Term Paper on

**Web crawling**

[2020]

Submitted by

**Shirsangshu Dutta**

**Roll no. 001811003002**

PG – ME in Software Engineering

Dept. of Information Technology

**Jadavpur University**

Salt Lake Campus, Kolkata

Under the Guidance of

**Mr Uttam Kumar Roy**

H.O.D

Dept. of Information Technology

**Jadavpur University**

**Acknowledgement**

As the end of my 2nd st year in M.E. Software Engineering is approaching I look back at the past days and can’t help myself from thinking how fortunate I was that I had been given an opportunity to be a part of the family of Jadavpur University. Throughout the year I havent learned a lot of new things and that had also helped me to become a calm and patient student who wants to learn more. Thanks to Dr. Uttam Kumar Roy for guiding me in this pursuit of knowledge

**Contents**

Acknowledgement…………………………………………………………………………………2

Contents……………………………………………………………………………………………3

Abstract…………………………………………………………………………………………… [4](#Abstract)

1. Introduction……………………………………………………………………………………[.4](#Introduction)
   1. What is web crawling…………………………………………………………………… 4
   2. Motivation of web crawling………………………………………………………………[5](#Motivation)
   3. History of web crawling……………………………………………………………………[4](#History)
   4. web scraping vs web crawling............................................................................................. [6](#CrawlingScrapingDiff)
2. Design goals of a web crawler......................................................................................................[11](#DesignGoals)
   1. Crawling Algorithm................................................................................................................12
3. Architecture of a crawler................................................................................................................[13](#BasicCrawlerArchitecture)
   1. Basic crawler architecture.......................................................................................................[13](#BasicCrawlerArchitecture)
   2. Data structures commonly used..............................................................................................[13](#DataStructures)
   3. Distributed crawler architecture..............................................................................................[15](#DistributedCrawlerArchitecture)
   4. Page visit and freshness...........................................................................................................[17](#Pagerevisit)
   5. Synchronization policies..........................................................................................................[19](#Synchronization)
4. Performance metrics........................................................................................................................[20](#OSN)
5. Crawling social networks................................................................................................................[21](#OSN)
   1. Brief about OSN.....................................................................................................................[.22](#OSN)
   2. Data collection.........................................................................................................................[22](#DataCollection)
   3. Challenges scraping OSN ......................................................................................................22
6. Demo application............................................................................................................................[23](#DemoApplication)
   1. Overview of linkedin..............................................................................................................[23](#linkedinoverview)
   2. Challenges faced......................................................................................................................[24](#challengesfacecd)
   3. Strategies attempted.................................................................................................................[25](#Strategiesattempted)
   4. How the app works..................................................................................................................[27](#HowTheAppWorks)
7. Future scope of work.......................................................................................................................[29](#Conclusion)
8. Conclusion ......................................................................................................................................[29](#Conclusion)
9. References.......................................................................................................................................[30](#References)

**Abstract**

Web crawling has become an important activity given the phenomenal growth rate of websites aided by the ever increasing mobile users.With emergence of social media platforms like facebook,twitter,linkedin it has become a rich source of information worth mining.

This paper discusses web crawling and how the technology has evolved with the growth of websites, the challenges that it has presented on how to effectively crawl hundreds of website and find just the correct information, how to keep up with the ever dynamic web contents by updating and maintaining the contents of website.

Several strategies how relevant information can be incrementally updated while having good efficiency is another point of discussion.

The discussion finally leads to crawling Online Social networks which are treasure trove of information whether it be for commercial purposes like business relationship , feedback from customers or sentiment analysis in general or digital marketing which has become very popular snowaday

**Introduction**

**1.1 What is crawling?**

The term crawling has been extensively used for automated agents for traversing WorldWideWeb graph

The web crawler aims at discovering the web pages of a web application by navigating through the application.This is usually done by simulating the possible user interactions considering just the client-side of the application

Web crawlers are also know by other names like spider,web robots,bots

Googlebot is a well know example of web crawler

* 1. **History of web crawlers**
* Web crawlers were written as early as 1993.This year gave birth to four web crawlers: World Wide Web Wanderer, Jump Station, World Wide Web Worm and RBSE spider. These four spiders mainly collected information and statistic about the web using a set of seed URLs
* In 1994 WebCrawler and MOMspider the concepts of parallel crawler,politeness and blacklists were introduced
* In next few year pioneering commercial web crawlers Lycos,AltaVisa
* In 1998 revolutionary large scale web crawler Google was released by Bin Sergy and Larry Page.

It addressed issues of scalability ,reduced disk access time through techniques such as compression and indexing.

Google calculated the probability of a user visiting a page through an algorithm called **PageRank** which calculates the probability of a user visiting a page by taking into account the number of links that point to the page as well as the style of those links.

Having this probability,Google simulated an arbitrary user and visited a page as often as the user did. Such approach optimizes the resources available to the web crawler by reducing the rate at which the web crawler visits unattractive pages. Through this technique, Google achieved high freshness

Architecturally,Google used a master-slave architecture with a master server (called URLServer) dispatching URLs to a set of slave nodes. The slave nodes retrieve the assigned pages by downloading them from the web. At its peak, the first implementation of Google reached 100 page downloads per second.

**1.3** **Motivations of web crawling**

1. Content indexing for search engines. Every search engine requires a web crawler to fetch the data from the web.As the amount of information on the web has been increasing drastically, web users increasingly rely on search engines to find desired data. In order for search engines to learn about the new data as it becomes available, the web crawler has to constantly crawl and update the search engine
2. Machine learning is fueling today’s technological marvels such as driver-less cars, space flight, image, and speech recognition. However, one Data Science professional would need a large volume of data to build a robust & reliable machine learning model for such business problems. Data mining or gathering data is a very primitive step in the data science life cycle. The gathered data can be used for various purposes
3. For other commercial purposes like gathering business intelligence like information about competitiors and potential customers
4. Related use is web archiving where large sets of web pages are periodically collected and archived for posterity

