WolfTutor

A system to enable peer tutoring built on Slack

Monica Metro
NC State University
3021 F Dorner Circle
Raleigh, NC
mgmetro@ncsu.edu

Zachery DeLong
NC State University
2305 Horizon Hike Ct
Raleigh, NC
zpdelong@ncsu.edu 3rd. author

Zhangqi Zha
NC State University
1800 Vienna Wood Dr
Raleigh, NC
zzha@ncsu.edu

ABSTRACT

Peer tutoring is generally accepted as a great way to help students engage with material. WolfTutor is one system, integrated with the popular chat program Slack, that seeks to facilitate peer-to-peer tutoring interactions by providing a mechanism for registering tutors, scheduling tutoring sessions, and incentivizing students to become tutors. This paper proposes an enhancement to WolfTutor's tutoring matching process to help enable students to better choose what tutors they want to match with.

1. INTRODUCTION

1.1 What is WolfTutor

WolfTutor is a system that seeks to enable students to tutor other students in a course-setting. Wolf Tutor is a slack-based chat app that attempts to connect potential tutors in given subjects with students who need help with a course. The idea here is peer tutoring, not expert tutoring. At first blush, it seems like the app is an app to actually facilitate peer tutoring, which is not entirely accurate. WolfTutor has functions to register tutors and students and to schedule tutoring sessions. It does not have functionality to actually perform the tutoring itself, but is a logistic tool to enable the coordination required to schedule tutoring sessions. WolfTutor is also gamified. It rewards tutors who are highly rated with a points system which can be implemented in a number of different contexts to help incentivize students to tutor other students.

Make the case for why a tool like this needs to exist.

1.2 WolfTutor Use Cases

WolfTutor can be broken down into two types of users: tutors and students. After enrolling in the system, the user's name, email, and phone number will be taken from slack and stored if available.

Students can:

PLDI '13 June 16–19, 2013, Seattle, WA, USA

ACM ISBN 123-4567-24-567/08/06...$15.00

DOI: 10.475/123_4

1. Find a tutor by from a complete list of all available tutors after selecting a subject from an existing list
2. See the reviews for at tutor
3. Book a tutor by choosing one of WolfTutor's defined 30 minute slots created from the availability given by the tutor
4. Review a tutor from their last session (students have until the end of the day to review their tutor)
5. View reservations and reward point balance (points are used to schedule sessions; 100 points are given at signup)

Tutors can:

1. Become a tutor. WolfTutor only allows one major and one degree per tutor, but a tutor can represent multiple subjects. Tutor self-determines points pay rate and availability. Both apply to all subjects.
2. View subjects and availability given to WolfTutor
3. View reward point balance that can be used commercially

1.3 Similar or Comparable Systems

The application of WolfTutor is not actually specific to tutoring entirely. It can reasonably be compared to any scheduling system. For example, the appointment scheduling tool within NC State University's epack system functions similarily. First you pick from an available list of appointment types. Then, you can opt to filter your search further by location, names of possible appointees, time of day, etc. The tool will show available appointment time slots a user can book. This type of scheduling mechanism is commonly used by other services as well e.g. medical facilities with multiple practitioners, personal trainers, etc.

There are no similar applications relevant to automative tutoring matching. Most services only offer a place to find tutors manually, such as Craiglist. Some services take other approaches. The popular tutoring service, Chegg, does not facilitate scheduling or tutoring sessions, but instead lets multiple tutors cater to a student's question.

1.4 Initial Study

Detailed next is the proposed change and the process of user surveying we followed to identify that information. Following that, we will briefly discuss the design of the WolfTutor app In the final sections of the paper, we will discuss our evaluation plan for the proposed enhancement and talk about the plan to accomplish that goal.

2. ENHANCEMENTS

2.1 Pain Points

The original creators of WolfTutor proposed three additional features for future development. The first was to integrate an online platform to conduct tutoring sessions online. The second was to allow both the tutor and the student to sync their session reservation with their calendar, e.g. Google Calendar. Lastly, to update the scheduling algorithm such that a user could edit or delete reservations in addition to being able to create and view them. Our team provided two other features: increasing matching options between tutors and students and allowing students to browse their reservation history.

