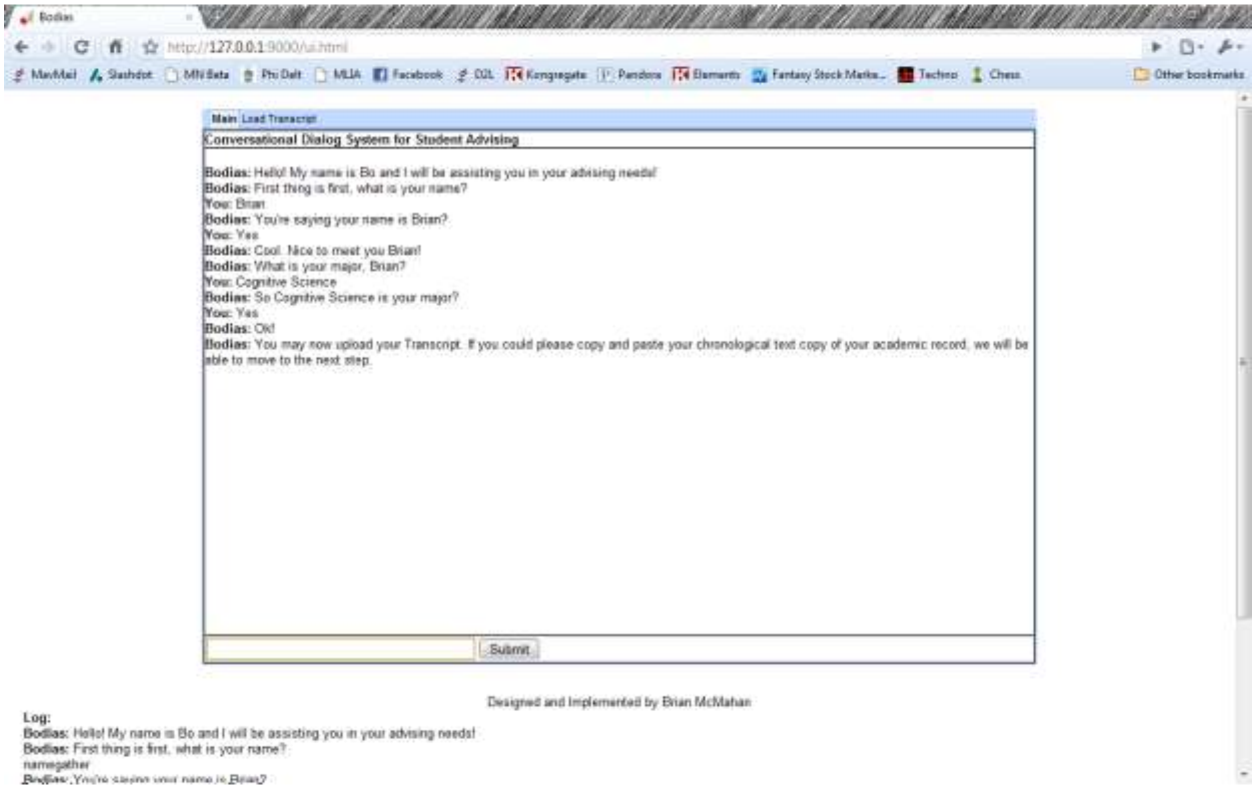

Input
Actual server receives input
•State Manager
•Coordinates States
•Transitions between states appropriately
•AIML
•Takes input sentence and state
•Find proper response
Back to State Manager, which checks to see if response dictates state change
Back to Server which sends response back
Output

Figure 3. A graphical representation of the back-end server receiving an input, the path the input follows, and the subsequent output.



Conversational Dialog System for Student Advising

Bodias: Hello! My name is Bo and I will be assisting you in your advising needs!
Bodias: First thing is first, what is your name?
You: Brian
Bodias: You're saying your name is Brian?
You: Yes
Bodias: Cool. Nice to meet you Brian!
Bodias: What is your major, Brian?
You: Cognitive Science
Bodias: So Cognitive Science is your major?
You: Yes
Bodias: Ok!
Bodias: You may now upload your Transcript. If you could please copy and paste your chronological text copy of your academic record, we will be able to move to the next step.

Designed and Implemented by Brian McMahan

Log:
Bodias: Hello! My name is Bo and I will be assisting you in your advising needs!
Bodias: First thing is first, what is your name?
namegather
Bodias: You're saying your name is Brian?

Figure 4. A snapshot of the user interface during the development of the dialog.