**1.4** **Crawling and scraping and their difference**

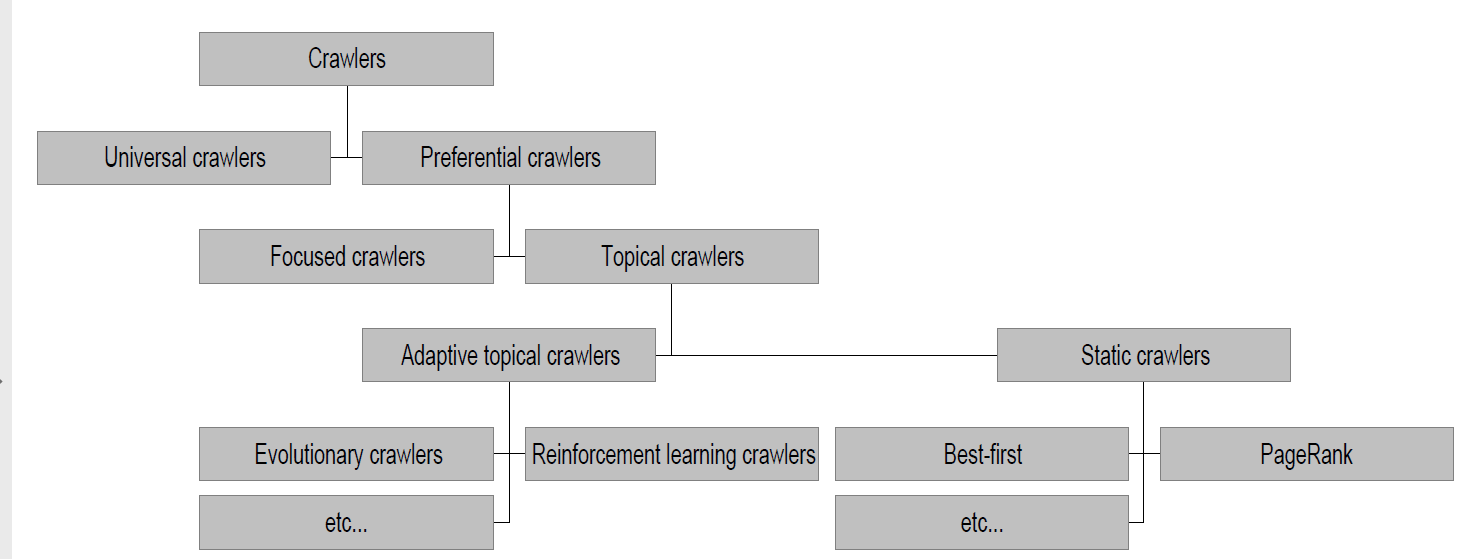
A crawler gets web pages -- i.e., given a starting address (or set of starting addresses) and some conditions (e.g., how many links deep to go, types of files to ignore) it downloads whatever is linked to from the starting point(s).

Crawlers surf the web, following links. An example would be the Google robot that gets pages to index.

A scraper takes pages that have been downloaded or, in a more general sense, data that's formatted for display, and attempts to)extract data from those pages, so that it can (for example) be stored in a database and manipulated as desired.

Scrapers and crawlers do not always distinguish .[Scraper Crawler](https://www.scrapercrawler.com/) is does both it crawls to a URL i.e. indexes all the URL in that main URL

**1.4** **Taxonomy of crawlers**

****

**Fig1. Types of crawler**

**1.4.1 Universal crawler**

This is a sequential crawler Seeds can be any list of starting URLs Order of page visits is determined by frontier data structure .Stop criterion can be anything

**1.4.2 Preferential crawlers**

Assume we can estimate for each page an importance measure, I(p) we Want to visit pages in order of decreasing I(p) Maintain the frontier as a priority queue sorted by I(p)

Selective bias toward some pages, eg. most “relevant”/topical, closest to seeds, most popular/largest PageRank, unknown servers, highest rate/amount of change,

1. **Focussed crawler**

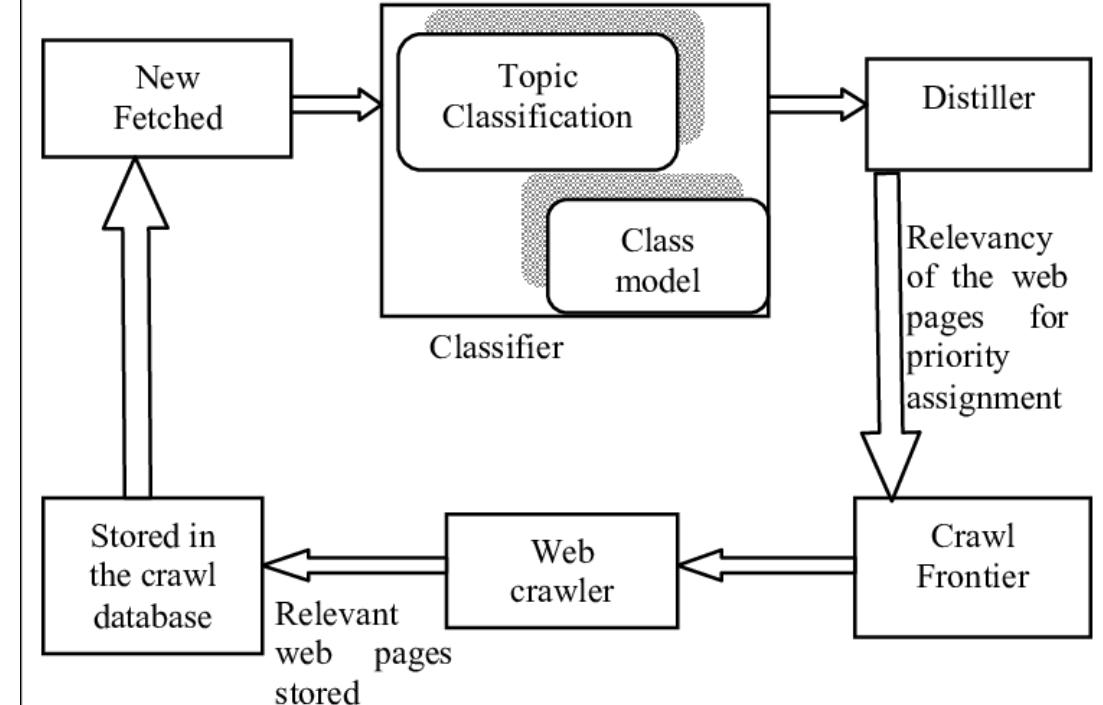


Fig 2.Focussed web crawler

A general purpose Web crawler gathers as many pages as it can from a particular set of URL’s, Where as a focused crawler is designed to only gather documents on a specific topic, thus reducing the amount of network traffic and download. The goal of the focused crawler is to selectively seek out pages that are relevant to a pre-defined set of topics.The topics are specified not using keywords, but using exemplary documents. Rather than collecting and indexing all accessible web documents to be able to answer all possible adhoc queries, a focused crawler analyzes its crawl boundary to find the links that are likely to be most relevant for the crawl, and avoids irrelevant regions of the web.