After careful consideration, three main pain points were decided upon that consisted of the proposed enhancements. Integration of an online platform was discarded because it was not thought to facilitate the quality of the match between tutor and student.

Scheduling and Calendar Sync.

WolfTutor currently only allows users seeking a tutoring session to create and view reservations. In real life, people change their mind or events come up such that a schedule change is in order. The ability to cancel or reschedule reservations would make WolfTutor more applicable to real world scheduling scenarios. In addition, facilitating intergration with commonly used calendars such as Google and IOS Calendar extends this principle of making scheduling easier.

Increasing Matching Options.

In regards to matching with tutors, WolfTutor currently displays a list of tutor's and their ratings based on the student's desired subject. Then, the student may attempt to book a reservation with a tutor they choose. By expanding matching options, a student should be able easily find a higher quality match with a tutor. For example, a student could filter tutors by location by selecting only locations they want to meet up at in a checkbox. Similarily, if a student has a few tutors they prefer, they could select those names from a checkbox such that WolfTutor only displays those tutors. This type of filtering system is often used by medical facilities with multiple practioners and multiple practing sites. It is also used by NC State University's scheduling tool on epack.

History and Recommendations.

WolfTutor does currently allow for student's to review and rate the tutors that they met with previously. However, there is no way to view a student's reservation history past the most recent tutor. WolfTutor also does not provide a way for a student to easily re-book a new reserveration with a tutor they chose previously if they liked the tutor and would like to schedule a reservation with them again. Adding these enhancements would increase ease of use by helping a student distinguish between a competent and non-competent tutor they've had in the past.

2.2 Initial Study

A survey was conducted to determine which enhancement the intended user base (students) prefered the most. The survey was separated into three parts.

Background.

A background section measured how prevalent scheduling was within the daily life of the participants. Most participants admitted having to schedule a meeting within the last year.

Priorities.

The next section revealed which enhancement the participants thought was a priority by asking them to rate the level of important the enhancement was on a scale from 1 (least important) to 5 (most important). To avoid bias, instead of explaining the enhancements out, the survey questions were created around the base point of the enhancement:

- "When scheduling a tutoring meeting, picking the location is very important to me."
- "When scheduling a tutoring meeting, the competency of the tutor is very important to me."
- "When scheduling a tutoring meeting, being able to agree on a time quickly and easily is very important to me."

The question regarding competency scored the highest level of important among the participants collectively. Scheduling and increased matching (by location) followed respectively.

Trade Offs.

The objective of the third section was to validiate the results in the second section by asking the participants which enhancements they would be willing to compromise on in order to get the enhancement they thought was more important. The questions included:

- "When scheduling a tutoring meeting, I am willing to make trade-offs on the competency of the tutor and time if I can specify the location of the meeting."
- "When scheduling a tutoring meeting, I am willing to make trade-offs on location and time if I can specify the competency of my tutor."
- "When scheduling a tutoring meeting, I am willing to make trade-offs in location and competency if I can specify the time of the meeting."

Again, competency scored the highest as the most important enhancement collectively. Scheduling followed in second place and location matching in third.

Other.

An option was given to participants to suggest a new enhancement. The only received response was to integrate the application with Skype.

2.3 Chosen Enhancement

Given the surveying discussed above, it seems clear that what students in our class want is a way to more easily match with competent tutors. To that end, WolfTutor currenly does very little in terms of matching. It provides students with the ratings of the available tutors, which is a good first step, but it does nothing to help organize the available tutors and prioritize them to make searching sutdents' lives easier.

WolfTutor also has a significant amount of data about a student's history with their tutors, but it does fairly little with that information. To help improve matching between students and tutors, we propose to use that historical data during the matching process. To do that, we will implement a suggestion algorithm that will pull out the students' previous interactions and use their positive reviews to push certain tutors higher in the list and their negative reviews to push other tutors lower on the list.

