CS410 Text Information System
Final Project Report
Fall 2021

Team Name: Mission Lucy

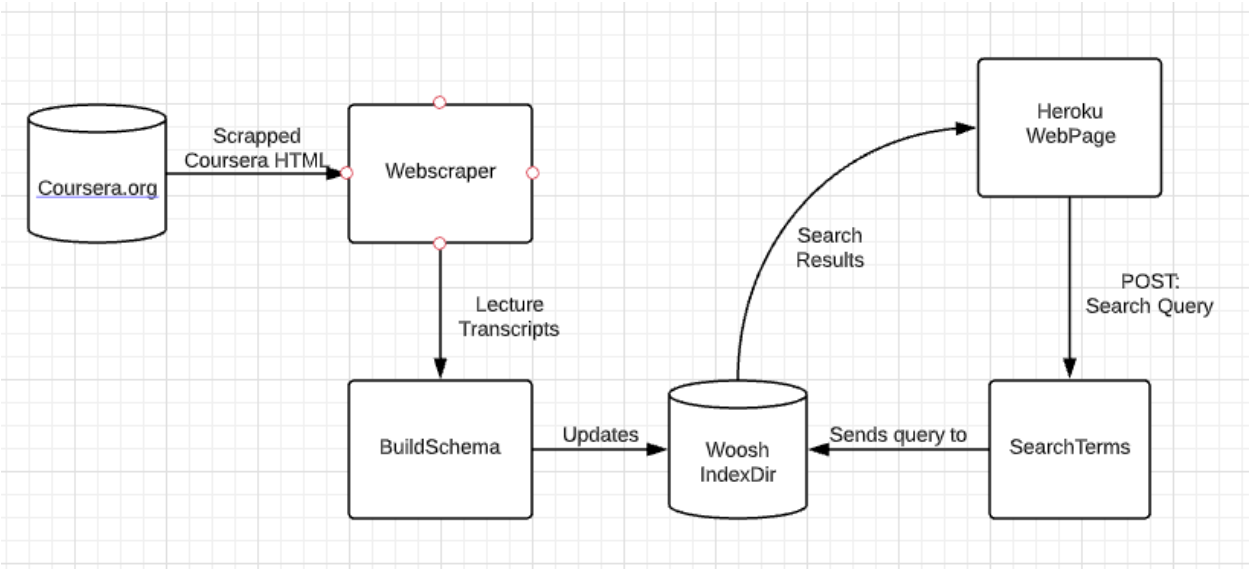
Topic name: Intelligent Learning platform -Video segment search

Team member	Student-id	Captain
Sam Song	samsong2	Yes
Kavithaa Suresh Kumar	ks64	
Uttam Roy	uroy	
Azim Keshwani	azimmk2	

Github: https://github.com/samsong2/CourseProject
Video Demo: https://uofi.box.com/s/6lb3ayuvgb1ytcreukt77shivhk48jy2
Application Demo: https://coursera-video-search.herokuapp.com/

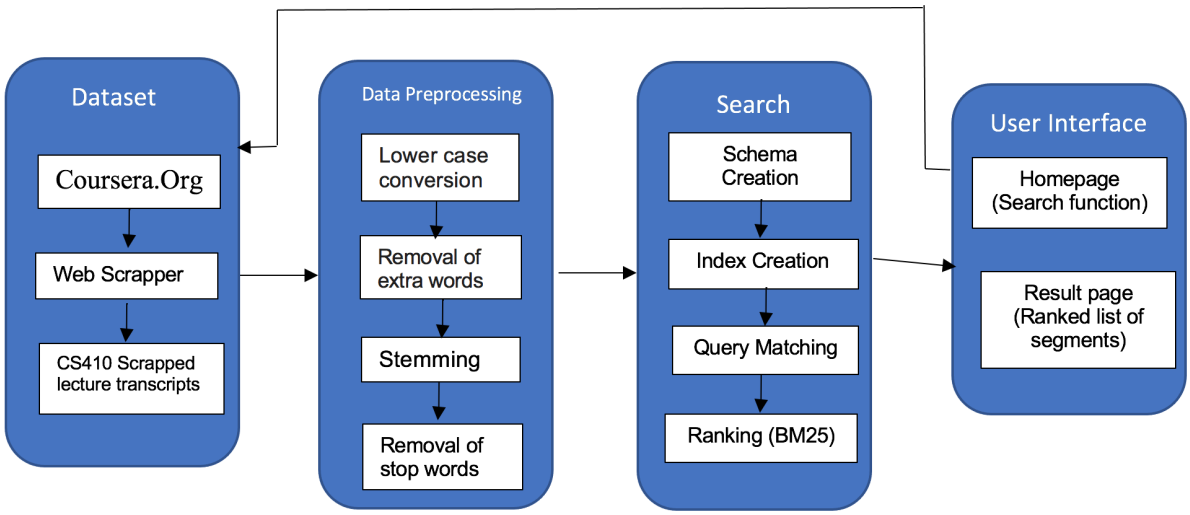
An overview of the function of the project

Our project helps users to easily access shorter segments of the lecture video that are relevant to the user query. Generally, the user enters a query and the search engine searches the CS410 lecture videos to output a ranked list of relevant clips from the lectures and the user can watch the relevant clips of the video.



