EECS 767 – Information Retrieval Project

Web Search Engine

Team Name: Data Devils

Nishil Parmar (2917887)

Shruti Goel (2927572)

Contents

Introduction

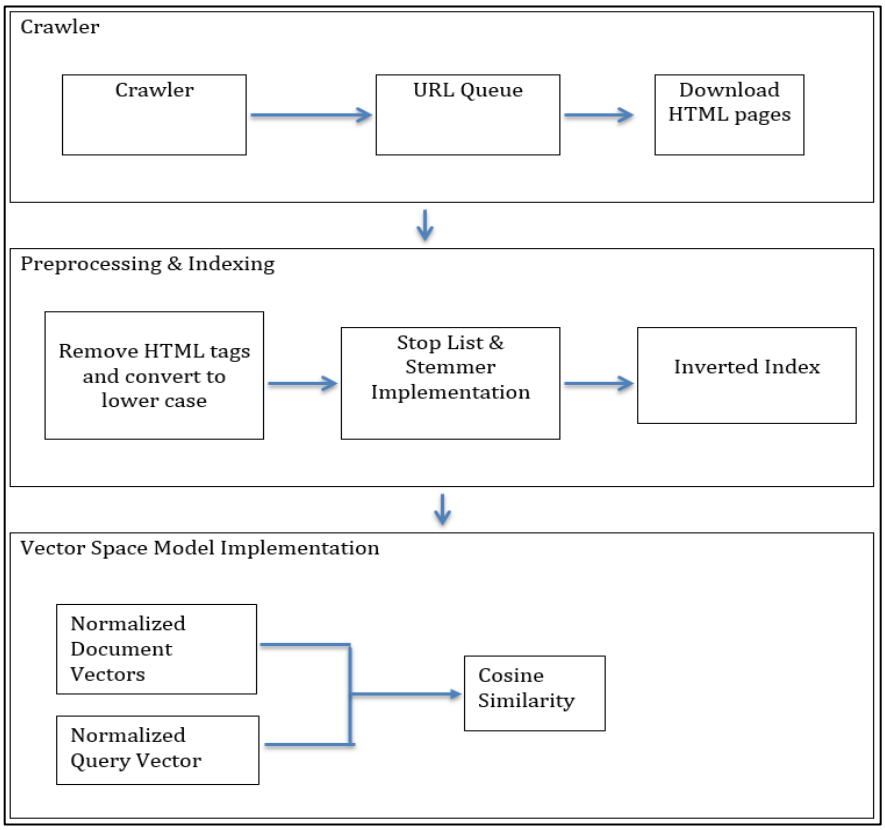
Web search is designed under this project to fulfill user's information need. The user can enter the query in the search bar on the web interface and expect top 7 most relevant ranked results. This is basically a static search engine where we have crawled about 3000 web pages from the seed domain – https://www.techcrunch.com. The corresponding html web pages were downloaded for further processing and computing the results. The main modules of this project include – crawling, preprocessing, tokenization, Indexing, Vector Space Model and web interface.

Platform specifications

- Coding Platform - Python 3.6.5

- Web Interface – frontend: Bootstrap + jquery | Backend: Python + Django

Overall Web search Engine Architecture



Implementation Modules

1. Preprocessing & Indexing

a. Preprocessing

- Downloaded HTML web pages (from the crawler) are saved in a folder and corresponding file names for the links are saved in a separate text files (which is used for returning the final results on the web interface)
- The preprocessing of the html pages was performed by utilizing the python module HTMLParser to remove the unwanted html tags
- After converting the text into lowercase, we have implemented stop words and stemmer using NLTK package of python
- Finally, the files are stored as tokens in text files for further computations

b. Indexing

The sequence of terms in each document (after tokenization), tagged by their documentID is sorted alphabetically. Instances of the same term are then grouped by

word and then by DocumentID, the terms and DocumentIDs are then separated out. The dictionary stores the terms and has a pointer to the postings list for each term. Each postings list stores the list of documents in which a term occurs, term frequency and the position of the term in each document.

Data Structures:
- index = {term0:[df, tf(all), posting_list],...}
- posting_list = {'file1': [[positons1],tf],...}

Steps:

- The inverted index is the main data structure of our search engine. We used a Hash table (python's dictionary) to store the inverted index in memory. The reason is we perform a lot of lookups (one for every term in the document), and we also add lots of keys (every term is a key), so we want these operations to be very efficient. Since Hash tables have average O (1) lookup time and amortized O (1) insertion time, they are very suitable for our needs.
- Every term is a key in our dictionary, whose value is its postings list (the list of documents that the term appears in). We also store positional information of the term in the postings to answer phrase queries.
- We build an inverted Index for the current page first and then merge the index of the current page with the main inverted index.
- We continue and build our main inverted index for the whole collection and then save this index to a pickle file.

