Justin Feinfield
Ryan DeStefano
CS 410: Text Information Systems
October 27, 2023

# Project Proposal: Intelligent Browsing with Advanced Search

## I. Team Members

- Justin Feinfield (captain), justinf6@illinois.edu
- Ryan DeStefano, ryanfd2@illinois.edu

## II. What topic have you chosen? Why is it a problem? How does it relate to the theme and to the class?

We've chosen to develop a Chrome extension to search a web page's content beyond a simple exact keyword match. While hitting Ctrl-F on a page and searching for a keyword is useful, users often don't know exactly what to search for, so if they don't search for the exact right words / phrases, their search might result in any of the following scenarios:

- They find no results.
- They find the wrong results.
- They find too many results and have to manually search through those.

This makes searching pages containing things like long news articles, comments / reviews on a video / product, or several disparate sections cumbersome and unproductive. Searching would be much more helpful if users could enter a word / phrase that would be matched more closely without requiring exact precision. This relates to the theme of intelligent browsing in that this kind of more advanced searching is more similar to how a human would search from a query. We don't look for exact matches, but instead we search for contextual matches that match the overall meaning / sentiment of the provided keywords. This relates to the class as it represents a culmination of the first half of the course on text retrieval. We spent 6 weeks learning about different ways to rank and index documents and how to retrieve them efficiently. We can treat a webpage as a document or a collection of documents, so through this project we will be able to employ what we've learned to build an extension that indexes a page and provides intelligent search results efficiently and accurately.

## III. Briefly describe any datasets, algorithms or techniques you plan to use.

While we won't be using any preloaded / downloaded datasets, we can essentially treat the webpage as a dataset in that it represents a corpus of text that we need to index and search across to give users correct search results. We can leverage the Okapi BM25 algorithm to rank different sections of the webpage to give us better matches based on a user's search query. Due to the time frame of this project and the expected workload, we'll likely only support pages with static text and not reactive pages (e.g., Google Docs, pages with auto-generated text like ChatGPT or transcription sites, or pages that reactively show/hide text). We'll also need to create a simple UI search popup that users will interact with as a replacement for the default search bar that appears when a user enters Ctrl-F. To do this, we can use very basic React or vanilla JavaScript with minimal CSS to style the search bar.

**IV. How will you demonstrate that your approach will work as expected?**

To demonstrate that this approach has worked as expected, we can simply test the extension against a set of webpages and queries to benchmark its performance. We will have a list of webpages and a set of expected results for each query, and the effectiveness of our extension can be measured by how well each search's results match the expected results. We will also be able to demonstrate the improvement over Chrome's basic searching functionality by showing our extension returning results that, while they aren't exact matches, are relevant to the provided query.

**V. Which programming language do you plan to use?**

As most browser-related tools do, we will mainly use JavaScript, HTML, and CSS for this project to develop a minimal UI (potentially using React) for searching. We'll also use JavaScript for the underlying logic of the search extension. We have two options for using the BM25 algorithm:

- As the BM25 algorithm is relatively simple to implement, we can implement it ourselves using JavaScript. This would enable us to fine-tune our logic for our specific purposes.
- There are a few open-source JavaScript libraries that support the BM25 algorithm already, so we could use those instead of implementing it ourselves, but this won't allow us to tailor our implementation as much as the first option would.

**VI. 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.**

The total workload for this project is estimated to be 44 hours. This workload includes the high level tasks of creating the user interface for the extension, creating a data set to test the performance of the ranking function, creating the ranking function, and fine tuning different factors of the ranking to achieve the greatest performance. This estimate is broken down into smaller tasks below, with a task-level time estimate and detailed description of the task.

- Create initial extension - 2 hours
  - Create a very basic extension, with minimal UI and backend, This will allow both group members to work on different tasks in parallel
- Create set of web pages to benchmark performance - 4 hours
  - Collect a set of 10-15 web pages that can be used to test the extension. Having a fixed set of web pages, with expected search queries and results, will allow benchmarking results when fine tuning the algorithm
- Create search bar UI - 6 hours
  - Create the search bar UI using JavaScript and CSS for styling. The text typed into the search bar needs to be read in and used as input to the ranking function
- Backend to parse web page content - 4 hours
  - Parse the text of a web page. Potential challenges include parsing texts in headers, footers, and menu bars
- Break up content into "documents" - 4 hours

        - Break the parsed web page content into chunks, or documents, to be searched by the BM25 algorithm
    - Implement initial BM25 ranking algorithm - 2 hours
        - Implement the initial ranking algorithm, which can be finetuned in future tasks
    - Create UI element to display the number of relevant documents - 2 hours
        - This should display the number of relevant documents, as well as the current one being shown, such as “1 of 5”.
    - Create UI element to navigate between the relevant documents - 2 hours
        - Add up and down arrows to the search bar to allow the user to view all relevant documents
    - Highlight the relevant document in the web page - 4 hours
        - Highlight the currently selected relevant document in the web page itself so that it can be easily viewed by the user
    - Scroll to the relevant document in the web page - 4 hours
        - The web page needs to scroll to center the relevant document in the center of the page
    - Fine tune the BM25 algorithm for better results - 4 hours
        - Improve the results of the BM25 algorithm by testing different parameters. The results should be compared across the testing set of web pages to choose the best set of parameters
    - Fine tune minimum relevance - 4 hours
        - Fine tune what score for a document should be the minimum for that document to be shown to the user as matching the query. If this is too low, the user will get results that are not actually relevant, but if it is too high, they will miss some potentially valuable results
    - Fine tune the chunk size of the web page contents - 2 hours
        - The size of the documents that web pages are broken into will have an impact on the performance of the search results.