A focused crawler predicts the probability that an unvisited page will be relevant before actually downloading the page.This leads to significant savings in hardware and network resources, and helps keep the crawl more up-to-date. [5]

1. **Topical crawler**

 A topical web crawler is to collect web pages that describe some pre-specified topics. The web pages collected by the topical crawler share the same or similar words and however among them not a few pages can be irrelevant to the given topics. In particular, the performance of topical crawler degrades for a more specific topic. To achieve successful topical crawling, an additional job is required to actively filter out the pages irrelevant to the given topics. For this a machine learning architecture is used that can effectively handle not only literal term features but also numeric meta-features to improve topical web crawler; in our work we intend to more precisely crawl the web pages about ‘Jadavpur university IT department” as a specific topic. For the numeric meta-features logistic regression and random forest learning algorithms and for the literal word features, Naive Bayes and support vector learning algorithms. [6]

1. **Adaptive crawler**

Although [crawling frequency](https://www.promptcloud.com/blog/data-scraping-vs-data-crawling/) can be specified, optimal frequencies are hard to determine. The problem is that sites may not update as frequently as they are crawled. The result is suboptimal crawling, redundant data and a negative impact on the target site due to frequent, unproductive crawls.

The solution is [intelligent adaptive crawling](https://www.promptcloud.com/travel-reviews-aggregation/) where the crawler identifies pages that are updated more frequently by machine-learning. As a radical solution, crawls run more frequently on updated pages than dormant. The crawlers modify automatically to establish optimal frequencies based on site behavior and changes. They refine the list of URLs to process and extend the archive with semantic information about extracted content

[Adaptive focused crawling](https://www.promptcloud.com/web-crawling-service/) is largely beneficial to [extract data from forum-based sites](https://www.promptcloud.com/crawl-scrape-and-extract-rss-feeds-blog-forum-news-content-aggregator/) where certain threads are more active than the ones that remain closed / latent.

**1.5 Challenges of building a crawler**

* **Scale**

The web is very large and continually evolving.Crawlers that seek broad coverage and good freshness must achieve extremely high throughput, which poses many difficult engineering problems

* **Content selection tradeoffs.**

No crawler intends to scrape whole web or entire web page .Its mostly interested in a certain information (e.g like an almnni,country he is currently living ,organization he is working for) which may change or not and at the same time and bypass low-quality, irrelevant, redundant, and malicious content. The crawler must balance competing objectives such as coverage and freshness, while obeying constraints such as per-site rate limitations. A balance must also be struck between exploration of potentially useful content, and exploitation of content already known to be useful.

* **Ethics of crawling**

Crawling has to be done so that it does not stress out the target websites and cause a DoS attack

**2 Design goals of a web crawler**

* **Coverage and completeness**

Coverage measures the relative number of pages discovered by the web crawler. Ideally given enough time the web crawler has to find all pages and build the complete model of the application. This property is referred to as Completeness.

* **Freshness**

The search engine index has to be updated constantly to reflect changes in web pages created dynamically. The ability of the web crawler to retrieve latest update sis measured through freshness.

The crawler has to determine when and how it will pull for the changes in the crawled sitw

* **Politeness**

An important and old issue in designing web crawlers is called politeness Early web crawlers had no mechanism to stop them from bombing a server with many requests. As the result while crawling a website they could have launched an Denial of Service(DoS) attack unintentionally A barrage of requests in short order is considered “impolite”. So delays must be introduced between requests

This especially has to be maintained by the URL distributor among concurrent or distributed crawlers which uses so that there are no redundant requests

* **Black-lists and traps**

Traps are seemingly large set of websites with arbitrary data that are meant to waste the web crawler resources. Integration of black-lists allowed web crawlers to avoid traps

* **Scalability**

The web has been growing and evolving exponentially.So web crawlers have to keep up with those changes and still produce correct data

* **Efficiency**

**2.2 Algorithms of a crawler**

**General algorithm**

Initialize v v0.

