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 4.

1.3 Literature Review

1.4 System

2. DESIGN

2.1 Enhancement

In this section we will design our goals for the system and outline what we wanted to accomplish.

2.2 Bugs

In this section we will talk about the issues we had to fix in the system before working and how we helped avoid them in our new code

2.3 Use Cases

testing

2.4 Architecture

testing

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

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

DOI: 10.475/123_4

2.5 Infastructure

testing

3. EVALUATION

4. CONCLUSION

4.1 Results

4.2 Future Work