Finally, to both make the dialog content seem more natural and closer to a model advising interaction, Level 1 Human Subject approval was obtained through the Institutional Review Board (IRB) for the purpose of performing and recording mock interviews with academic advisers. A mock advising and interview session was held with a College of Science, Engineering, and Technology Academic Advisor. The recorded session was then transcribed and the sentences and dialog flow incorporated into the ADS to give the perception of a more natural conversation and human-like sentences. A sample of the questions and scenarios used can be seen in Appendix A. Several heuristics were also taken from the transcribed conversation and embedded into the expert system to better imitate an advising session.

Assessment and Analysis

During the development of this project, the largest challenge was to get a working implementation that operated seamlessly. This challenge arose from the front and back end communication. The JSON-RPC and Pyjamas required much fine-tuning in order to work without problems. However, the time consumed to get the two working together proved time well spent as it allows for streamlined communication and more efficient future development.

The system currently uses files to save and retrieve past information, but progress is being made towards incorporating a MySQL database so that students could use the ADS system over the course of their academic careers. The ADS would keep track of their progress from semester to semester. Further, it would allow for quick storage of all the potential classes for the following semester as well as the requirements listed in the current academic bulletins. Until this happens, the current storage system, while functional, is inefficient and wastes computational resources.

The largest critique of the system is the inflexible nature of AIML. The process of explicitly coding the possible sentences and responses is time consuming and painstaking, and after it is coded, there is always the possibility of an input not being valid. For example, if a user says “yeah” instead of “yes” and there are not measures in place to allow for this, then the system won’t recognize “yeah” as a form of “yes”.

Summary and Future Goals

Currently, the system is an off-line, browser-based dialog system that talks through JSON-RPC protocol to a back-end server. This back-end server uses the state manager to reference AIML for a proper response to a question asked or initiating the next topic of discussion. The dialog system asks for a user’s academic history in the form of an academic transcript. This is stored in the knowledge bank and is referenced by the expert system when making academic scheduling decisions.

After the project is integrated into an online website where it can be accessed from anywhere, it will undergo development to allow MySQL databases to be incorporated in various ways. All course bulletin information (core major class requirements, general education requirements, and elective requirements), user profiles and their progress, and the total list of all possible classes will be stored in the MySQL databases. This will eliminate the need for repetitive loading as well as giving faster access to the information. When this framework is developed, the dialog system (the AIML and state manager) can be further developed to create a dynamic conversation that will ask questions relevant to its user. The information that is gained during the dynamic conversation will be fed into an expert system that will use the domain knowledge of scheduling to develop a schedule that fits the student’s needs. It will then be tested on both a wide variety of students and academic advisers who will be asked to fill out a survey to rate the experience of the ADS so that improvements can be made.

An advising dialog system designed to supplement student advising raises challenges in both formulating an expert system and developing natural language processing techniques. However, much progress has been made and the work is continuing. The next stages of development will allow for students and advisers to use it to supplement advising, hopefully making the entire process much more efficient.

References

[1] *JSON-RPC*. Web. March, 2010. <http://json-rpc.org/wiki/python-json-rpc/>

[2] Murray, W. S. and Le Blanc, L. A. 1995. A decision support system for academic advising. In *Proceedings of the 1995 ACM Symposium on Applied Computing* (Nashville, Tennessee, United States, February 26 - 28, 1995). K. M. George, J. Carroll, and D. Oppenheim, Eds. SAC '95. ACM, New York, NY, 22-26. DOI= http://doi.acm.org/10.1145/315891.315897

[3] *Pyjamas: write your own AJAX Framework*. Web. March, 2010. <http://pyjs.org/>

[4] *Python Programming Language -- Official Website*. Python Software Foundation, 1990. Web. March, 2010. <http://python.org/>

[5] Schumaker, R. P., Ginsburg, M., Chen, H., and Liu, Y. 2007. "An evaluation of the chat and knowledge delivery components of a low-level dialog system: The AZ-ALICE experiment." *Decision Support System* 42, 4 (Jan. 2007), 2236-2246. DOI= http://dx.doi.org.ezproxy.mnsu.edu/10.1016/j.dss.2006.07.001

[6] Siegfried, R. M., Wittenstein, A. M., and Sharma, T. 2003. An automated advising system for course selection and scheduling. *Comput. Small Coll.* 18, 3 (Feb. 2003), 17-25.

[7] Stratton, Cort. *PyAIML (a.k.a. Program Y). A Python AIML Interpreter, 2003.* Web. March, 2010. <http://pyaiml.sourceforge.net/>

[8] Wallace, Richard. *ALICEBOT*. Alice A.I. Foundation. Web. March, 2010. <http://alicebot.blogspot.com/>

[9] Wallace, R.S. The Anatomy of A.L.I.C.E. in *A.L.I.C.E. Artificial Intelligence Foundation, Inc.*, 2004.