while NOT FINISHED do

Fetch data for user v and obtain N(v), the list of friends of v

Add N(v) to list of observed users, according to used technique

Select next user v from the list of observed users, according to used technique

end awhile

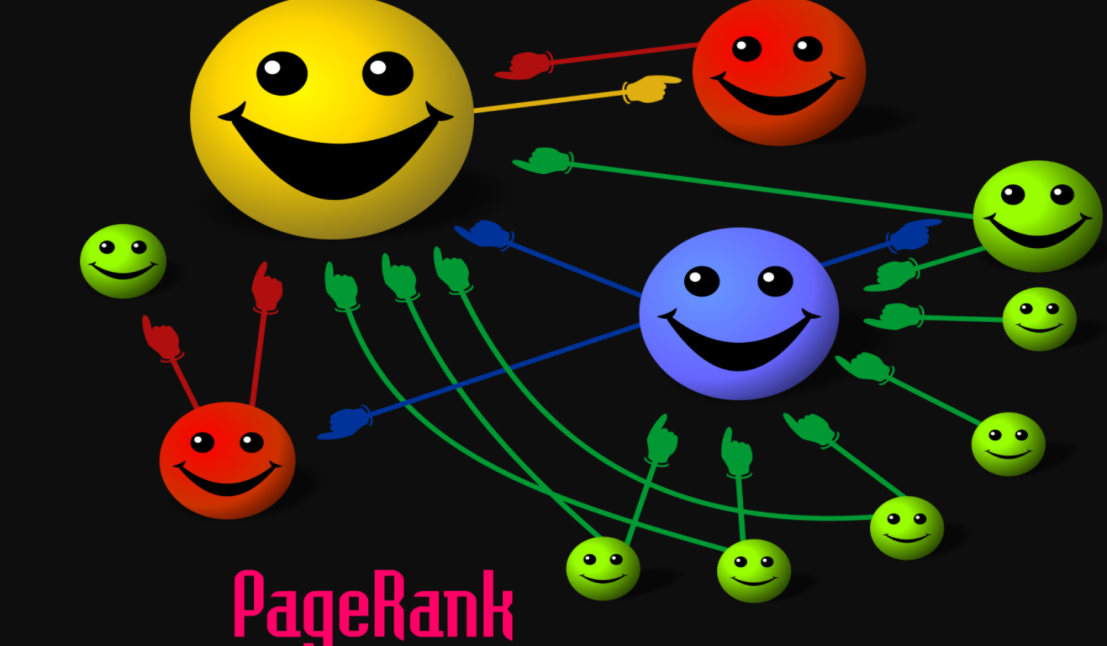
return all users v;

* **Best first search**

The Best-First algorithm focuses on the retrieval of pages which are relevant to a particular given topic. It’s an algorithm that uses a score to define which page has a best score. This algorithm uses a rule to select the best page. In most cases it uses artificial intelligence algorithms (Naïve Bayes, Cosine Similarity, Support Vector Machine, k-nearest neighbour algorithm, Gaussian mixture model, etc.) as a classifier to detect the best result. In many articles this algorithm has the best crawling results.

* **PageRank**

PageRank is a [link analysis](https://en.wikipedia.org/wiki/Network_theory#Link_analysis) algorithm and it assigns a numerical [weighting](https://en.wikipedia.org/wiki/Weighting) to each element of a [hyperlinked](https://en.wikipedia.org/wiki/Hyperlink) [set](https://en.wikipedia.org/wiki/Set_(computer_science)) of documents, such as the [World Wide Web](https://en.wikipedia.org/wiki/World_Wide_Web), with the purpose of "measuring" its relative importance within the set.Weight of a page is proportional to number of links poiting of the page - weight of linking pages



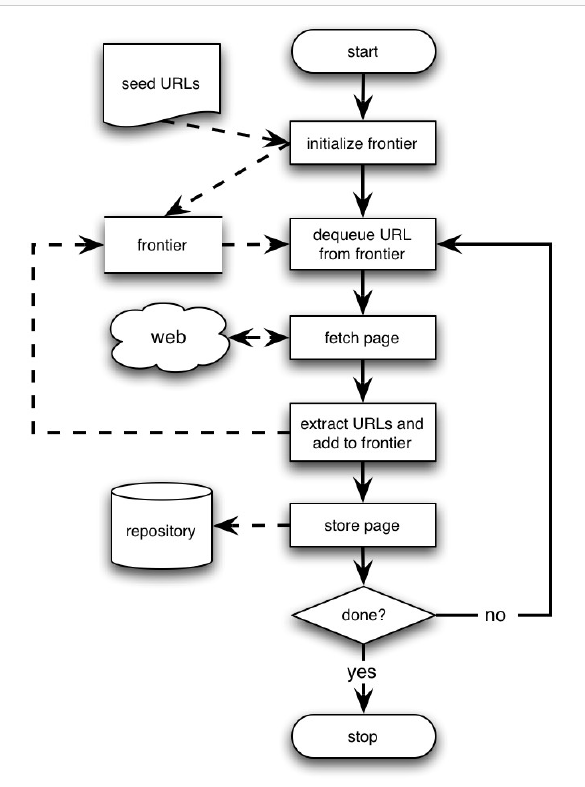
**Fig 3.**

Cartoon beuatifully illustrates basic principle of PageRank. The size of each face is proportional to the total size of the other faces which are pointing to it.

* **Fish Search**

Fish Search is a dynamic heuristic search algorithm. It works on the intuition that relevant links have relevant neighbours; hence it starts with a relevant link and goes deep under that link and stops searching under the links that are irrelevant. The key point of Fish Search algorithm lies in the maintenance of URL order.

**3.Architecture of crawlers**

****

**Fig 4:Basic crawler architecture**

**3.1 Basic crawler architecture**

Roughly, a crawler starts off by placing an initial set of URLs, S0, in a queue, where all URLs to be retrieved are kept and prioritized. From this queue, the crawler gets a URL (in some order), downloads the page, extracts any URLs in the downloaded page, and puts the new URLs in the queue. This process is repeated until the crawler decides to stop.

**3.2** **Commonly used Data structures in crawlers**

Virtually every modern web crawler splits the crawl state into folowing major data structures:

**URL-seen test (UST) and duplicate URL eliminator(DUE)** : structure for maintaining the set of URLs that have been discovered (whether downloaded or not) It must support set addition and set membership testing

These are essential to reduce redundancy and ensure freshness

**Frontier queue :** For maintaining the set of URLs that have yet to be downloaded.must support adding URLs, and selecting a URL to fetch next.

A straightforward implementation of the frontier data structure is a First-in-First-out (FIFO) queue. Such an implementation results in abreadth-first traversal of the web graph.

However, this simple approach has drawbacks: Most hyperlinks on the web are “relative” (i.e., refer to another page on the same web server). Therefore, a frontier realized as a FIFO queue contains long runs of URLs referring to pages on the same web server, resulting in the crawler issuing many consecutive HTTP requests to that server. A barrage of requests in short order is considered “impolite,” and may be construed as a denial-of-service attack on the web server. On the other hand, it would be wasteful for the web crawler to space out requests to the same server without doing other useful work in the meantime.

This problem is compounded in a multithreaded or distributed crawler that issues many HTTP requests in parallel.

Most web crawlers obey a policy of not issuing multiple overlapping requests to the same server. An easy way to realize this is to maintain a mapping from web servers to crawling threads, e.g., by hashing the host component of each URL.

