WolfTutor

A system to enable peer tutoring built on Slack

Monica Metro
NC State University
mgmetro@ncsu.edu

Zachery DeLong
NC State University
zpdelong@ncsu.edu

Zhangqi Zha
NC State University
zzha@ncsu.edu

ABSTRACT

In this abstract, we need to preview our experiment and our results

1. INTRODUCTION

1.1 WolfTutor

WolfTutor is a system that seeks to enable students to tutor other students in a course-setting. It is a slack-based chat app that attempts to connect potential tutors in given subjects with students who need help with those subjects. Put another way, WolfTutor is an applicatio nthat focuses entirely on enabling peer 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.

There are a number of things that WolfTutor does extremely well, chief among them its novel interface. Being a chatbot makes the application easy to port to new languages and new devices, since its entire UI consists of simple english sentences and some rudimentary web inputs like text boxes and drop-downs. That being said, there are a few potential areas for improvement, chief among them scheduling and tutor-student matching.

1.2 Tutor Matching

WolfTutor's existing mechanisms for matching students to tutors are fairly rudimentary. When tutors register for the application, they are asked to give a set of subjects they are comfortable helping on (the list of which is determined and maintained by system administrators) and a list of days and times they are available to instruct.

While immediately useful, this is not nearly as far as the system could go in terms of matching students with tutors. The team has built out a mechanism for enabling the matching of tutors on multiple criteria and has created an easy pathway to add or change the criteria easily.

One major concern for the team during development was the choice of a recommendation mechanism. While the team knew immediately that adding some kind of recommendation algorithm was going to be necessary, the actual mechanism is something that was debated significantly. In the end, the team opted for Occam's Razor: the simplest algorithm possible to create the most value possible, which in the future could easily be replaced or enhanced to compare performance.

The tool the team chose was a simple weighted average. The process is fairly simple: several criteria were chosen to each generate a respective "score". That score is then assigned a weight, and each tutor's score is averaged together with these weights, which can then be normalized and used to rank tutors. In this way, the problem of recommendation becomes a sorting problem, which can easily be solved using a number of very fast algorithms.

This approach also has the advantage of leaving each tutor with an individual score which can be used as input to a number of other possible algorithms, which will be discussed in section 5.

1.3 Literature Review

While the merits of peer tutoring are not the major focus of this paper, it is prudent to give a brief overview of the topic. Peer tutoring is a type of tutoring where a student seeks out instruction from another student who has already studied the subject that the student is interested in learning more on. There are a number of terms for the tutor in this situation such as "mentor" and "proctor" but for the purpose of this paper we will simply use tutor to refer to the student-tutor and student to refer to the student seeking help. [1]

There are a number of different well-documented benefits to peer tutoring, and they are not limited simply to the students being tutored. While the intention is to improve the students first, the process of teaching another student has many well-studied benefits for the tutor as well. [1]

While it is absolutely true that tutors benefit significantly from teaching their stduents, the obvious goal is to transfer knowledge from the tutor to the student. It is well documented that students tend to feel more engaged during one-on-one peer tutoring and that the peer tutoring can give more feedback than lecture-style learning which can in turn

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

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

DOI: 10.475/123_4

help reduce student anxiety and improve learning outcomes. [2]

1.4 System

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.

1.4.1 Student Use Cases

Students can:

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)

1.4.2 Tutor Use Cases

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

2. DESIGN

2.1 Enhancement

As mentioned in section 1.2 the goal of this project is to improve the matching done by WolfTutor. In its original form, students input only what subject they were interested in a tutor for and the system returned back all students in that given subject. While effective, this brute-force strategy can wind up showing students an extremely large number of potential tutors in their chat app, which can be somewhat challenging to navigate.

The original system also displayed fairly little information about the tutor when searching. The list of tutors returned by the "find a tutor" functionality contained basic information such as name and major, but did not include information about the performance of the tutor academically or their reviews. To see reviews, the searching student would have to hit a "review" button which would then display all the reviews that tutor had at the bottom of the screen. While this technically made the necessary information available, this UI was challenging to use, and after getting feedback from the first round of reviews, the team decided to make some changes to this process.

First and foremost, however, the goal of this project is to improve the process of chosing a tutor in WolfTutor by providing a filtered list of tutors to students when they search for tutors in the system. To do this, the team identified four major metrics of a tutor to use as a basis of comparison.

2.1.1 Overall Review Score

The overall review score is the most obvious metric for evlauating a tutor currently. This score is simply a mean of the reviews (on a scale from 1 to 5) of a given tutor. A simple average, though, has a few problems.

- Tutors who have used the sytem for a longer period of time are less sensetive to fluctuations in their rating and a dramatic change in quality of tutoring may not affect them appropraitely.
- Tutors who had a temporary period of poor reviews or a single bad experience early in their career may struggle to find students to redeem their record after a small number of poor reviews.

Given the potential for abuse, the team decided to break the reviews into a rolling thirty day window. By keeping reviews to the past thirty days, the team hopes to help prevent some of the potential abuses of the mean review score by allowing outlier reviews to fall off on a regular schedule. This also serves as a small optomization to the review calculation process to help keep the calculation of the mean scores fast enough to scale in a production system.

