CS410 Course Project – Project Documentation

Name: Yibo Li
NetID: yibol2
Email: yibol2@illinois.edu
Team Leader: Yibo Li

Project Description

The project proposal is to create a Chrome extension to improve search experience for current web page. The user can have the similar search experience in the current page by typing in words. The extension will index and retrieve paragraphs in the current web page.

Project Documentation

1. Documentation of how the software

The majority of our functionality is contained within the content.js and popup.js files. The popup.js file accesses the html elements defined in popup.html. This includes a text input for our search term, a button to initiate the search, and a text box to contain the results of our search. When the button is clicked, we send the information from the text input as a message to the content.js file.

In the content.js file, we listen for a message to be sent from popup.js with the information for the search. Then we send that search term into our findText() method as a parameter. From there we normalize our search term to lowercase for easier querying and send it off to a thesaurus API. The results of the API can either contain an array of synonymous terms or it can return null. If the API does return data, we concatenate all the possible synonymous search terms into one large array named syns, and pass it to our helper() method. Note that regardless of synonyms being returned by the API, we still search for the original term provided by the user.

In helper(), we search through each HTML element of the current web page and match the terms in syns with the text contained in those HTML elements. We change the background color of the HTML element to red and store the sentence containing the search term in outputArray. Finally, now that the outputArray has been filled and the various HTML elements have been highlighted for the user, we create a string containing the number of search results and a list of all outputArray elements and store it in chrome.storage.local for use in popup.js.

After sending the message to content.js, popup.js enters a sleep() method that simulates a sleep statement in many other coding languages. This allows the chrome storage to successfully update the information within it before popup.js prints the results found within that storage. The code inserts the results string as HTML into our extension popup so users can view a bulleted list of the results and better understand how their term and its synonyms are used within the document.

2. 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.

https://www.dictionaryapi.com/products/api-collegiate-thesaurus

We use this API described in the link above to search for query synonyms. We just have to pass a query word in an api link with our api key in our link to gain access to an array of synonyms of our query. We also used the FETCH api to make the request towards the web server. The code of us using the api is in our content.js file in the findText function. We call our API every time we use our extensions button to activate its functionality.

Video Link: https://www.youtube.com/watch?v=9VvweLK5e8U

Example use case shown in video:

https://en.wikipedia.org/wiki/Black%E2%80%93Scholes_model

https://en.wikipedia.org/wiki/Document_retrieval