BM25 Page Search Chrome Extension

Lowell Tyner - Functionality and Code Documentation

Functionality

With the extension installed and enabled, clicking on the extension's icon will display a popup. This popup has a text entry box and search button. This allows the user to search for text on the current tab's webpage. A list of matching results will be displayed in the popup. When performing a search, the extension attempts to get all visible text on the page, then for the purposes of BM25 ranking, treats each word as a document. This was done to return multiple results since technically we are only focusing on one 1 document (the current webpage). Though not as useful as originally intended, for a POC (proof of concept), it achieves the goal of using a BM25 search algorithm on page text and displaying the results. Note, the extension must be used on an actual web page, not an internal Chrome page (one starting with *chrome://…*). If no results are found on the page, the popup will indicate this.

Implementation

The project is a basic Chrome extension with its logic written purely in JavaScript. Not wanting to rewrite the BM25 algorithm from scratch, I found a JS implementation on **npm** that appears to be recently maintained, documented and tested called Wink: https://www.npmjs.com/package/wink-bm25-text-search. This package expects documents to be in JSON format. Since we are treating each word as its own document, once all the page's visible text is retrieved, an array of individual words are converted to JSON objects, each representing a document with format "{content: 'word1'}". When we configure the bm25 "engine" we need to define field weights - but each document consists of just one field - "content" so a weight of 1 gets defined (only because this is a required step, but the weight has no bearing). We define a prepTask that can handle stop words. All JSON docs get added to the engine, then consolidation is performed (a required step after adding docs) and we can then search for the user entered query. Any subsequent queries on the same page won't re-index all the page's content.

As for BM25 search parameters, by default Wink uses values for k1, b and k as 1.2, 0.75 and 1 respectively. Currently we are not modifying the defaults, but we could when performing "engine.defineConfig()" which is the same place where field weights get defined.

Build / Installation

As is present in the package.json file, I installed the `wink-bm25-text-search` package by running `npm install wink-bm25-text-search --save`.One problem with using a

Node based JS package for a Chrome extension is that it is not written with being run in a client-side (browser) in mind. For example, the first line of Wink's example code uses the *require* syntax, which is not vanilla JavaScript (it's for Node). Therefore I had to find a way to convert it, for this purpose I went with **browserify**. The majority of the logic for my extension is located in the popup.js file, but it has to be converted to another js file (after installing browserify, using the command *browserify popup.js -o bundle.js*) which is then referenced in my html file. This step is not required for installation since my code includes the *bundle.js* output file, but something to keep in mind for future development. Anytime popup.js gets modified, *browserify popup.js -o bundle.js* needs to be run to see the changes take effect in an updated *bundle.js* file.

To install the extension in Chrome, simply navigate to "chrome://extensions/" and click "Load unpacked" and select the top level "src" directory of the project obtained from Github. Then navigate to any webpage and perform a search as described above.

Presentation

https://mediaspace.illinois.edu/media/t/1_kcdyqsoq