One problem with this approach is that it discourages students from matching with new tutors, because their lack of history will push them further down on the list. In the interests of helpign alleviate this problem, a third dimension should be added to the suggestion algorithm: the interactions from students to tutors. In the proposed model, the ratings of tutors and students and the ratings of the current student to the tutor in quesiton are the major focus of the algorithm, but no consideration is given to the interactions of students with their tutors. This new third dimension should encourage tutors who have already tutored a new tutor in the past (and had a positive experience with them) to match. This should help both encourage students who have used the platform as students to convert to tutors, and it should also help give those students who do convert a leg up in being found by students to tutor.

This will hopefully make it easier for students to match with not only tutors that are well reviewed and rate well in the subjects they are interested in, but also encourage students to build longer-term mentoring relationships with tutors that work well with them.

Next we will discuss the application's architecture in section 3 and the evaluation plan for this idea 5.

3. ARCHITECTURE

3.1 Original Architecture

This section outlines the various components of the system and how they interact with one and another. The detailed architecture is described in figure 3.1.

Broadly speaking, the architecture is divided in three main components: the slack app, the heroku cloud and the mongoDB database. This separation of concerns is a major advantage since database is independently hosted on mlab server and can be accessed from anywhere with valid authorization credentials. The heroku cloud hosts the logic of the slack app, which communicates with the user. The system also implemented the continuous deployment, continuous integration pipeline with Travis CI, which directly pushes the code to heroku cloud once the build passes (also indicated in the readme section of the repository). The test cases was also generated which acts as a sanity check for any commit made to the master branch, before deployment so as to not push broken code on the production server at heroku. Following is the detailed description of each component of the architecture.

Slack App.

Slack App is generated in api.slack.com dashboard. It is currently developed only for one workspace and it is configured to communicate to the heroku server which is hosted at wolftutor.herokuapp.com. Also creating a slack app gives us various authentication tokens like access token and verification token which are required for Slack to authorize that the requests and responses are coming from a valid production server. Slack app also gives us configuration of the bot like its name its icon, etc. This will be visible to the user when the user interacts with the bot. Slack bot is a part of the slack app, where bot is considered like a user (bot-user) in slack terminology. Slack app also allows access to interactive components like dialogs and message menus for the overall user experience to be more rich and interesting.

BotKit.

Botkit is an external library that integrates with Realtime API which can detect patterns in the user queries. The logic to be executed when a particular pattern is detected is written in the slack app (NodeJS code hosted on heroku). For instance, on saying hi user should be given an option to enroll in the system. This is one example of working of the Botkit module.

Code Base.

Code base is the Github repository where we maintain our code. Following the general convention, we made new branch for every new logical feature and also linked issues with particular merge commits. All this activity can be viewed in our repository. We have also used conventional practises of separating database queries and put them all together in model directory. Also all of our interactive components are in distributed in different folders.

Continuous Integration Module.

The master branch of our repository is linked with Travis CI to be pushed to heroku server if the build on travis CI passes. The tests written in mocha acts as a final sanity check before deploying the code on the production server. Any code that is pushed on the master is directly built on Travis CI and deployed on heroku server if build passes.

Heroku Server.

