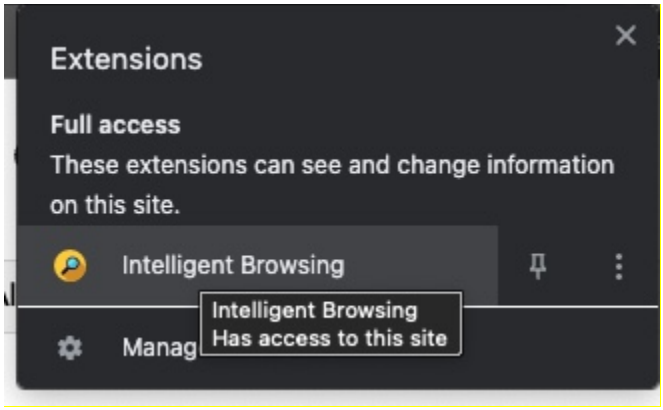
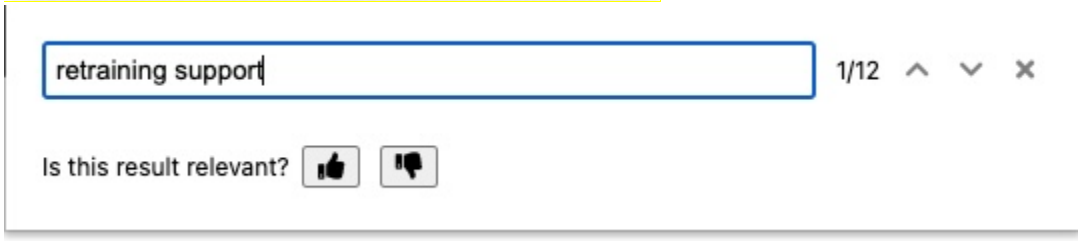
CS410 Project Progress Report – Browser Extension for Intelligent Query Matching

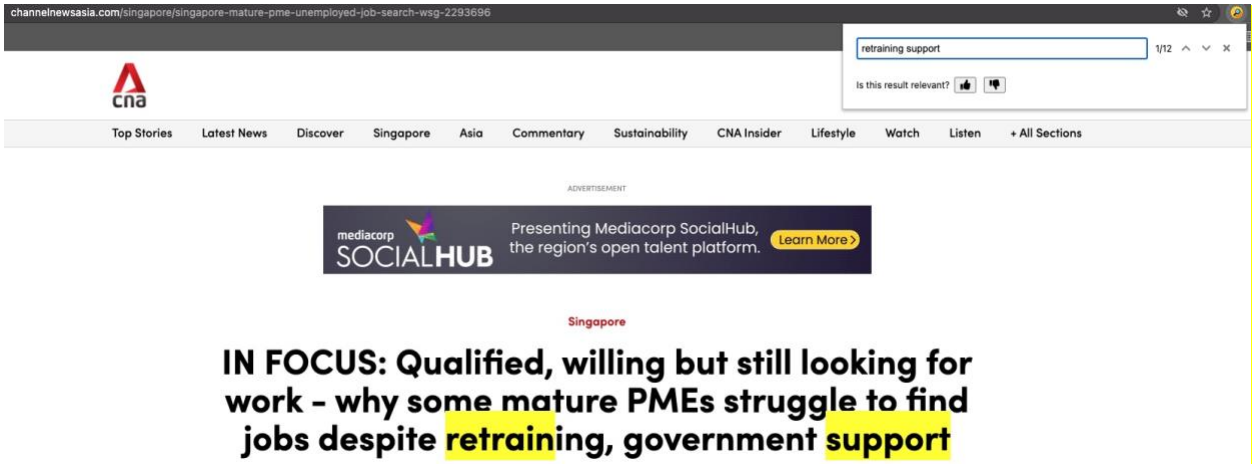
Marcus Hwai Yik Tan (leader)	Omid Afshar	Xue Ying Lin
htan8	oafshar2	xylin2

