Software code submission with documentation

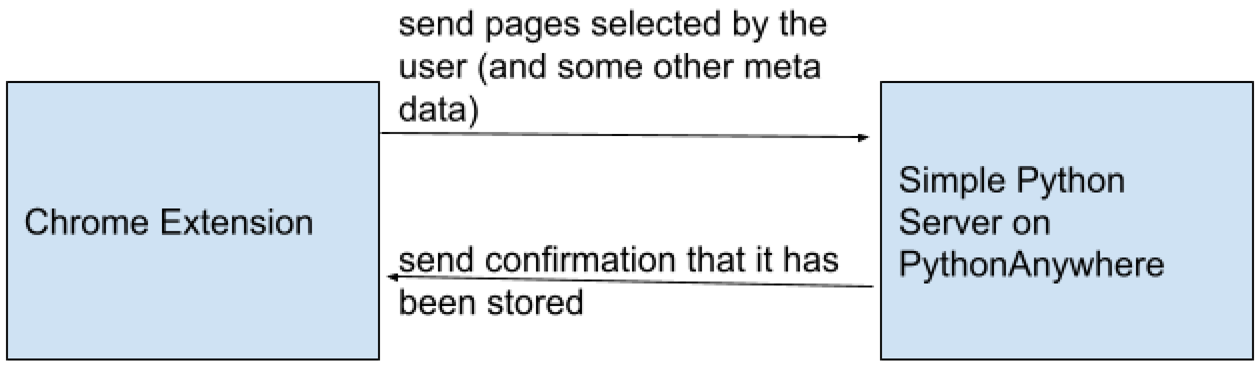
IMPORTANT NOTE ABOUT CHROME EXTENSION:
Credit goes to CS 410 TAs (or whoever developed it originally) for originally developing this extension (I took the relevant parts and slightly modified/extended it [simplified some parts of it for this final project], the server is something I did by myself). I do not claim to have developed the original version of this extension, but I have modified it to work with my server (where most of the work was done).

1) An overview of the function of the code (i.e., what it does and what it can be used for).

This project that I have worked on is to create an extension to index the current page and allow users to search over the page using a common retrieval function (ie. BM25), also extracting some of the topics from the pages.

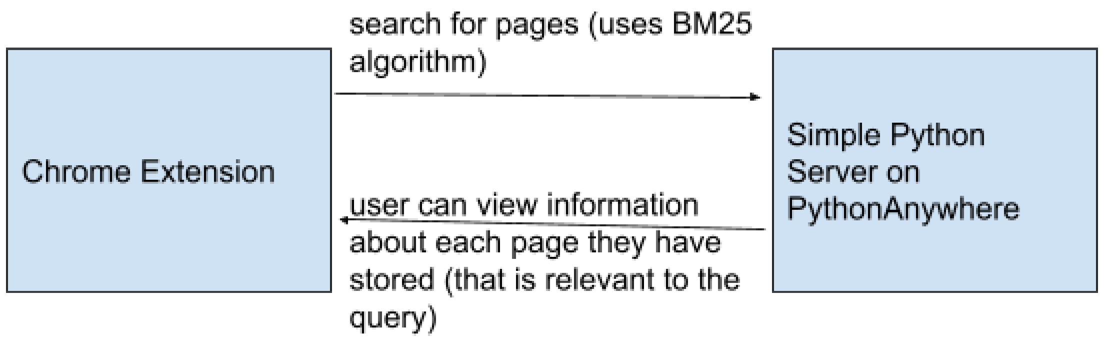
2) Documentation of how the software is implemented with sufficient detail so that others can have a basic understanding of your code for future extension or any further improvement.

There are two main components
- the webserver (Flask Server) which handles all of the requests by (1) the user [search queries] and (2) the chrome extension (which the user uses). This server also displays the results
- the chrome extension which is used to send pages selected by the user to the server, and provide another way to search the database


send pages selected by the user (and some other meta data)
Chrome Extension
Simple Python Server on PythonAnywhere
send confirmation that it has been stored

WebServer receives:
- URL
- title of page
- article/page excerpt/ highlighted text
- summary text about the webpage

<u>After at least one page saved… (can query before any data is saved, but there will be no results)</u>


Chrome Extension
search for pages (uses BM25 algorithm)
Simple Python Server on PythonAnywhere
user can view information about each page they have stored (that is relevant to the query)

<u>On the Server…</u>

When Chrome Extension sends some data to be saved

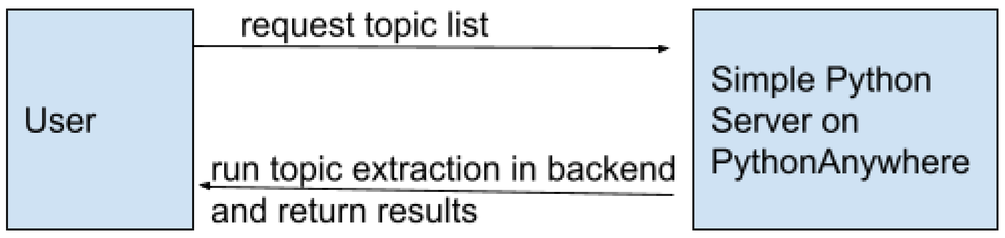
1. parse the request and retrieve the text data
2. save it on disk
3. Send back a “Successful” message or an error message to be displayed as an “alert” box (user should see this popup in the browser)

When Chrome extension sends a search query (using URL parameters) or the user types a query on the webpage.