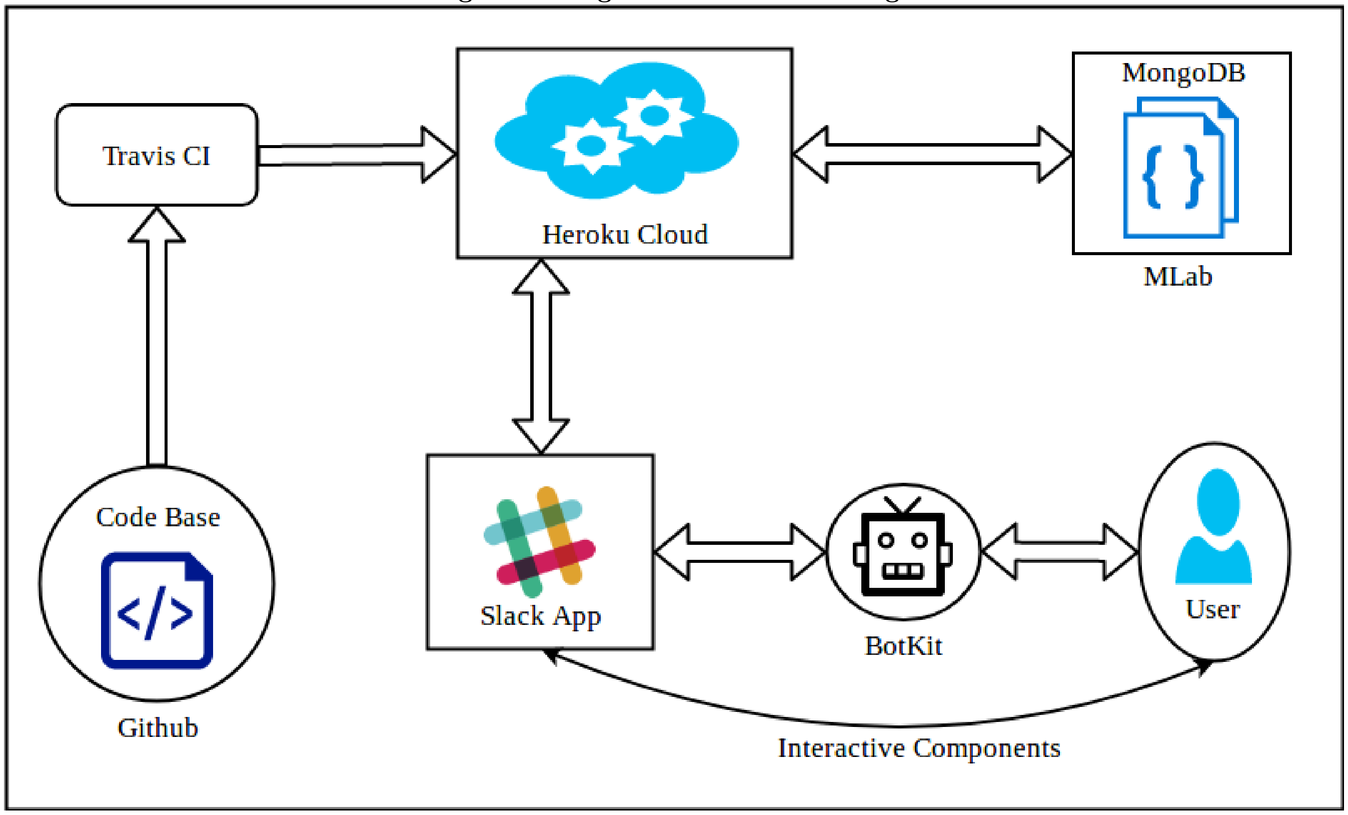
Heroku server is the home of our NodeJS application. We have used single dyno (free version) to host the application. The code pushed on master is deployed here by Travis and we always have the latest code in the production server. Also we have included the Procfile where it is indicated what command to execute to run the application (npm start). This is necessary for heroku to understand the starting point of the application.

Database.

Mlab is a server that hosts our Mongo Database. The advantage of having a remote server for mlab is that everyone can access the same data at simultaneously. In mongo, only a mongo URI is required for accessing the database along with valid user credentials. This makes it very simple to test the application locally as well as run in on the server.

User.

Figure 1: Original Architecture Design



User is anyone who is signed in into the Slack workspace where the slack app is added. He/She can communicate with the app in variety of ways as described in the use-cases section. Also he/she can interact with the app using dialogs and message drop-downs and buttons to give a particular command. See details in use-cases section.

4. PROGRESS

Based on the experience of previous project, the team chose the spiral model of software development methodologies. A spiral model is a form of risk-driven development where the most risky parts of a project are identified early and planned into prototypes that can inform the final product. See the breakdown of the prototypes the team chose in table 1.

This afforded the team the ability to have regular check-ins and to measure progress against the prototypes. The team will convene two to three times a week to do pair programming and to check progress against the prototypes. When combined with regular check-ins with the TAs, the team shall be able to build prototype programs to perform design feature and functionality.

We also used Gitup project boards to create and track our progress. This method featured the ability to send a notification to the team member to remind and collaborate within the team members. Here is the example image.