In this design, each crawling thread has a separate FIFO queue, and downloads only URLs obtained from that queue

**Robots exclusion cache**

Web crawlers are supposed to adhere to the Robots Exclusion Protocol , a convention that allows a web site administrator to bar web crawlers from crawling their site, or some pages within the site.

This is done by providing a file at URL /robots.txt containing rules that specify

which pages the crawler is allowed to download. To avoid repeatedly requesting /robots.txt, crawlers typically cache the results of previous requests of that file.

When visTing other websites this need to e refreshed

**DNS cache**

DNS requests can take quite a long time due to the requestforwardingnature of the protocol. Therefore, crawlers often maintain their own DNS caches. As with the robots exclusion rule cache, entries are expired according to both a standard eviction policy like LRU

**3.4 Distributed crawler**

Web crawlers can be distributed over multiple machines to increase their throughput.This is done by partitioning the URL space, such that each crawler machine or *node* is responsible for a subset of the URLs on the web.

The URL space is best partitioned across web site

boundaries where a web site may refer to all URLs with the same symbolic host name, same domain, or same IP address). Partitioning the URL space across site boundaries makes it easy to obey politeness policies, since each crawling process can schedule downloads without having to communicate with other crawler nodes.

Moreover, all the major data structures can easily be partitioned across site boundaries,i.e., the frontier, the DUE, and the DNS and robots exclusion caches of each node contain URL, robots exclusion rules, and name-to-address

mappings associated with the sites assigned to that node, and nothing else.

Cho[1] and Garcia-Molina studied two types of policies to distribut crawling tasks beween crawlers

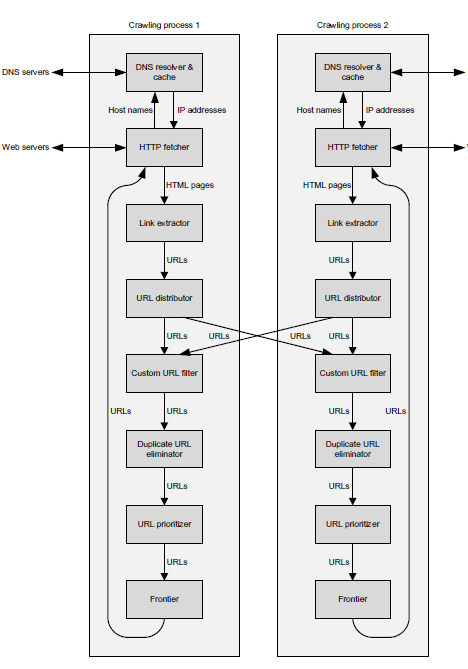
**Dynamic assignment :**

With this type of policy, a central server assigns new URLs to different crawlers dynamically. This allows the central server to, for instance, dynamically balance the load of each crawler.

With dynamic assignment, typically the systems can also add or remove downloader processes. The central server may become the bottleneck, so most of the workload must be transferred to the distributed crawling processes for large crawls.

**3.4.1 Overview of Standard distributed crawler architecture**

Here is a standard overview of a distributed architecture covering the design goals which aim to cover the design goals discussed in section i.e politeness,efficiency,redundancy eliminimaton

****

**Fig5:Architecture of distributed crawler**

The crawler consists of multiple processes running on different machines connected by a high-speed network. Each crawling process consists of multiple worker threads, and each worker thread performs repeated work cycles

1. At the beginning of each work cycle, a worker obtains a URL from the Frontier data structure, which dispenses URLs according to their priority and to politeness policies. The worker thread then invokes the HTTP fetcher.
2. The fetcher first calls a DNS sub-module to resolve the host component of the URL into the IP address of the corresponding web server (using cached results of prior resolutions if possible), and then connects to the web server, checks for any robots exclusion rules (which typically are cached as well), and attempts to download the web page.
3. If the download succeeds, the web page may or may not be stored in a repository of harvested web pages (not shown). In either case, the page is passed to the Link extractor, which parses the page’s HTML content and extracts hyperlinks contained therein.
4. The corresponding URLs are then passed to a URL distributor, which assigns each URL to a crawling process. This assignment is typically made by hashing the URLs host component, its domain, Since most hyperlinks refer to pages on the same web site, assignment to the local crawling process is
5. Next, the URL passes through the Custom URL filter (e.g., to exclude URLs belonging to “black-listed” sites, or URLs with particular file extensions that are not of interest) and into the Duplicate URL eliminator, which maintains the set of all URLs discovered so far and passes on only never-before-seen URLs.
6. Finally, the URL prioritizer selects a position for the URL in the Frontier, based on factors such as estimated page importance or rate of change.

**3.4.2 Page revisit and freshness policies**

Pages need to be refreshed and revisited to updated to represent current

data

* **Freshness**

Most of the web pages the crawler downloads on any given day were by it multiple times in the past. The crawler revisits them in order to keep the search engine’s index — and its search results — fresh.

Indeed, serving stale content such as YouTube pages with taken-down videos or failing to serve pages that the user knows to be relevant to her query due to new content is a cause of user dissatisfaction

Let S = {e1,....,eN} be the local database with N elements. Ideally,all N elements will be maintained up-to-date, but in practice, only M(< N) elements will be up-to-date at a specific time. (By up-to-date we mean that their values equal those of their real-world counterparts.) We define the fresh-ness of S at time t as F(S; t) = M/N.

Clearly, the freshness is the fraction of the local database that is up-to-date.

F(S; t) will be one if all local elements are up-to-date, and F(S; t) will be zero if all local elements are out-of-date.

The freshness of a local element ei at time t is

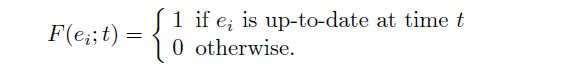
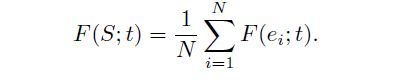


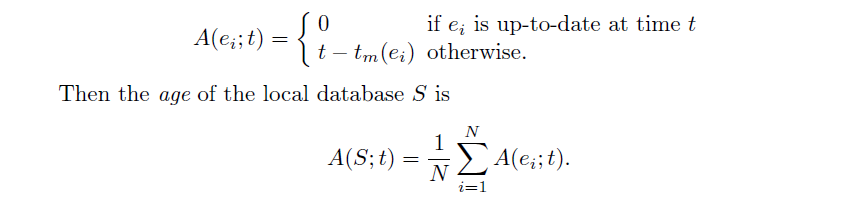
Fig 6

Then, the freshness of the local database S at time t is

****

**Fig 7**

* **Age :** To capture \how old" a database is, we de\_ne the metric age as follows:

****

**Fig 8**

The age of S tells us the average \age" of the local database. For instance, if all real-world elements changed one day ago and we have not synchronized them since, A(S; t) is one day.

