Theme 1: Intelligent Browsing

Instructions

Description

Most people who browse the Internet do so through popular browsers, such as Google Chrome, Mozilla Firefox, or Safari. Indeed, we complete many of our tasks by using a browser, thus how well those browsers support our tasks may significantly affect our productivity. The current browsers are great for quickly finding or saving information, but they lack "intelligence". Adding more intelligence to such browsers can turn a browser into a personal intelligent assistant that can broadly impact many users, potentially transforming how they access information on the entire Web.

The goal of this track is for groups to build on top of existing browsers using the information retrieval techniques learned in this course. One straightforward way to extend browser functionality is to develop browser extensions (primarily written in JavaScript). Some extension-based project examples include:

1. Index the current page and allow users to search over the page using a common retrieval function, such as BM25 (the current search capabilities are limited to exact keyword match)
2. Scrape and index Campuswire pages and Coursera pages in order to link questions to content, and vice versa (you can think about how in general you might leverage a browser extension to help linking the scattered educational content such as Coursera lecture videos, textbooks, and relevant discussion on the Web)
3. Create a collective bookmarking, question-answering, or annotation system among specified groups of users
    a. As an example, see Fermat's Library

The above examples are meant to illustrate the problem domain; groups are free to propose a topic within this track that isn't in the above list, especially if it solves a well-known problem or shortcoming of current browser systems. Students are also encouraged to coordinate group work: e.g. one group could focus on the front-end design and another group could focus on the back-end server. This coordination would allow groups to collectively solve problems beyond the scope of a single group.

The following links may be helpful to get started:

1. Chrome extensions
2. Mozilla Extensions
3. Safari Extensions

Requirements

If you choose this theme, please answer the following questions in your proposal:

1. What are the names and NetIDs of all your team members? Who is the **captain**? The captain will have more administrative duties than team members.
   - Timothy Crosling (tgc3)
   - Avi Nayak (anayak5)
   - Srishti Sharma(srishti9)
   - Deepthi Abraham(deepthi7) - **Team Captain**

2. What topic have you chosen? Why is it a problem? How does it relate to the theme and to the class?
   Intelligent Browsing
   Extracting the keywords from the static pdf document

3. Briefly describe any datasets, algorithms or techniques you plan to use
   - Create and inverted index based on the text on a webpage
   - Rank the key terms based on an optimized BM25
   - List the top 5 terms and allow the user to find out more about these topics
   - Aiming to integrate this into the browser as an extension

4. How will you demonstrate that your approach will work as expected?
   User testing
   Manual testing of the BM25 ranking based on different documents (to fine tune the k1 and b parameters)

5. Which programming language do you plan to use?
   Python (backend and text processing)
   Javascript (web frontend)

6. Please justify that the workload of your topic is at least 20*N hours, N being the total number of students in your team. You may list the main tasks to be completed, and the estimated time cost for each task.
   Total 80 hours
   - Requirement Analysis, Algorithm analysis - 10 hours
   - Architect and Design (Frontend and backend design)10 hours
   - Coding and Implementation -40 hours
   - Testing and evaluation of code-10 hours
   - Demo (Code execution and recording the demo) -10 hours
   - Project report-10 hours

At the final stage of your project, you need to deliver the following:

- Your documented source code.

- A demo that shows your implementation actually works. If you are improving a function, compare your results to the previously available function. If your implementation works better, show it off. If not, discuss why.