5. VALIDATION

For this project, the team is interested in focusing on two areas for evaluation: the value of matching and the usability of the system. The value of a match could be evaluated in any number of ways. In a perfect world, there would be a good algorithmic way to evaluate matches as low or high quality and then a multi-year study of peer tutoring would be done where actual students would give feedback on the matches they recieve. Unfortunately, the project's roughly one month timeline makes that kind of evaluation, however valuable, impossible. For that reason, Section 5.1 will focus on algorithmic evaluation almost exclusively.

Usability for this task is also a major concern. One of the hilights of the original app is its ease of use. It is exceedingly easy to register and schedule your first tutoring session in a matter of seconds, not minutes, and that is something we are adamant about not changing, and our goal is to make that even faster with repeated sessions. To that end, Section 5.2 details a strategy for evaluating the changes being proposed in light of the time it takes to schedule tutoring sessions before and after the enhancement.

5.1 Evaluating Matching

The first priority for this enhancement is validating the matches created. Without the ability to run a longitudinal study evaluating actual students, the only option is to generate matches on test data and evaluate the usefulness of the matches themselves.

To that end, this test will require hand labeling a set of randomized students, tutoring sessions, and ratings as "high quality", "low quality" and "reasonable quality". Once that is done, the team will need to run some form of automated tests on that data and validate what percentage of the desired high quality matches are actually generated by the

Table 1: Prototypes and Schedules

Schedules	Prototypes/Work	Descriptions
Week 1	Planning and Pre-survey	Get the original project to work, plan a few changes and enhacenments, do a user survey
Week 2	Project Proposal	Detail the plan and get ready the project proposal
Week 3-1	Prototype 1	Gernerate dummy data, build loading history interface
Week 3-2	Prototype 2	Gernerate dummy data, build suggestion interface
Week 4-1	Prototype 3	Build suggestion algorithm and intergated into final system
Week 4-2	Prototype 4	System evaluation, user testing, update system
Week 5	Presentation	Present the final system
Week 6/7	Final Paper	Draft and finalize the paper

Table 2: Progress Tracking Board

progress_tracking.png

suggestion algorithm. There is actually a reasonable path to do this in the current code base, as current automated tests for the system rely on data being in a local database for testing. While this was originally considered a mistake by the previous maintainers, this actually works as an advantage in this situation. A test database can be prepared with a number of students, say 50 students and 300 tutoring sessions spread out randomly among them. Then data about what matches should be found and what ones should not be found can be encoded into automated unit tests, which can be re-run easily and often to help validate the effectiveness of the suggestion algorithm.

This does leave one major problem: where to get data and how to label it. This problem is one that must be solved by hand. Test data generators are readily available online to generate people and calendar events easily enough which should be easily imported into WolfTutor's Mongo database, but several hours of hand labeling will need to take place for this evaluation method to produce useful results. As of the time of this writing, the team does not have a better approach to this specific manual-labor task outside of possible croudsourcing.

5.2 Evaluating Usability

The next major priority is evaluating the usability of the addition to the application. In this case, the primary metric of usability is the time it takes the schedule tutoring sessions in the system. This is something that can and should be tested with actual users rather than relying on an algorithm.

To evaluate usability the team will conduct a focus group with ten to twenty potential users from this CSC510 class. Each respondent will be asked to use both the old and the new system while being timed. They will be asked to schedule some number of tutoring sessions one after the other using test data generated during the previous phase of evaluation. The team does intend to offer the new and old system in a different order each time, to help control for the fact that users will not necessarily already know how to use the system the first time, which may skew results.

The team also intends to ask respondents to give feedback about the matches they recieved when using the application. This is obviously not as interesting as if the actual tutoring sessions took place, which is why it is not being solely relied on as the measure of the effectiveness of this enhancement, but if these results contradict the previous results, the enhancement may be flawed.

6. CONCLUSION

6.1 Future enhancement

During our initial brainstorming and pre-survey, we actually come out with a few enhancements features, these features have been discussed in previous sections. Consider the short time frame for the project, we chose to implement one of the most rated features. the other features will be considered as the futrure enhancements.