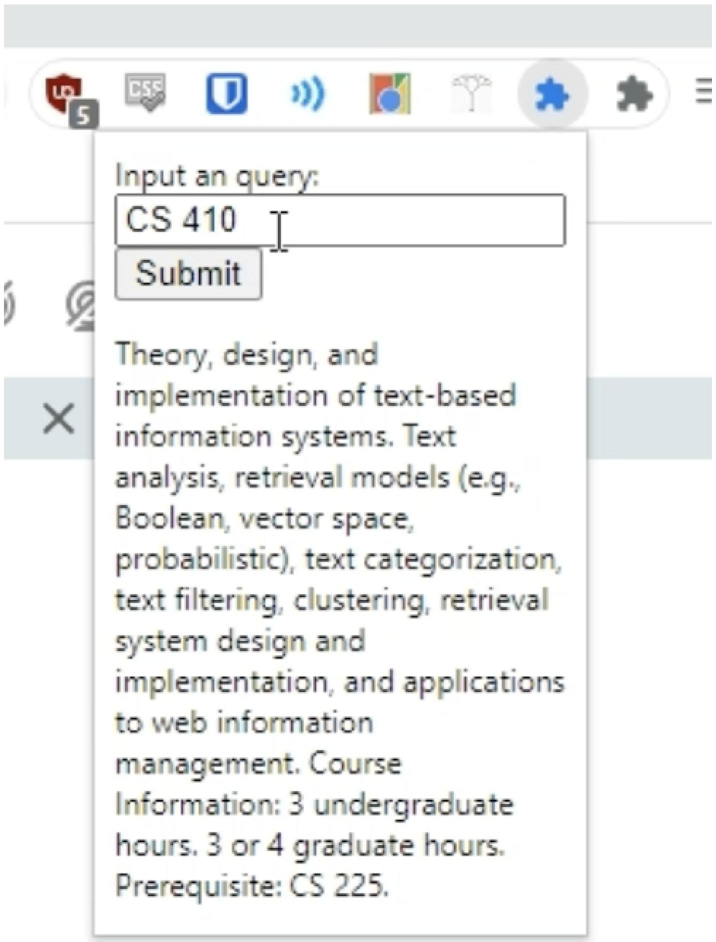
Progress Report - Courseling

(Captain) Jason Xu, jasonx5
Andrew Ko, hyunjun5
Nick Chun, nechun2
Sophia Yu, sophiay3
Sahil Agrawal, sahila4

In regards to the tasks outlined in our Project Proposal, we have completed at least half of the total hours required for our project's minimum viable product. Specifically, there are four main areas that have reached a completed state. We also have four areas that require more work before the project is completed. And there are a couple of challenges that our team has faced that will be discussed.

The first step we completed was setting up a NoSQL MongoDB database, which is where we planned to store all of the data required for model training and tuning. We then undertook the task of collecting the data required to train our probabilistic language model. This entailed going through the list of Computer Science courses at UIUC (https://cs.illinois.edu/academics/courses – All courses) and using Python web scraping libraries called BeautifulSoup and Selenium to scrape the useful information off of the respective course pages. Each course was uploaded to our MongoDB database with parameters including the course's name, level, professor, credit hours, any prerequisites, semester, year, and description. For our minimum viable product, we have decided to complete the Chrome extension in Javascript and link it with a backend Python webserver using a tool called Flask. We have accomplished the initial task of setting up Flask and linking it with the frontend Chrome extension, showing that we are able to access the course information in our database from the frontend UI. Consequently, this means that we have also provided a search proof of concept where a user can input a query and receive a result. An image of our POC is shown below in **Figure 1** below.



Input an query:

CS 410

Submit

Theory, design, and implementation of text-based information systems. Text analysis, retrieval models (e.g., Boolean, vector space, probabilistic), text categorization, text filtering, clustering, retrieval system design and implementation, and applications to web information management. Course Information: 3 undergraduate hours. 3 or 4 graduate hours. Prerequisite: CS 225.

Figure 1: Image of initial Chrome extension querying for a UIUC CS course

There are still some remaining tasks for our project, most of which are related to the construction of our text retrieval and ranking system. We have not yet begun integrating our model with the other aspects of our extension. However, we have decided that we are using the BM25 algorithm provided by Python's metapy library and have confirmed that it is compatible with our data. Before we can train the model with our data, we realized that we needed to first convert our database to a ".dat" format, which will provide compatibility with metapy. This is especially important because our data needs to be in a proper structure such that the BM25 metapy algorithm can understand the training data we are providing. In summary, for the backend we still need to complete the text retrieval and ranking system and the related topic extraction from our dataset. On the frontend side, we still need to add more components. Specifically, we need to render the ranked list of results that will be returned after a query is submitted.

There have not been any significant obstacles that are blocking our project’s progress. However, there are a couple of points that we still need to clarify and find a final solution to. For one, we need to find a method of understanding vague queries and returning the most relevant information back to the user. In case the user inputs an ambiguous query, we realized that we might need a fallback method of text retrieval. We came up with the solution that we could concatenate all of the parameters from each course and compare the query itself with that concatenated string to find the most relevant course. Additionally, we plan to optimize the text retrieval algorithm in order to give the best results possible, but we have yet to fully validate a method of optimization. For example, there are several extensions to the BM25 algorithm, but those methods require editing the scoring function itself. If time permits, we will research this further and come up with a viable method of optimization.