[10] Wallace, R.S. The Elements of AIML Style, 2003.

Appendix A

Student Advising Interview

I plan to ask participants the following questions. Anything to be said directly to the interviewee is shown in italics. Possible sub-prompts for the core questions are in the bulleted list following the question. While we have multiple scenarios, no more than three will be asked of each advisor.

Interviewer Script:

> *"Thank you for taking the time to talk with me today. I will be using this information to model interaction between advisors and students which will aid in the creation of an automatic dialog system intended for student advising. In this interview, I will establish different student roles and I would like you to best address the questions and issues with responses as natural as possible. You may stop this interview at any time, or choose not to answer one or more of the questions. All of your responses will be kept confidential. Is it OK with you if we tape this interview? The recorded responses will be transcribed and used in the automatic dialog system. Do you have any questions before we get started?"* [Wait a moment for any questions.]
>
> *"We will cover up to three scenarios. After describing the scenario, I will give you the opportunity to ask me questions or initiate the session as you normally would. I will respond according to the character and will also ask questions. If you prefer, I can start with an opening question after I describe the character. Do you have a preference?"* [Let the interviewee decide on the approach.]

1. *In this scenario, I will be a student who is a freshman trying to schedule classes for Spring semester. The major isn't specific, but if you feel the need to speak to a specific major please do so.*
    a. I'm not sure what classes to take. I took a couple general education classes last semester. Should I just continue to take general education classes?
        i. Well, I was thinking about taking two lab classes. Is that a good idea?
        ii. Should I get started on my prerequisites?
        iii. I was thinking about getting a part-time or full-time job so I would have more money. Is this a good idea?

2. *In this scenario, I will be a student who is a sophomore that had a pretty bad freshman year. My GPA is low and I need a more balanced semester, however I want to take a class load that is obviously too hard with a writing intensive course, an upper level math course, and a physic lab course.*
    a. Is this schedule going to be good for me?
        i. What kind of classes should I take instead?
        ii. Should I be worried about applying to my major soon?
            1. What steps should I be taking to make sure I get in?
            2. Why is it so important?
3. *In this scenario, I will be a student who needs to be encouraged to try more things, including in math and science. I have abilities but low self-confidence. In my first semester, I took a balanced load and surprised myself by getting a 3.8. I'm afraid of taking harder courses.*
    a. I'm not sure what classes to take. I did pretty well last year, but I think it might have just been because they were easy classes
        i. Can't I just keep taking easy gen ed classes and do my major classes later?

4. *In this scenario, I will be a pre-engineering major who does not have the pre-requisites to take calculus I yet but really wants to be in classes with my friends.*
    a. I really want to take this calculus. Is there any way I can take it?
        i. Are you sure there is no way we can wave the prerequisites?
            1. What if I take the prerequisite at the same time as calculus?
        ii. What should I take instead?

5. *In this scenario, I will be a student who did really well on the ACT but have never gotten good grades. I don't know what I want to major in but I really like video games.*
    a. I'm not sure really what I want to go into. What do you think I should take? I don't really like to work hard, though. Everything is kind of boring.
    b. I really don't want any morning or Friday classes, is there a way I can avoid them?

Brian McMahan is a senior majoring in Cognitive Science with a focus on Computer Science and a minor in Mathematics at Minnesota State University, Mankato. He was born in Brainerd, MN and graduated from Pine River-Backus High School in 2006. During his enrollment at MSU, Mankato, Brian presented at the Undergraduate Research Conference twice, traveled to Johns Hopkins University for a computational linguistics summer school, and involved himself in Minnesota State Student Senate, Campus Kitchen Project, Phi Delta Theta Fraternity, Interfraternity Council, and several other registered student organizations. Brian is also a part of the McNair Achievement Program where he has performed over 100 hours of research. Through the McNair Achievement Program, Brian also presented his research at the University of California, Berkeley. After graduation in May, 2011 he plans to attend graduate school to attain a PhD in Computer Science or Computational Neuroscience.

Dr. Rebecca Bates is a professor in the Department of Computer Science. Her PhD in Electrical Engineering is from the University of Washington where she worked to develop computer modeling of pronunciation to improve automatic speech recognition. She has a degree in theological studies from Harvard Divinity School, an M.S. in Electrical Engineering from Boston University and a B.S. in Biomedical Engineering from Boston University. Current research projects include investigations of community and connection in STEM education, working on automatic speech recognition in noisy environments, analyzing recorded meetings for style and for areas of importance, and analyzing prosody in adolescents with Williams Syndrome.