Here is nice representation of change frequency and age and relationship between the

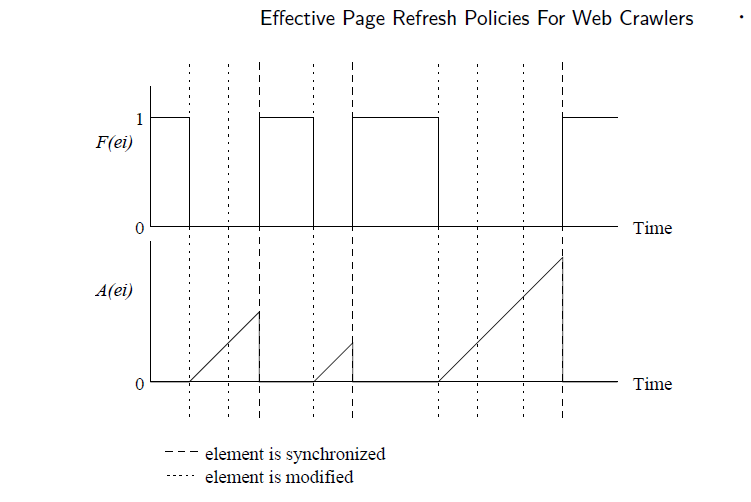


Fig 9

We assume that the real-world element changes at the dotted lines and the local element is synchronized at the dashed lines. The freshness drops to zero when the real-world element changes, and the age increases linearly from that point on.

When the local element is synchronized to the real-world element,its freshness recovers to one, and its age drops to zero it is possible that an element changes multiple times between synchronization.

**3.4.3 Synchronization policies**

1. **Synchronization frequency:**

Fresness of local database depends how frequently we synchronize the local database. In our analysis, we assume that we synchronize N elements per I time-units. By varying the value of I, we can adjust how often we synchronize the database.

1. **Resource allocation:**

Even after we decide how many elements we synchronize per unit interval, we still need to decide how frequently we synchronize eachindividual element. as all parts of a website need not be refreshed throught the day Depending on elements we may synchrnnize them using **uniform allocation policy** (all elements at same rate regardless of how often they change or **non uniform allocation policy** (where frequency is proportional to change frequency)

1. **Synchronization order:**

We maintain a local database of 10,000 Web pages from site A. In order to maintain the local copy up-to-date, we continuously update our local database by revisiting the pages in the site.

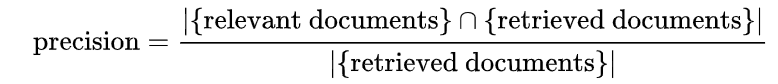
**4. Metrics of a crawler**

**• Quality metrics**

**Relevance**: The degree to which returned answers meet user’s information need. Commonly used relevance metrics are precision and recall

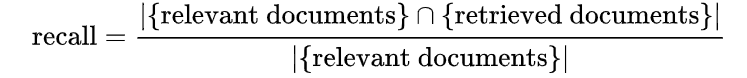
**Precision**

Precision is the fraction of the documents retrieved that are [relevant](https://en.wikipedia.org/wiki/Relevance_(information_retrieval)) to the user's information need.

****

**Recall**

Recall is the fraction of the documents that are relevant to the query that are successfully retrieved.



**Performance metrics**

The following metrics are calculated in measuring performance

**Hit rate**: fraction of total requests that are answered by the cache

**Cost:** total processing cost incurred to the backend system

**Harvest rate:**

Harvest rate is a common measure on how well a focused crawler performs. It is expressed as HR= r/p,where HR is the harvest rate, r is the number of relevant pages found and p is the number of pages downloaded.

Harvest Rate of focused crawlingis better than unfocussed crawling

**5.****Crawling social networks**

**Online Social Networks**

Here, we define an OSN crawler to be a computer program that explores (or visits or samples) users in the social graph using an orderly method.

**5.1 Data Collection**

The value of an online social network is largely defined by the richness and magnitude of the user data it stores.

The quality and quantity of data you gather in this step will determine how efficient the model is .

However after the EU GDPR resolution any Personally Identifiable data (PI data) cannot be collected and stored without consent which entails hefty penalties

**5.2 What to check before scraping**

**Terms and robots**

Scraping currently has barely any limitations; there are no laws defining what can be scraped and what cannot.

However, there are guidelines that define what one should respect.

Before you start any scraping task *Terms & Conditions* and *Privacy Policy* of the website you want to gather data from should be carefully checked

When reading the terms and conditions of a website, one should search for following keywords to find restrictions:

* scraper/scraping
* crawler/crawling
* bot
* spider
* program

Most of the time these keywords can be found, and this makes your search easier.

**Robots.txt**

Most websites provide a file called robots.txt, which is used to tell web crawlers what they can scrape and what they should not touch.

Here is example of such a file:

User-agent: \*

Disallow: /covers/

Disallow: /api/

Disallow: /\*checkval

Disallow: /\*wicket:interface

Disallow: ?print\_view=true

Disallow: /\*/search

Disallow: /\*/product-search

**Allow: /\*/product-search/discipline**

Disallow: /\*/product-search/discipline?\*facet-subj=

Disallow: /\*/product-search/discipline?\*facet-pdate=

Disallow: /\*/product-search/discipline?\*facet-type=category

This is how robots.txt file of linkedin looks like

---BEGIN ROBOTS.TXT ---

User-Agent: \*

Disallow:

Sitemap: http://cowork.com/sitemap.html

--- END ROBOTS.TXT ----

**which**  means it forbids all web crawling bots/user\_

**6 . Demo application**

The demo applications aims at crawling OSN linkedin to gather information about the alumni of Jadavpur university information techno logy department

This makes it focused crawler

**6.1 Overview of linkedin**

LinkedIn, a social networking site focused on professional and business relationships.Member trying to convey the specific value and expertise that they can bring to the professional marketplace.So is expected that information in linkedin is mostly genuine.