1. parse the request to retrieve the query
2. retrieve all currently stored documents, tokenize them
3. tokenize the query
4. run bm25 algorithm, return the results

<u>After at least one page saved… (can query before any data is saved, but there will be no results)</u>

<u>How the topic extraction works</u>


User
request topic list
Simple Python Server on PythonAnywhere
run topic extraction in backend and return results

1. Load all stored documents
2. use CountVectorizer to create a matrix that represents the documents
3. Use TfidfTransformer to “Transform a count matrix to a normalized tf or tf-idf representation.”[1]
4. Fit NMF (Non-Negative Matrix Factorization) model on the data [2]
5. Get the feature names (which are part of the model in sklearn)...use this information to get the top words for each topic

[1] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfTransformer.html
[2] https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.NMF.html

How the webpage flask app code is structured (in general):

mysite/flask_app.py
- contains all of the code for the APIs, where all of the processing is done

mysite/data
- would be where the documents are stored

mysite/static
- where the icons are stored (in this case just the scissors icon)

mysite/templates
- where the html templates are stored (that the flask_app.py uses to generate the html page to show

3) Documentation of the usage of the software including either documentation of usages of APIs or detailed instructions on how to install and run a software, whichever is applicable.

Chrome Extension

Installation Instructions (these instructions are basically the same as the DL Extension installation instructions)

The Chrome Extension can be downloaded from here (410clip_extension.zip file in https://github.com/brysonli12/CourseProject/tree/main/extension) . After downloading, please follow the instructions below:

1. Decompress the extension folder, and move it somewhere permanent.
    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", and select the decompressed extension folder.
5. Toggle off “Developer Mode”
6. Click the "Extensions" icon in the Chrome toolbar (puzzle at top right of toolbar).
7. Pin "CS 410 Project: Clip 'n Search'".
8. You can now use the extension!
    a. Note that it will not work on the chrome://extensions/ page

Using the extension

1. navigate to the webpage you want to save info about
2. **optional**: highlight text you want to save
3. click on the extension
4. if you didn't highlight text before or there was some error, copy/paste some text into the highlighted text area
5. type a short sentence/description of why this is important
6. Click on "Submit to Clip 'n Search" If all goes well you should see a "Successful" popup message.

Using the website https://brysonli.pythonanywhere.com/

- can search for articles using the search bar ⇒ view results
- click on topics in nav bar to view topics

4) Brief description of contribution of each team member in case of a multi-person team.

- single person team (just myself)

Resources used

- Chrome Developer Tutorial https://developer.chrome.com/docs/extensions/mv3/
    - https://developer.chrome.com/docs/extensions/mv3/getstarted/tut-tabs-manager/
    - https://stackoverflow.com/questions/38321951/create-chrome-extension-to-send-data-into-localhost-server
    - after reading through the tutorials, also used some parts of the 410 extension (provided by the TAs)(http client) to help with the interaction with the server
- How to create a public API / Server that runs 24/7 on PythonAnywhere for FREE https://www.youtube.com/watch?v=z6q9kug0PL0
- Python3 Documentation: https://docs.python.org/3/
- https://www.digitalocean.com/community/tutorials/how-to-use-web-forms-in-a-flask-application
- scissor icon https://www.freeiconspng.com/img/25529
- BM25 ranker: https://github.com/dorianbrown/rank_bm25
- nltk treebank word tokenizer package https://www.nltk.org/_modules/nltk/tokenize/treebank.html
  Topic Extraction with Non-Negative Matrix Factorization (tutorial): https://ogrisel.github.io/scikit-learn.org/sklearn-tutorial/auto_examples/applications/topics_extraction_with_nmf.html

- CS 410 Lectures

What I completed in this project

1. **This was accomplished while doing local testing, and eventually resulted in more testing on PythonAnywhere flask app**
    a. [3-5 hours] simple proof of concept - python program with python server - scrape from selected webpage(s) and send to local python server
2. **Went through Chrome Extension Tutorial and looked at many examples. Adapted TA's CS 410 Extension. A lot of time was spent working on the APIs (simple version, then add more, then try to generalize to work with any amount of documents) and debugging any errors**
    a. [1-2 hour] look into building extensions
    b. [5 hours] simple extension (javascript) - take current webpage send to server (local server for simplicity of debugging)
    c. [3 hours] add search functionality (if the extension part was done well then this will mostly be for using BM25/search algorithm and formatting the results page, etc.)
    d. ***majority of the work went into the server, some time also spent on the extension, understanding the code, and making sure interactions between extension and server are smooth***
3. **Went through some sklearn tutorials for topic extraction, found a nice example for topic extraction which I adapted for the documents**
    a. [5 hours] add additional functionality of **topic extraction**

Software usage tutorial presentation

https://www.youtube.com/watch?v=2mwbti2gppo

https://www.youtube.com/watch?v=2mwbti2gppo
(1) sufficient instructions on how to install the software if applicable
(2) sufficient instructions on how to use the software
(3) at least one example of use case so as to allow a grader to use the provided use case to test the software.