Courseling

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

For our project, we realized that many students are struggling with finding the right courses that offer the kind of materials they want to learn or getting more information about courses that they have been interested in. Consequently, we decided to come up with an intelligent Browsing system for UIUC MCS program courses where given a query about a topic or course, the system will return the ranked list of the courses that are related or give information about the course if the course is directly queried. This is related to the theme and to the class scope since we are going to be implementing state-of-art text retrieval methods for our search engine to be able to provide users with information related to the query.

We will scrape data like the course names, content and syllabus from the UIUC course explorer and the respective course websites. For the sentiment analysis, we will scrape data on the reviews of courses from websites like uiucmcs.org and rate my professor. We will be using a probabilistic language model to model the data and use the BM25 ranking function to rank the courses and provide the top ranked ones. We will use precision, recall and F1 score to measure the success of our returned results. Additionally, we will use techniques like stop word removal, stemming, tagging, etc to improve the accuracy of our model.

We will demonstrate that our approach will work as expected by visually inspecting that the top 5 returned results are relevant to a sample set of queries. In the case where a course is directly queried, the top result should be about the queried course information. We can also look at the accuracy of the system's ability to rank relevant documents before others by measuring precision, recall, and the F1 score for the model.

As the final product is a Chrome extension, our team will be using JavaScript to build it. Additionally, if we need to store any documents, reviews, course information, etc. in a database, we may utilize a relational or NoSQL database (such as MongoDB for unstructured data). For relational, we would use SQL to populate that database. For NoSQL, we can also use JavaScript again.

The hours listed below are estimates of the minimum needed time for each task. Our team has 5 members, so the total minimum hours required is 100. We expect the frontend UI to take 20 hours, the implementation of the text retrieval system and

integration with UI to take 30 hours, model training/tuning to take 35 hours, and other logistics (e.g., corpus collection, tagging, stopwords, etc.) to take 15 hours.