**6.2 Motivation of scraping Linkedin :**

* Building business networking
* Prospective employers and employees
* Keeping touch with collegues
* Meeting place for alumni of companies,universities
* approaching a customer base who mean serious business
* Keeping track of completion,customer for enterprises
* Information about research activities of various topics

**Questions that we are going to ask LinkedIn**

* Where are the alumni of Jadavpur University IT department located?
* They are from which batch?
* What degrees do they posses?
* Where are they working now?

Answers to all these questions will definitely help the university to reach out to its dear students and also help networking for industry collobarations,placements of students and inform about what research and cultural activities university is doing currently

**6.3 Challenges faced**

**Authetication :**

Linked is not an open forum type website one has to be a member only (which is free).All the data was available behind closed doors of the login window.Any scraping request could only be sent after one logs in and a session started So for this selenium web driver was used to automate one time login before running the queries in the search box on every run

**Non uniform profile data:**

Lot of non uniformit due to missing data.(e.g Some one has not mentioned year of passing

while someone has not mentioned department

**Different data privacy setting :**

While many results come up in the search query that matches the data is not worth collecting as their name is showing as”LinkedIn member” instead of names as they have chosen to expose their profiles to their chosen acquaintances only. So lot of information could not be collected for that

**Linkedin block profiles suspecting scraping :**

Acccoring to terms and condition sraping is forbidden in linked in so my original profile got blocked while running a few tests .Its has a way to understand request.First time when it detected it asked for a captha and the next time it blocked my profile for a few days

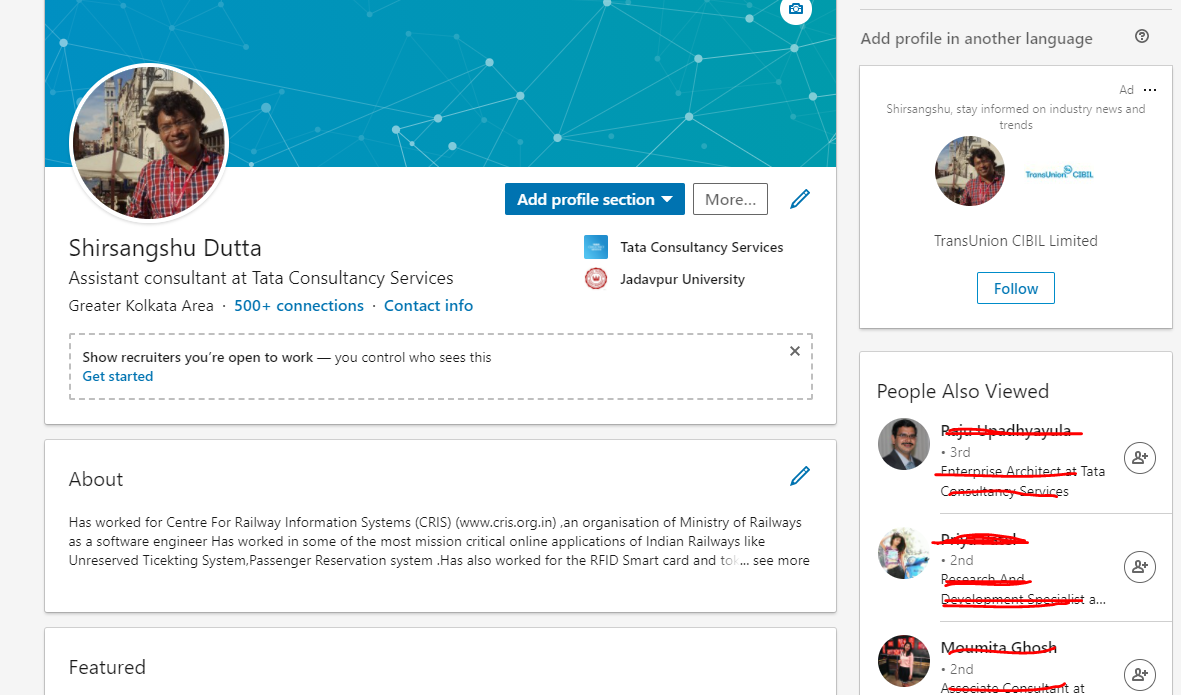
Had to mail to support to assure that it was educational and would nothappen again to unblock it

From next time onwards used another profile and had to resort to politeness by inserting delays between requests which on the other hand made execution slow

**6.4 Crawling Strategies attempted**

Approach 1 :Exploring matching profiles

Start with a seed profile matching all criterrai(i.e JU IT dept alumni) and check the “People also viewed” section where it shows similar profiles in a list and then scrape that section for profiles matching target criteria

