Assignment 3 Report

- **Task 1: Crawling Documents**

I. *Canonicalization:*

We Canonicalized the given links such that:

1. All the links are converted to lowercase.
2. Parsing the URL that is splitting an HTTP URL into host and path components.
3. Making relative URLs active.
4. Removing unwanted fragments from the URL. For example, removing data after "#" from given link www.example.com/a.html#anything-> www.example.com/a.html ->
5. Removing duplicate slashes like for given link http.//www.example.com//a.html -> http.//www.example.com/a.html.
6. We are crawling only .html documents and the pages which have content language as "en".

II. *Politeness Policy*:

We have taken care of the politeness policy in our code, which means that our crawler strictly obeys the politeness policy and we send each ping to the site in a gap of 1 second.
If the policy is not taken care, is it can harm the web sites, we are crawling and cause the web site administrators to block the respective IP address from which we are crawling.
We are using "urlib.roboparser" to keep a check on the same policy.

III. *Frontier Management*:

The frontier is the data structure we are using to store pages we need to crawl, and, in our code, we are using priority Queue for creating the frontier.

As mentioned, we have multiple URLs so we need to give each of them priority and amongst all the URLs we need to crawl the top 40,000 URLs only, thus arrives the need to give priority to each of them.

We have decided the priority of a URL on various aspects as follows:

1. *Scoring*: We can give each URL score based on the key terms that exist in our given seed.
   For example, my seed for this assignment is Governors of Massachusetts and specific governor that was assigned to me was "Mitt Romney" thus the scoring I gave for creating priority of the URLs was:

"mitt" : 3, "romney": 3, "businessman": 2, "lawyer" : 1, "politics" : 2, "massachusetts": 0.5, "bishop" : 2, "government":1, "boston" : 0.5

Total sum: 15

Keyword matches: Sum (number of keywords found in domain) / Total sum.

Thus, by using this method we get the score of each URL and we assign them in the Priority Queue based on the highest to lowest score.

2. We are also scoring and saving the wave number and anchor test as well.
   Wave number: Is the number of the base URL.
   The above scoring method uses Graphs and is used for calculating the distance between the link and other outlinks produced and then generating the priority of the URLs accordingly.

"mitt" : 3, "romney": 3, "businessman": 2, "lawyer" : 1, "politics" : 2, "massachusetts": 0.5, "bishop" : 2, "government":1, "boston" : 0.5

Achor_text (Base URL): Sum of anchor keywords

- **<u>Task2: Link Graph</u>**

  Inlinks: Inlinks, also known as inbound links or backlinks, which refer to hyperlinks on other websites that lead to a particular webpage or website.
  In other words, inlinks are links that direct users to a specific webpage from other websites.

  Outlinks: Outlinks, also known as outbound links, are hyperlinks on a particular webpage that lead to other web pages or websites.

In our code to fetch the inlinks and outlinks for a given url we are using BeautifulSoup library.

We have created a Json file "savedata.json" which stores the inlinks and outlinks for the given url. Please note that the the given json file is not preprocessed yet.

Preprocessing data:

- After creating json file, we use parsed data file along with the queries to preprocess the data. Below are the steps included in preprocessing the data:
- We do preprocess of the data in two steps:
  1. We stem the parsed data file along with the queries and remove stop words from them.
  2. During the class creation we load the stop words and remove the stopwords from parsed data and queries.
  3. Then we convert the text to lower case and split the text using regex to break any text other than continuous characters or separated by single period. ([^A-Za-z0-9.]+|\.{2,})

- **Task3: Merging team indexes:**

Merging is a process where the crawled documents of all the team members are combined, and then we further create indices for all the crawled documents together.
After crawling the documents are in the provided format, that for each link, its respective inlink, outlink and name of the author is stored, we upload all of them to Elastic cloud.
We are 4 team members working on current assignment thus we had approximately 170,000 documents to be combined.
As suggested all the team members uploaded the crawled documents to the Elastic cloud and we performed the merging documents on the cloud.

After this we performed Indexing on the entire data corpus of approximately 170,000 documents on Elastic Search tool itself.

- **Task4: Vertical Search:**

Vertical search, also known as specialized search, it is a type of search engine that focuses on a specific topic or domain. Unlike general search engines like Google or Bing that cover a wide range of subjects, vertical search engines specialize in a particular industry, field, or type of content.
Vertical search engines are designed to provide more relevant and accurate results for users searching for specific types of information, making them a valuable tool for researchers, professionals, and people looking for highly specific information.
So, we have created a webpage using Python and its library streamlit which is linked to our Elastic Cloud and makes an API call to the cloud whenever a search action is performed.
So, whenever a user performs a search action then an API call is made to the Elastic cloud, and it uses the Indexed document for efficient and fast searching related only to the specific domain.