Team Centrosaurus Milestone 3 Report

State of the Project

We are currently on track with the Milestones, and maybe even slightly ahead of schedule. For Milestone 3, we planned to have the RL model working and tested, separately from the web-app. Currently, the RL model is working in isolation, but when integrated into the sample control flow the web app will follow, it requires changes that may not maintain its integrity as an RL model. This will be further examined and we will work to ensure the model is indeed learning by the end of the week. In isolation, the RL Model was able to update a stored vector based on a loss function determined by a label provided at runtime. On the front-end side of the web app, most of the UI has been designed and implemented. The API has been established as well, but we are currently working on connecting them so we are able to receive and send requests from the front-end to the back-end.

Proposal Changes

There are no major proposal changes to the ML component at this milestone. Due to time limitations we have elected to reduce the complexity of the non machine learning part of the application. Rather than have multi-users we will focus on a singular user, that can be extended to a multi-user system if time permits. We will also be focusing on getting a local deployment for the final milestone, rather than a full deployment. These targets are more realistic while still preserving the components of the original project. The decision to modify these was also brought about in part due to the delay in receiving access to Google Cloud, which had been planned to play a major role in both of these set-ups, through both Google's Identity Platform and Google App Engine.

Current Challenges

The biggest challenge that we are facing right now is integrating the front-end and back-end of the project. We did not work out the exact details of how we were planning on doing this earlier as we did not know how all the elements were going to work. Due to this, we are finding it tricky to receive and send information between the front and backs how we originally intended. We will try to edit what we have to make it work, but we are also aware of a way that the API could be restructured that may yield better results. We are also working through issues with the RL model when implemented into the control flow the finished project will have. In order to produce what appears to be learning, we had to keep the produced graph between back propagation calls. Whether this has a significant effect on the efficacy of the model is yet to be fully tested, and will be an immediate focus for our work at the beginning of this week.

Team Member Breakdown

Byrne, Declan

Most of Declan’s working for this Milestone was working on developing the backend with Harvir. He implemented a small scale model of the RL component, before working with Harvir to extend it and define the functions that will be called by the API calls. Additionally he worked on integrating those function calls into the API functions, getting it to a point where the backend can initialize without any apparent issues. Currently he is working to debug a sample control flow written in colab that represents all the function calls a typical use of this app will make, without actually working on those calls through the front end or API. This is where he will continue his immediate work for this milestone, and once completed will work on ensuring the API calls are working.

Dhaliwal, Harvir

Harvir worked to complete the development of the backend code for the project. Currently, it may have some small computational bugs, however the majority of the code has been tested and seems to work correctly. The majority of this work consisted of developing the RL model that was designed by Declan and Harvir. Harvir is currently working on further testing the backend code and the RL model to ensure that it is learning correctly. He is also working a bit in the front end of the web app with Ritchie, mainly with the UI design.

Riley, Anna

Anna worked primarily with the API and setting up a Mongo database to store user data and models. She created a REST API using Flask. She initially created three API functions: get, put, and patch. The functionality of these functions was to send 5 book recommendations to the front end, store a user's initial ratings of 5 random books, and update user data each time a user swiped, respectively. She initially created a Mongo database for the user's initial ratings and swipe history. After the team decided to make some changes to the structure of the app, she changed the API functions to be just get and put (and a simple delete user function)to get 5 updated book recommendations, and update the model of a user using their swipe data. Finally she made some simplifications to the Mongo database to account for the overall design changes.

Xia, Ritchie

Ritchie worked on front end web development. He created a web page using React and has completed the machine learning portion of sending responses and getting the next books to recommend through fetching with the API. The current state of the web page allows the user to swipe on images to send positive or negative feedback on a displayed book. It also saves the books swiped right into a separate state that stores the user’s reading list. For the upcoming milestone, he will be implementing the code to render the reading list as well as a method of removing books from the reading list. He will also need to debug the API calls accordingly to ensure all calls are correct and working.