Group Members:
Felicia Wang (felicia5) - Captain
Phyllis Wang (pjwang3)
Eshia Rustagi (eshiafr2)

Final Project Video Demo: https://mediaspace.illinois.edu/media/t/1_i1tlblax

Documentation

Code Function Overview

The function of the code is to provide a more sophisticated way to search for specific queries in a Wikipedia article, as opposed to Ctrl-F, which only matches exact queries. The code is for a Google Chrome extension, and can thus be loaded into the browser. By clicking on the extension icon, the popup for the extension will open and allow the user to search for a query on the page. After doing so, the extension will display the most relevant paragraphs based on the ranked results which users can navigate to to find the locations of those paragraphs relating to their query. Using the BM25 algorithm, our extension hopes to find the most relevant results to a query, and be a better alternative to built-in page search tools.

Software Implementation Details

This code for our BM25 Document Search Chrome extension is separated into two main parts: the frontend interface which the user interacts with, and the backend. The backend will run all the necessary functions both immediately when a Wikipedia page loads (ex. to obtain a cleaned and stemmed version of the article's paragraphs) as well as after a user interacts with some part of the interface (ex. user searches a query that returns relevant results after performing the BM25 algorithm or clicks on a result to be sent to the paragraph's location in the article). All project files are located inside the "code" folder in the repository. A more detailed explanation of each important project file will be explained below:

manifest.json:

All Chrome extensions are required to have a JSON-formatted manifest file named manifest.json. This file contains important metadata, from basic required information such as name and manifest_version to more project-specific details that show what Chrome extension permissions and host permissions (strictly Wikipedia in our case) we will use as well as what file will contain our content script (content.js) and popup (popup/popup.html). We use Chrome's "tabs" permission to get the user's current active tab and the "scripting" permission in order to execute a script on that tab that will sort all resulting paragraph scores after a user query in descending order and navigate to the location of the paragraph on click.

content.js

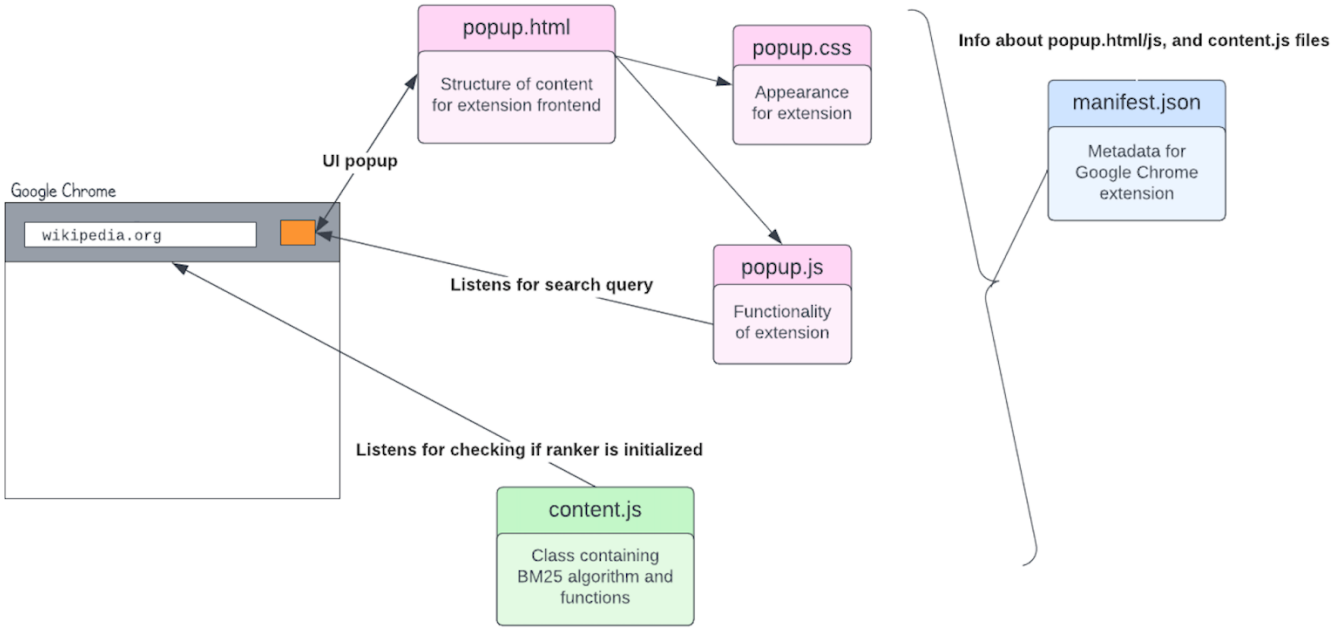
The content file primarily contains the declaration of our BM25 class, with functions used to clean up the paragraphs obtained from the current Wikipedia page (remove stopwords, punctuation, whitespace, and trailing spaces), stem the resulting filtered words using a JavaScript implementation found on the internet (properly cited), and extract any other necessary information about the text needed to perform the BM25 algorithm. This includes a unique list of vocabulary, doc frequency, and term frequency, among many others listed in the constructor for this class. We instantiate an instance of the BM25 class at the bottom of the file. We also have a Chrome listener that first checks if the ranker is initialized. If not, we clean the text and obtain all the BM25 information just described above. If so, we perform BM25 on the user query and send a list of resulting scores back for popup.js to process.

popup/popup.html, popup.css, popup.js

The popup folder with these three files contains all of the frontend of the extension that the user directly interacts with. After the user enters a query and presses search*, the listener in content.js will catch this and perform the BM25 algorithm also written inside the BM25 class (explained under content.js). After doing so, a list of all paragraph scores will be sent back, which is then sorted in descending order and the top 3 relevant paragraphs associated with those scores will be outputted underneath the “Results” section on the extension. Users can then click on each of those relevant paragraphs to be sent to its location on the article.