Software Implementation

Flow Diagram:



The webscraper will, given a course name, attempt to download all of the lecture transcripts and video links for that course by scraping Coursera.org's class, week, and

lecture webpages. Selenium is to download the pages from Coursera and BeautifulSoup to scrape the coursera lecture pages of CS410 course to extract the lecture contents. The webscraper uses a 3 step process to find and download the lecture contents. First it visits the class's home page. This can be found at https://www.coursera.org/learn/{class_name}/home/welcome. It scrapes the home page for links to each week. Then from every week page it scrapes for the link to each lecture in that week. Finally from each lecture page we scrape for the lecture transcripts and the link for the lecture video. The contents from each lecture is then stored under /data/{class_name}/{lecture_name}.txt.

Search Engine

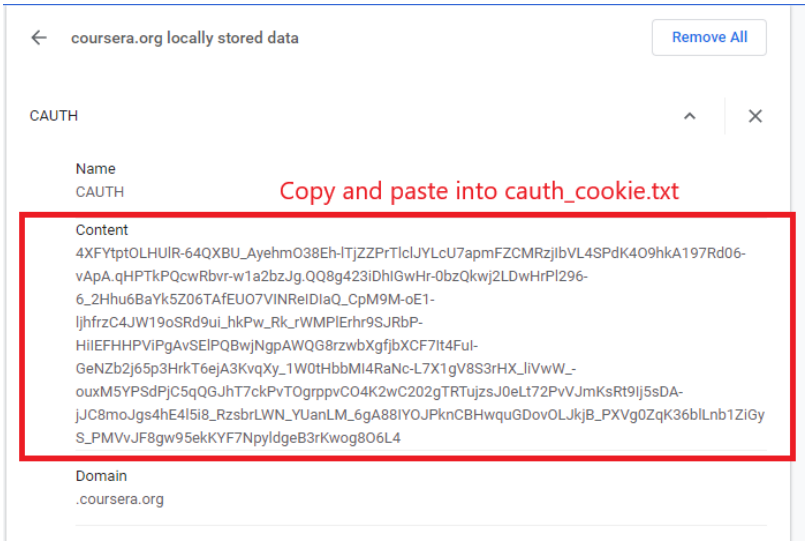
We used Whoosh, a pure Python search engine library used for searching the content. Before the data is used for searching, certain data preprocessing like lower case conversion, removal of extra words, stemming and removal of stop words are performed for the content and the query with the help of Whoosh python library. Next, we created schema and indexing for the content using Whoosh(python library) to use in the search function. Then the query is matched from the content using the schema and index and a list of results are generated with the help of whoosh library. We used the BM25 method to rank this list, then this result (Start time, content and the video link) is returned to the front end.

For the front end, we have 3 html pages, one for the home page, one for the results, and other to display the video clip. We are using gunicorn as the web server and the flask library is used to give the pages functionality. In the first page, the user types the query they can press search and will be directed to the results page, which displays the result set, the user can click on any result and will be directed to the relevant area of the video clip. To make things easier we have deployed the front end through Heroku.

Documented usages for this Project:
Requirements:
Heroku account
Python Modules :
-Install whoosh
-Install Selenium
-Install Beautiful soup
-Install flask
-Install gunicorn:

Instructions:

1. Fork and clone this repository
2. Webscraper: Requires[Selenium, Beautiful Soup]
    a. Navigate to /src/webscraper
    b. Download the chromedriver for your version of chrome.
        i. Find the chromedriver for your version of Chrome here https://chromedriver.chromium.org/downloads
    c. Update cauth_cookie.txt
        i. Login to Coursera.org
        ii. Go to chrome://settings/cookies/detail?site=coursera.org and copy the value of the CAUTH cookie into cauth_cookie.txt



    d. Run scraper.py to download lecture data for CS410
3. Creating/updating search index directory: Requires[Whoosh]
    a. Navigate to /src/Search
    b. Run BuildSchema.py to build the search index directory
    c. Add all the files from the newly built index directory to git repo

4. Launching Search Page:[Flask, Gunicorn]
    a. Create an account with Heroku
    b. Create a new app in Heroku at https://dashboard.heroku.com/new-app
    c. Use Heroku (Web) UI to Deploy the Application
        i. Connect your Heroku app to the GitHub Repository you created (*by forking*)
        ii. Enable Automatic Deploys
        iii. Push current changes in forked repo to have heroku start building new page
    d. Navigate to your newly created Heroku search application
    e. Enter query
    f. Click on a lecture title to view lecture video starting at that segment.

Team Contributions:

Sam - Scraping Coursera's website for lecture transcripts and lecture video links. Hosting search application in Heroku.

Kavithaa-Data preprocessing, schema and index building, query search and ranking the results

Uttam--Data preprocessing, schema and index building, query search and ranking the results

Azim- Developing UI and testing