**HW 9**

**Learning Experience**

**Part 1:**

The first part of the article explains what a search engine is and how it works. Basically, a search engine is a program that collects the user input as a query and matches in the database so that it can output the results to the user. The article gives us a motivational example about how Google search works, which intrigued me to read more about in this article. Additionally, the point of the article was to create a search engine for TED talks from scratch. The first step was to download and install certain packages such as Django, Scikit-Learn, Numpy, NLKT. I decided to install one by one since I installed some of them and they are an updated version. So, installing again in their previous version would give me errors. I used the command prompt in order to install with the following command



where (package) can be any package to be installed like Numpy, Matplotlib, etc

The first part was to build the database which was done by using pandas to read the transcript as a csv file

The next step was to somehow relate the user search with the transcript in the database. Now, here is where articles become complicated. To do so, the articles introduce the mining technique called TF-IDF, which is a feature also used in Google’s earliest day. As the article mentions, the TF-IDF “is basically a statistic used to evaluate the importance of a word to a document in a collection.” The TF-IDF calculates that statistic by using some math to calculate the importance of a word in a document.

However, they are some issues with the documents as the article stated. Since they are some words with same meaning but written differently, the program cannot count those words because computer cannot tell the difference between them. That is when Stemming and Lemmatization come handy to fix this issue. In brief, this is actually to get rid of ‘-s’ or ‘-es’ of plurals, ‘-ing’ and ‘-ed’ in words. After getting rid of these, we have solved the problem.

Now, the normalized term frequency and the inverse document frequency are concept that I did not quite understand. I believe they are algorithms that sharpen the search engine to be more accurately.

The article also explained how the author visualize with a vector as she called ‘cosine similarity.’ What understood is that this algorithm match more accurately the query and the transcript so that when user input the query, this function will prompt the better result.

Overall, my thoughts are that search engine is a very interesting concept to learn and I wished the author would have posted the second part of this fascinating tutorial on how to build a search engine for TED talks.

**Part 2:**

The second article introduces a tutorial on how to create a World Cloud. In order to build a world cloud, we need words. The article extracts these words from Wikipedia by using the Wikipedia library. Once the text is extracted, we need to clean them up by deleting unnecessary special characters.

Next step, the articles shows how to plot our words by using the matplotlib package. We create a function called plot\_cloud which passes a wordcloud object. This create the background for our cloud word design.

Then, we create another function to create a word cloud object. Here is where we personalize our word cloud by setting the size, color background, color of the words, duplication of words, removing undesired words.

Finally, we pass this word cloud object into our function plot\_cloud to display our word cloud. The result is appealing!

**Part 2 Test 1**

**Text

Description automatically generated**

**Part 2 Test 2**

**Text

Description automatically generated**