Code Architecture Diagram

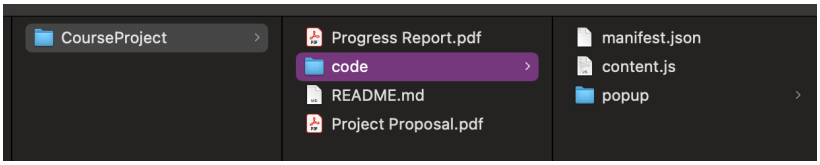
The following diagram shows how all the files are connected to each other.



Installation & Usage Instructions

Installation Instructions

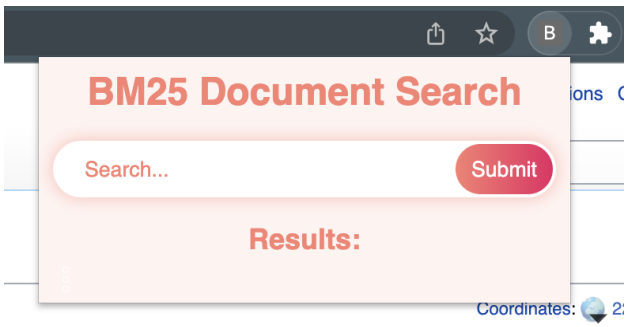
Our tool is a Google Chrome extension, and the installation instructions are similar to those from MP 2.1.

1) On the project github here (https://github.com/feliciawang08/CourseProject). Click the green “Code” button → click “Download ZIP”. The repo will download, and you will have to decompress/unzip the ZIP folder.
    a) Note that deleting or moving the decompressed extension folder after completing these steps will result in the extension not working in the browser.
2) In your Chrome browser, paste chrome://extensions/ into the address bar and navigate to the page.
3) Toggle on "Developer Mode" (at the top right of the screen).
4) Click "Load Unpacked". Inside the decompressed ZIP folder (titled “CourseProject”), select the “**code/**” directory.
    a) 

5) Toggle off “Developer Mode”
6) Click the "Extensions" icon in the Chrome toolbar (puzzle at top right of toolbar).
7) Pin the extension, "BM25 Document Search".
8) You can now use the extension!
    a) **NOTE:** This extension will only work on Wikipedia articles (not wikihow, wikia, etc). You will have to navigate to a valid wikipedia.org article to use the extension.
    b) Note that it will not work on the chrome://extensions/ page

How to use the extension

A. This extension will only work on wikipedia.org articles. You will have to navigate to a valid wikipedia.org article to use the extension.



B. Once you are on a Wikipedia page, you can click on the extension, and enter in a search query (with any number of words) to see the top 3 relevant paragraphs that match the search the best.*

C. When you get back the 3 paragraph results, you can click on any of them to navigate to that paragraph on the page.

Bugs/Errors with Usage

***NOTE:** Our extension has a minor bug that will require you to refresh the page after clicking on the extension icon in the toolbar, before using it. After you refresh the page (or alternatively, open the same wikipedia page in a new tab), then you should be able to enter a search query and use the extension.

Team Member Contributions

1. Felicia
    a. Created initial project structure for extension. This includes the manifest.json file, required for all chrome extensions which contains basic metadata (name, permissions, content scripts, etc.) about the extension. popup.html, css, and js files, will contain all of the frontend design and functionality which the user will interact with. Also created content.js which now contains the BM25 algorithm as well as various other functions that will help parse Wikipedia articles.
    b. Implemented function that takes all Wikipedia paragraph text, filters out the stopwords, and stems the remaining list using a Porter Stemmer algorithm written by different authors based on a paper about the Porter Stemmer.
    c. Helped create necessary functions and listeners in the popup.js and content.js files to pass the user query after the user presses the “Submit” button from the popup to the content.js file and run the BM25 function on that query.
    d. Designed frontend of extension (colors, font, spacing, etc.)
2. Eshia
    a. Added functionality for extension to look at current tabs and only work on Wikipedia pages based on the URL, and not a non-Wikipedia page. Accessed the webpage’s HTML to retrieve all the text in paragraphs when a user clicks on the extension. Worked on combining filtering and stemming logic with the vocabulary and document frequency logic.
    b. Implemented the functions to get the frequency of words in the search query, frequency of a word in all the documents, and the document lengths. Created the methods to compute the BM25 score for every document.
3. Phyllis
    a. Added form to search query in extension popup. Created functions for getting vocabulary and frequency of words per paragraph, which will be used for finding inverse document frequency as part of BM25.

b. Also added to functionality of popup, such as making results clickable and directing to its place on the page. Helped resolve some bugs with communicating between different parts of the extension.

Each of us contributed at least 20 hours to the project, totalling up to at least 60 hours. Although our initial proposal for our project included running our extension on PDF documents, we decided to prioritize getting our BM25 function to work properly and accurately on Wikipedia articles and thus dedicated all of our time on doing so. At the end, we were still able to exceed the minimum hours worked on the project without the BM25 functionality for PDFs.