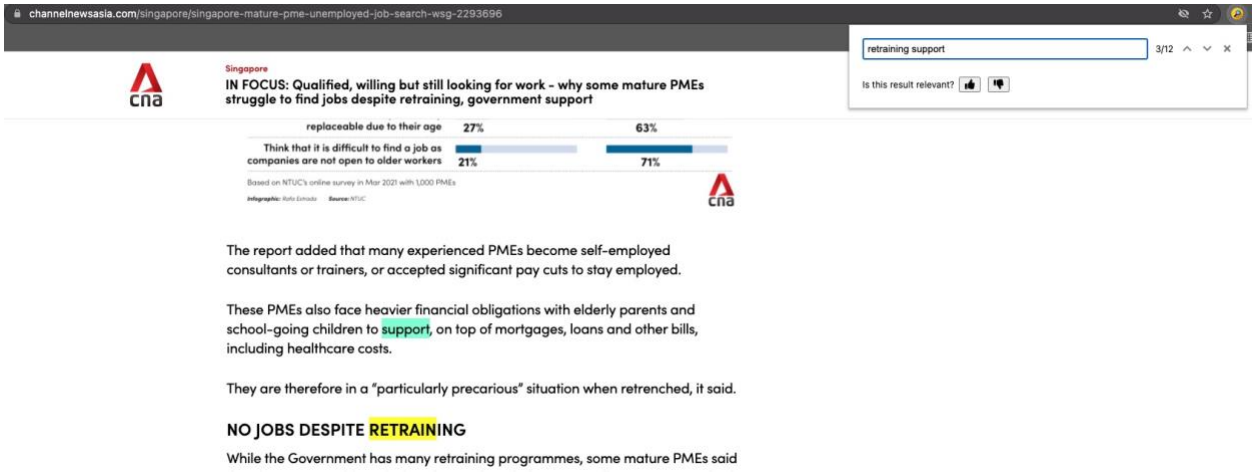
Progress Update

- Implemented the following for the frontend (Chrome extension)
  - Grab text nodes from webpage
  - Get query from user
  - Send http request to server with query and text nodes
  - Receive http response from the backend with matches and find the relevant sentences. Highlight words contributing to the match in a sentence
  - Order results by their ranking provided by the backend
  - Add next / prev buttons to navigate between result
  - Like/dislike buttons have been added to the extension pop-up, allowing a user to indicate whether a particular result returned by the extension for the query issued is relevant or not. This feature enables us to obtain relevance feedback to conduct measurements on how useful our extension is.









- Implemented the following for the backend (Python Flask and Gensim packages)
    - Receive text nodes from frontend and split them into sentences, each sentence corresponding to a document for ranking
    - Create a mapping from sentences to text nodes
    - Tokenize the query and the sentences, i.e., separate them into lists of words. Remove punctuations and special characters, make the words lower case and stem the words in the process
    - Create bag of words representations of the query and sentences
    - Use BM25 to rank the sentences
    - Find words in each ranked sentence that are matching the query words
    - Return results containing the text node indices and the relative locations of the ranked sentences and the matching words in relevant text nodes
    - We began implementing the backend logic to keep track of the relevance feedback issued for search queries. This logic keeps track of the document and corresponding search query issued on the document to log the user's relevance judgments for each result item issued by our application for the query. This log of relevance judgments can be later used to assess the overall utility of our application.

Remaining Tasks

- In order to better assess the performance of our extension, we want to surface an option for the user to toggle between conducting an exact match or BM25 ranking search. We will potentially add more ranking options as well.
- We will complete the implementation for recording user relevance judgements on the backend. As mentioned previously, each user judgment will be logged with the following information: website, query, result_ranking, relevance judgement (i.e. relevant or non-relevant).
- Calculate MAP
- Fix rare bug where sometimes a matching text is in dom but isn't visible

Challenges/Issues

- Setting up the frontend with limited and obsolete online documentations and templates
    - Templates for older versions of the extensions no longer work with current extension (Manifest 3.0)
    - Extension needs to do some work in popup js and other work in page js, and communicate between the two with executeScript
- Highlighting sentences with different ranks in each text node. Highlighting should be done in a single pass since previous highlighting would be removed in the next pass
- There are text nodes in DOM that might not be visible on screen - don't want to include these in the result, but sometimes it's hard to filter out.
- Highlighting is complex since it involves inserting spans into the dom and keeping track of how it changes offsets
- Figuring out what information needed to be included in network requests from the frontend to the backend to ensure that user relevance judgments would be grouped together properly was non-trivial. Initially, we considered generating a unique ID with each user "session", but since this would reset each time the user closed and re-opened the extension, we opted instead to rely on the website and search query as the basis of a "session".
- Lack of prior experience using flask to receive and send information to the frontend
- Hard to figure out when ranking order is wrong whether bug is in javascript or backend code