Preprocessing & Indexing Flowchart

2. Vector Space Model

Our Vector Space Model has a simple architecture. After the tokenized document files are generated, we create a vector for each document, in that way in our document vector we eliminate the dictionary terms which has zero tf-idf, which helps us create smaller document vectors than the conventional method. During the process of creating document vectors we normalize all the created vectors and store them as a pickled file. At this point a pickle file (which represents normalized document vector) is created for each tokenized document. When the user feeds query into our search engine, a normalized query vector is created and then is checked against the previously created document vectors to calculate cosine similarity. Once the cosine similarity is calculated, it is used to rank documents in the search engine. We rank top 7 documents in decreasing order of cosine similarity.

3. Web Crawler

A web crawler is an automated program that crawls through the web domain to fetch and parse the html and extract links.
We have implemented a multithreaded crawler for crawling the web. The seed URL for this project is the famous news website "Wall Street Journal" – https://www.wsj .com. But, we are going to make this dynamic, meaning user can enter the domain to be searched.

Crawler features implemented

1. **Implicit politeness**: In order to be polite, we have added a wait time(sleep time) before it accesses the host website again to fetch links
2. **Explicit politeness**: We have cached the robots.txt file of the domain to make sure that only allowed URLs are added to the frontier queue for further processing.
3. **Domain restriction**: We have restricted the crawling domain to techcrunch.com only to limit the number of fetched URLs
4. **Multithreading**: This is a multithreaded spider for fetching and parsing the webpages (currently with 12 threads) and has been tested for scaling up.
5. **URL depth restriction**: Currently, we are have added a depth limit of 3000 links to the crawled queue.

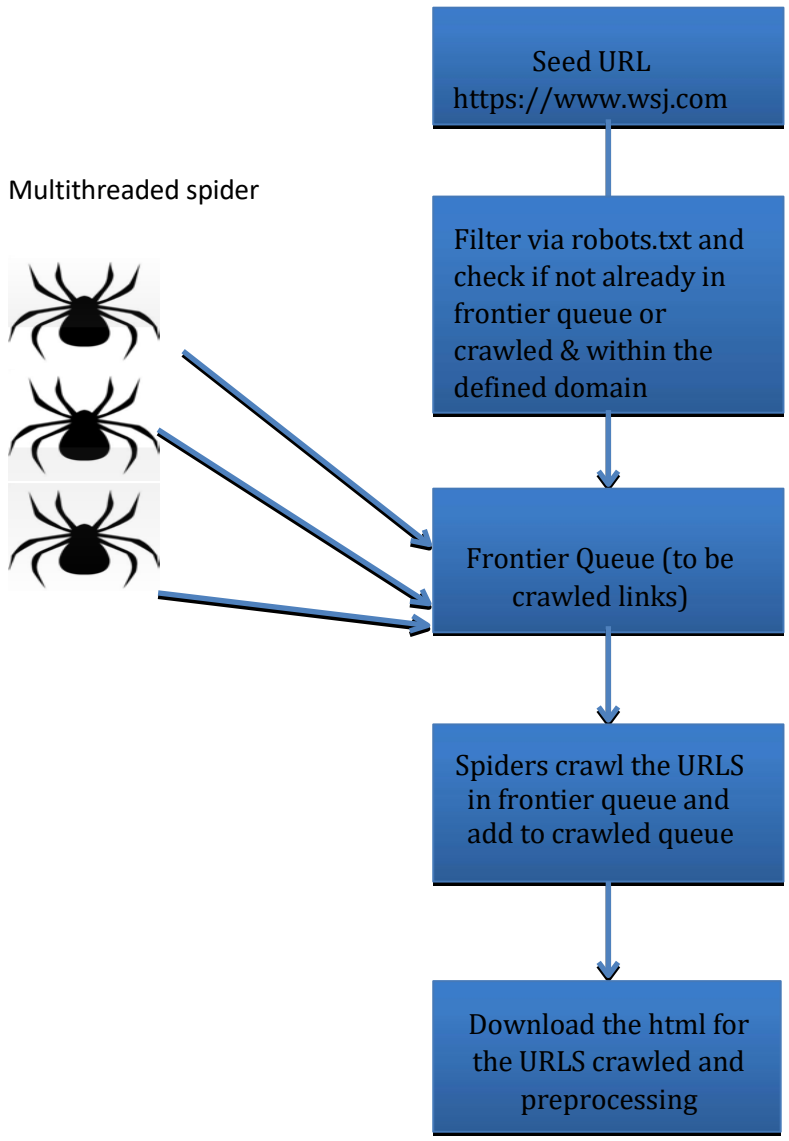
Web Crawler Program Flow

1. The spider fetches the seed URL from the frontier queue and parses the html page to fetch further URLs. We are fetching the absolute URLs in the domain and also handling the relative URLs.
2. The fetched links are checked via for allowed access through cached robots.txt for the domain. If it passes this filter, it is checked if it is not already present in the frontier queue or crawled queue. If not, the URL is appended to the frontier queue text file and ready to be crawled.
3. The spider maintains two queues – frontier queue and crawled queue. Both the

queues are stored as text files.

