Assignment 8 Prototyping your Final Project (Part 2)

a. **Part 1**: The purpose of my Chrome extension is to help the user discover what other people are talking about the content they're currently reading. It does this by getting keywords from the webpage they're on and displaying Tweets with these keywords as hashtags. The Tweets include the users who posted them (name, username, and profile picture), the actual body of the Tweet, and the first image in the Tweet if the Tweet has pictures. This information is engaging because the Tweets being displayed would change every time the user changes to a new tab, and even opening the extension on the same page can produce different Tweets since it always pulls the most recent Tweets. The target audience is anyone who wants to stay connected and likes to absorb information with people, not just on their own. Personally, I always like reading comments after I read something, like YouTube comments, Instagram comments, and Facebook comments, and now I could see comments on articles that don't have comments themselves!
b. **Part 2**: The interactions with this Chrome extension is very straightforward - there are only three:
    - When first installing and opening the extension, you can click the puzzle button at the top right of your Chrome browser and click the "Login" button to be redirected to sign in to Twitter.
    - After signing in, you could go to the page or article you want to generate Tweets for and click on the extension at the top right again. After waiting for a bit, the popup would populate with 10 Tweets with topics from the article.
    - You could scroll down in the popup to see more Tweets.
c. **Part 3**:
    - Chrome extension (https://developer.chrome.com/docs/extensions/): I chose to use this because I wanted to learn about how to make a Chrome extension - it seems cool! It's also the premise of my whole project (this answers how I used it and what it adds to the project)!
    - Extractor API (https://extractorapi.com/): I chose this because I needed a way to convert a website HTML to a paragraph of text to prepare for keyword extraction. I used it right when you open the extension popup - I get the active tab the user is on in the Chrome browser and input that to the API. The API then gets rid of all the HTML tags and discards irrelevant information that doesn't pertain to the actual body of the article. This API adds to the Chrome extension because most articles online have many ads and irrelevant information that can be hard to get rid of just programmatically with JavaScript (as I mentioned and showed examples of in the YouTube video linked at the bottom of this document).
    - MonkeyLearn API Keyword Extractor (https://app.monkeylearn.com/main/extractors/ex_YCya9nrn/tab/api/): I chose this because now that we have a cleaned up text, we need to find the most relevant

words in that article. This is still a very hard problem to tackle because we can't just take the most frequently appeared words - these words are often common ones like "of" and "the". For this context, I used this API by giving it the text I got back from the Extractor API (I cleaned it by removing all quotes ("") since those can be mistaken as strings and can mess up the MonkeyLearn API). I get in return a JSON object containing all the keywords. I then take the 5 most frequent keywords and display them at the top of the Chrome extension popup. The MonkeyLearn API adds to this extension because as I've mentioned, keyword extraction is still an ongoing problem computer scientists from all over the world are still trying to figure out. Getting keywords is very important to giving the user the best experience with the extension and actually bringing up topics discussed in the articles they've looked at.

- Twitter API (https://developer.twitter.com/en/products/twitter-api): I chose to use this API because it helps us search for Tweets that are prevalent to the topics we've determined with the MonkeyLearn API. I used it first when the user opens the Chrome extension for the first time, to log in and get the authentication details. I then also use it when I get the top 5 keywords back from the MonkeyLearn API and search Twitter for the most recent Tweets that are relevant to the keywords (I can't get the most relevant Tweets; those are reserved for those certified to be using the endpoint for research purposes). After getting back those Tweets, I then display them on my Chrome extension popup. This adds to my extension because then I could get details about the Twitter user that made the Tweet, making it feel more like the user is discussing with real people.
- https://github.com/lambtron/chrome-extension-twitter-oauth-example: I found this public GitHub page when I was looking up how to make API calls from a Chrome extension. I used it by setting up my API calls, especially the authentication part of verifying a user's Twitter login. I modified it by letting it call the Twitter API search endpoint as well. This tool adds to my project because a Chrome extension doesn't work exactly like normal Javascript - you can't "require/import" libraries that help you make API calls.
- jQuery: I know this isn't technically a new tool for Lab E, but I wanted to mention it because this was the first time I was using it for a big project. I found that jQuery's AJAX was very helpful for making API calls, and jQuery is also useful for iterating over arrays and objects. It definitely helped me make some GET requests that I could not figure out what was going wrong for the life of me!

d. **Part 4**: I changed a lot from my Assignment 7 - I ended up realizing that what I had planned was very ambitious of me for the timeframe I had to complete the project in. The main difference that you could probably see is that instead of having the Tweets come from a sidebar on the right, I just have the Tweets appear in the popup after clicking on the Chrome extension. I also removed the ability to remove and add back keywords to

search for in the Twitter API, and removed the ability for the user to Tweet about the topic directly from the Chrome extension itself. I did this because it was already a lot of work; I spent hours and hours trying to figure out API calls on all the APIs I used, especially for a Chrome extension which doesn't have many resources.

e. **Part 5**: As I've already briefly mentioned, it took a super long time trying to get API calls to work from a Chrome extension. The main thing is I couldn't figure out how to debug it and see what was printed to my console. Eventually, I figured out that I could first open the Chrome extension popup, then right click and press "Inspect" (instead of inspecting the background page itself). I also spent time figuring out how to convert the responseText I got back from an AJAX API call into a JSON object, and each API had its quirk with how the responses worked. Overall, though, I'm really satisfied with how everything turned out - nothing feels better than looking at the result and feeling proud at how it works.

GitHub repo: https://github.com/agliyt/TweetsfromArticle (instructions on how to install it are in the README)
YouTube link: https://youtu.be/-mim4KOIhUk