Team Centrosaurus Milestone 4 Report

State of the Project

We have successfully produced our project goal in a local environment. We have a webapp which upon initialization prompts a user for their ratings of 18 books, once the user rates these books, they are used to pair the user with an existing user in our embedding model. The features of this user are then used as the initialization vector for our RL model, which as the user swipes on books improves this vector. Since the last milestone the API has become fully functional, the dataset has been properly processed, a new embedding model has been trained, more books have been coded for the initialization, and the book info has been moved to the MongoDB database.

Proposal Changes

The initial proposal talked about having the user rate the books after they read them, and using the new user, book rating information to expand the embedding model. However since as discussed in M3 we are no longer going to be having multiple users for whom we store information, this idea of retraining is no longer possible. This is essentially part of the change, mentioned in M3, to move away from multiple users due to time constraints, and an initial lack of access to Google resources. We mention this here since it wasn't acknowledged when we first moved away from a multi-user setup

Current/Future Challenges

There are no current challenges that we plan to face, as the project is functionally complete. Instead here we will address challenges we would face in expanding the project. If we were to move towards having more users in the system, as well as persistent memory of the users vectors we would likely need to expand our use of the database. This might be able to be done similarly to how we've moved the book information to the database, but we would need to be cognizant of how we control information just for specific users to see. Looking into Google's Identity Platform could probably offer some clarity as to how to manage multiple users. Adding in the ability for the embedding model itself to improve by expanding its user base would offer it's own set of challenges, like how we train it in the background of the app running, and how frequently we pull from our user set to train it. To accomplish this it might make sense to abstract that functionality to separate web apps which can draw from a database the first one uses, and only runs at set times to improve.

Team Member Breakdown

Byrne, Declan

Declan primarily worked on debugging the backend, and the API. He worked both by himself, sending solitary API requests, devoid of context, as well as with the frontend and the team. He then worked on reprocessing the dataset for a myriad of reasons. First of which was to remove any books in foreign languages, by their language code in the books.csv file, next was to eliminate unnecessary parts of the dataframe in order to minimize the space taken up on Mongo. The dataset class used for the embedding model now needed to be changed to reflect the removed books so he did that as well, and used Harvir's determined parameters to train the final model, minimizing testing loss to 0.67 after 20 epochs.

Dhaliwal, Harvir

Harvir worked on retraining the embedding model to reduce loss as much as possible. He tested out different parameters for dropout, batch size, learning rate, and momentum and achieved a model that can get 0.69 test loss and 0.72 training loss in 15 epochs. The key to reducing loss as much as possible was to incorporate a dropout of about 0.25. There were also some other problems that were discovered when training this model on the re-processed dataset, so Harvir worked with Declan and Ritchie to debug those issues.

Riley, Anna

Anna changed the format of the API so that the calls from the front end could route to a specific path for requesting data and sending data to the back end. She helped with debugging the API calls from the front end. She then worked on storing the book data into the mongo server and adjusted the dataset to retrieve the data about each book from the database. Additionally, she selected the representative 18 books to be hard-coded into the initial user tutorial period, and found the goodreads ids of those books.

Xia, Ritchie

Ritchie worked on developing and finishing the front end. Working with the entire team to connect the API properly, he coded in the proper fetch requests to get and post information, as well as unpack the groups of 5 recommended books. He also fixed some code throughout the entire project to help debug the API calls and ensure information was sent and received in a consistent way. He also polished up the reading list to a fully working state.