4. To be polite to the web, we have added a sleep time for the crawler so that it waits before starting to crawl again.
5. After crawling successfully, the corresponding URL is moved to the crawled queue.
6. The crawl depth is set to 3000 URLs

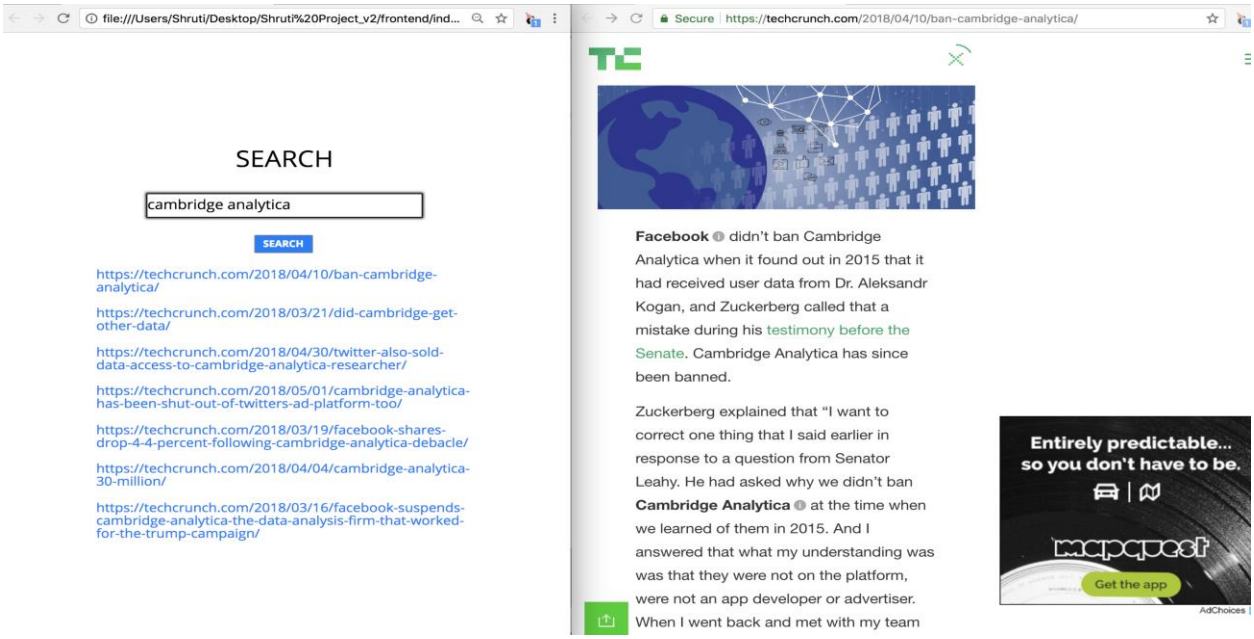
Web crawler architecture



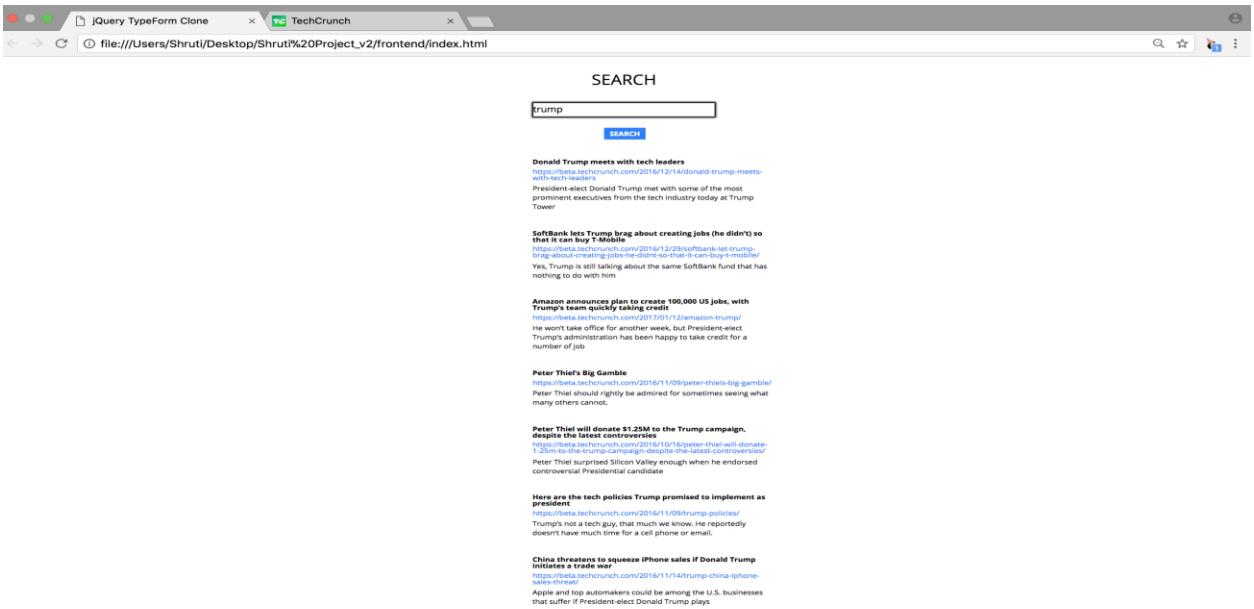
4. Web Interface

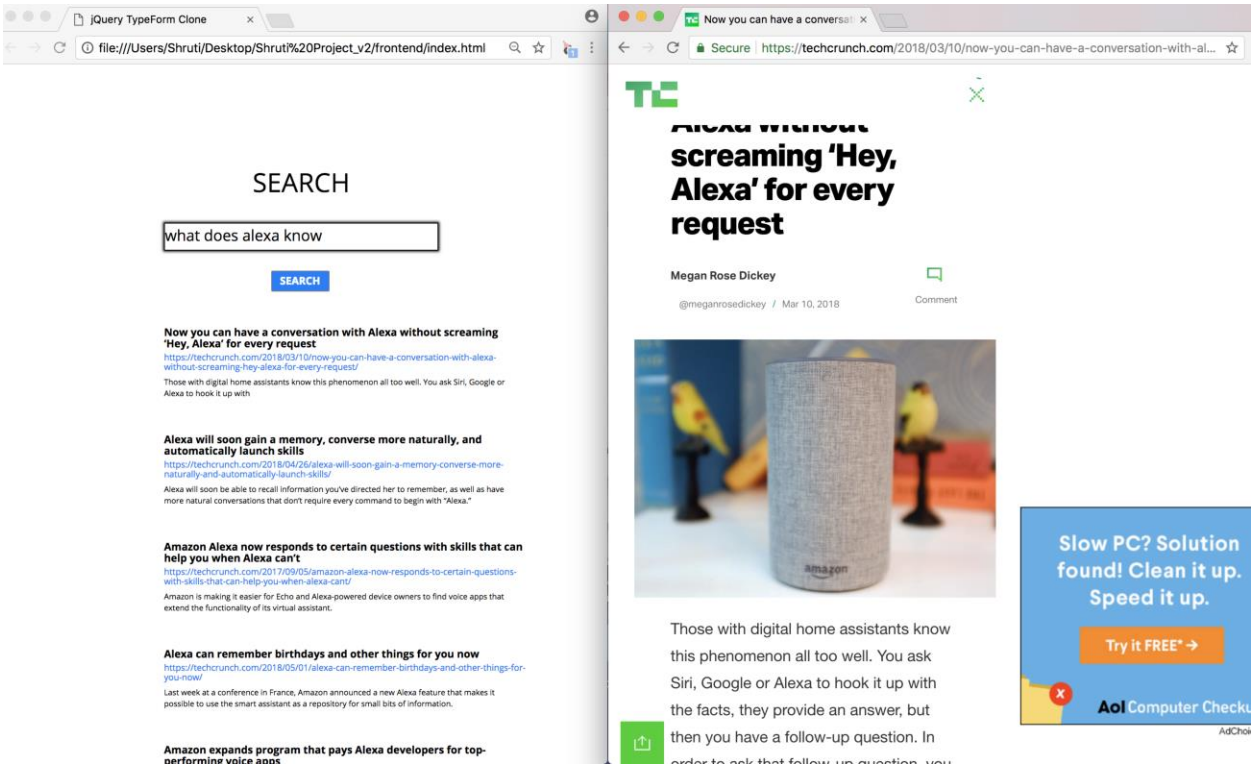
Web Interface is broadly divided into two parts, frontend and backend module. Frontend comprise of Bootstrap and jquery. User can enter the search term through the frontend and based on search query, results are returned to user. The backend uses Python with Django as framework. User hits an AJAX request from frontend, which is received by the Django server. Django server computes the query and returns the results.
We hosted our local server using ngrok tool.