It is also worth noting that this window is configurable, so if a problem were found during evaluation, this is one dimension that the system could be tweaked to help prevent abuse.

2.1.2 Individual Review Score

The individual review score is one dimension that offers personalization to the reviews. While it is obviously a good idea to order tutors on the basis of their overall review rating, it is also possible that particular tutors work well with particular students. To that end, any system making personalized recomendations should take into account the searching student's preferences and past experience. This is what the individual review score attribute attempts to do. This score is the mean review that the current student has given to each tutor.

2.1.3 Tutor score (working title)

The tutor score is similar to the individual review score. Like the individual review score, it is a score specific to the logged in student, but instead of being the mean review score the current student has given a given tutor, it is the mean review score a given tutor has given to the current user. This attempts to solve two problems.

- It helps new tutors rise in the rankings. It helps tutors with short review history but strong experience being tutored show up in recommendations.
- It helps further personalize recommendations.

2.1.4 GPA

The student's GPA is the last dimension the team found time to add, and it needs little introduction. At some universties such as Marshall University, students are not authorized as peer tutors unless they have attained a GPA

better than some threshold. While WolfTutor does not currently employ such a GPA filter, it is thought that such a minimum viable threshold would probably reveal itself in whatever culture the application was deployed to.

2.2 Bugs

The existing code was of a reasonable quality at the time that it was handed off to group-p. That said, there were a few issues that needed to be addressed before work could proceed.

The first and most glaring is in the documentation. The documentation provided to group-p was actually quite good. The team was able to get an application up and running with the basic specification within a few hours, but one step was left out that prevented any registration as a tutor or searching for tutors: a database entry had to be created for at least one subject manually. Once the team reached out to the original developers of WolfTutor, the problem was rectified within 24 hours and the appropriate documentation was updated.

The next bug that had to be fixed was in the review process. For whatever reason, the exisitng review functionality was nonfunctional when tested on group-p's development environments, and some time had to be spent getting that functionality working for testing.

Lastly, there was a very minor issue that would cause a timeout message to appear every time the bot was restarted and a new student would start interacting with it. This bug did not break the software, but the team did spend a little time to root it out just the same.

2.3 Use Cases

testing

2.4 Architecture

The main architecture of the chatbot was broken up between the ways the user interacted with it and the helper functionality to support those interactions.

Any interaction with the application via text is handled in the "main.js" module. In this module are event listeners for text entry events in the Slack chatbot. Any time a student is typing a response or initiating contact with the bot for the first time, the events for handling their inputs are in this file in a series of listeners and callbacks. Interestingly, the Botkit API used for this project does not natively support a more modern JavasScript promise-based design.

Dialogs are the other main way that students interact with the system. In Slack, a dialogue is a pop-up window that appears over the chat window and contains web-controls such as text boxes, radio buttons, or drop-down select boxes. These dialogs can be sent to the chat bot through the responses to user's input in the "main.js" module, and can provide a richer interface for providing structured data such as selecting subjects from a list or time slots in a calendar. The dialogs themselves also have a "message response" identifier which will be sent back to the slack bot when the user responds which is the other primary way users interact with the system. These response identifiers provide an easy way to re-order or re-use messages without having to make large code changes. Interestingly, slack only allows 5 controls to be in a single dialogue at one time.

The last major place for code is the modules folder. The database access is handled through the models but helpers are built around complex tasks and stored in appropriate JS classes in the modules folder. for example, the actual database manipulation for tutors happens in the tutor model, but a tutor module also exists that utilizes that module and adds business logic to it.

2.5 Infastructure

testing

2.5.1 Slack

As mentioned previously, the target for this chat bot is to run in the Slack platform. Slack is primarily a business tool intended to help facilitate communication and coordination between individuals and groups at organizations. In practice, however, Slack has found a home in many unintended audiences such as online communities and even university courses such as this one.

While it is true that this bot has been deployed to Slack specifically, it was developed using a popular API called Botkit, which actually has APIs for multiple chat services such as Facebook Messenger or Discord as well as Slack. Because of this, it is possible to port WolfTutor to these other platforms without needing to entirely start over.

2.5.2 NodeJS

NodeJS is the primary framework being used in this project. Node is a runtime for Javascript which has proven to be strong competition for more traditional LAMP stacks in the web. In this case, it provides a number of different libraries which make development of a chatbot easier. Running Javascript also has the benefit of making the chat bot easier to work on for a large number of people given Javascripts ubiquity in recent years.

2.5.3 Heroku

3. EVALUATION

4. RESPONDING TO CHANGE

5. CONCLUSION

5.1 Results

5.2 Future Work

6. REFERENCES

[1] M. M. Kim. Peer tutoring at colleges and universities. *College and University*, 90(4):2–7, Summer 2015. Copyright - Copyright American Association of Collegiate Registrars and Admissions Officers Summer 2015; Document feature - ; Last updated - 2017-11-22; CODEN - COLUBY; SubjectsTermNotLitGenreText - United States–US.

[2] K. J. Topping. The effectiveness of peer tutoring in further and higher education: A typology and review of the literature. *Higher Education*, 32(3):321–345, 1996.