1. Initial version of search engine – v1



2. Next version – v2 – improved results display


jQuery TypeForm Clone
TechCrunch
file:///Users/Shruti/Desktop/Shruti%20Project_v2/frontend/index.html
SEARCH
trump
SEARCH
Donald Trump meets with tech leaders
https://beta.techcrunch.com/2016/12/14/donald-trump-meets-with-tech-leaders
President-elect Donald Trump met with some of the most prominent executives from the tech industry today at Trump Tower
SoftBank lets Trump brag about creating jobs (he didn't) so that it can buy T-Mobile
https://beta.techcrunch.com/2016/12/29/softbank-let-trump-brag-about-creating-jobs-he-didnt-so-that-it-can-buy-t-mobile/
Yes, Trump is still talking about the same SoftBank fund that has nothing to do with him
Amazon announces plan to create 100,000 US Jobs, with Trump's team quickly taking credit
https://beta.techcrunch.com/2017/01/12/amazon-trump/
He won't take office for another week, but President-elect Trump's administration has been happy to take credit for a number of job
Peter Thiel's Big Gamble
https://beta.techcrunch.com/2016/11/09/peter-thiels-big-gamble/
Peter Thiel should rightly be admired for sometimes seeing what many others cannot.
Peter Thiel will donate $1.25M to the Trump campaign, despite the latest controversies
https://beta.techcrunch.com/2016/10/16/peter-thiel-will-donate-1-25m-to-the-trump-campaign-despite-the-latest-controversies/
Peter Thiel surprised Silicon Valley enough when he endorsed controversial Presidential candidate
Here are the tech policies Trump promised to implement as president
https://beta.techcrunch.com/2016/11/09/trump-policies/
Trump's not a tech guy, that much we know. He reportedly doesn't have much time for a cell phone or email.
China threatens to squeeze iPhone sales if Donald Trump initiates a trade war
https://beta.techcrunch.com/2016/11/14/trump-china-iphone-sales-threat/
Apple and top automakers could be among the U.S. businesses that suffer if President-elect Donald Trump plays


jQuery TypeForm Clone
file:///Users/Shruti/Desktop/Shruti%20Project_v2/frontend/index.html
SEARCH
what does alexa know
SEARCH
Now you can have a conversation with Alexa without screaming 'Hey, Alexa' for every request
https://techcrunch.com/2018/03/10/now-you-can-have-a-conversation-with-alexa-without-screaming-hey-alexa-for-every-request/
Those with digital home assistants know this phenomenon all too well. You ask Siri, Google or Alexa to hook it up with
Alexa will soon gain a memory, converse more naturally, and automatically launch skills
https://techcrunch.com/2018/04/26/alexa-will-soon-gain-a-memory-converse-more-naturally-and-automatically-launch-skills/
Alexa will soon be able to recall information you've directed her to remember, as well as have more natural conversations that don't require every command to begin with "Alexa."
Amazon Alexa now responds to certain questions with skills that can help you when Alexa can't
https://techcrunch.com/2017/09/05/amazon-alexa-now-responds-to-certain-questions-with-skills-that-can-help-you-when-alexa-cant/
Amazon is making it easier for Echo and Alexa-powered device owners to find voice apps that extend the functionality of its virtual assistant.
Alexa can remember birthdays and other things for you now
https://techcrunch.com/2018/05/01/alexa-can-remember-birthdays-and-other-things-for-you-now/
Last week at a conference in France, Amazon announced a new Alexa feature that makes it possible to use the smart assistant as a repository for small bits of information.
Amazon expands program that pays Alexa developers for top-performing voice apps
Now you can have a conversat
Secure https://techcrunch.com/2018/03/10/now-you-can-have-a-conversation-with-al...
screaming 'Hey, Alexa' for every request
Megan Rose Dickey
@meganrosedickey / Mar 10, 2018
Comment
Those with digital home assistants know this phenomenon all too well. You ask Siri, Google or Alexa to hook it up with the facts, they provide an answer, but then you have a follow-up question. In
Slow PC? Solution found! Clean it up. Speed it up.
Try it FREE* →
Aol Computer Checku
AdChoi

General Statistics:

- Documents crawled: 2934
- Dictionary Size: 23k words
- Index size: 16 MB
- Document Vectors: 15 MB
- Time to build vectors and Index: 17 minutes