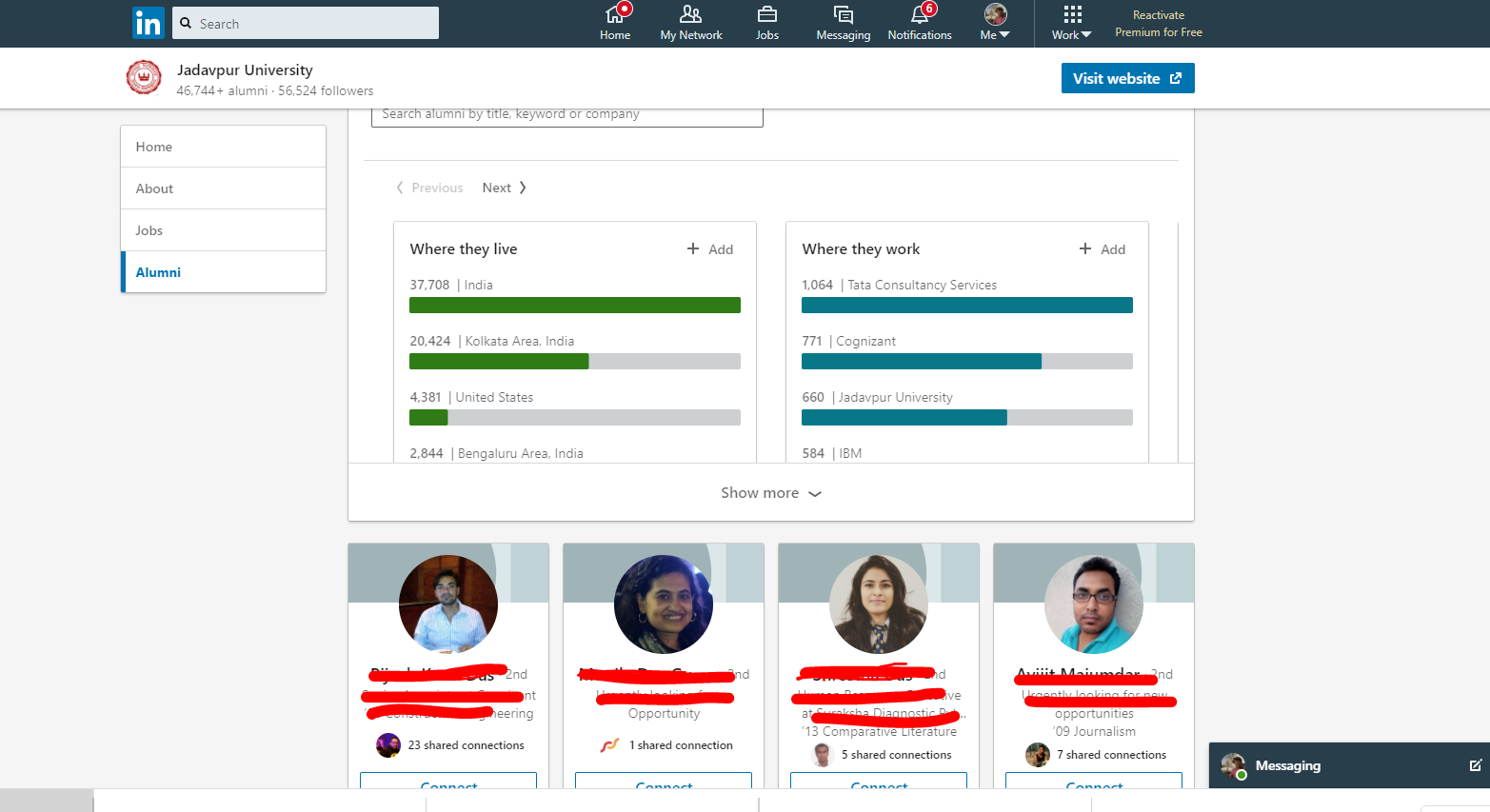


While that was doable that section contains lot of profiled not matching with criteria but has similarity with seed profile on other criterias like same company,same job profile

**Approach 2: Looking up alumni public groups**

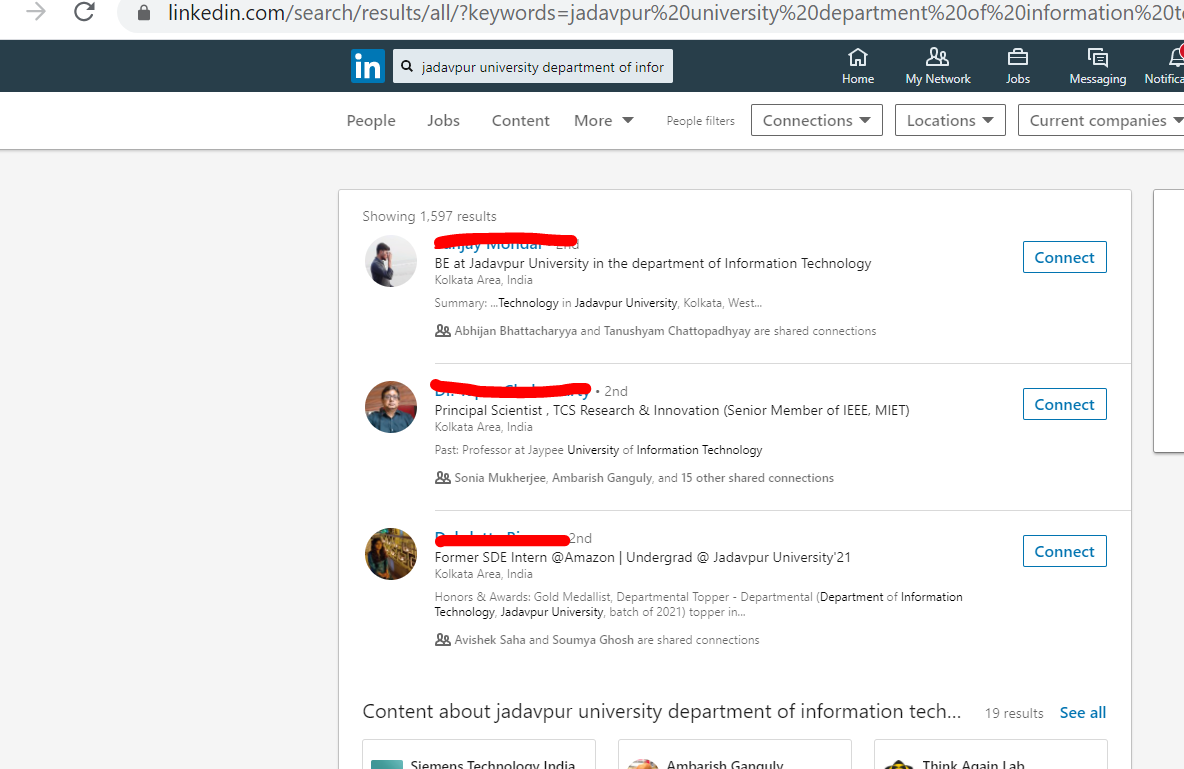
Linkedin has some already created public groups of JU alumni.This approach entail directly searching for such public groups and then scraping the member profile on by one but mostly these groups are having student from all department s so checking all profiles and matching target criteria makes this approach slow.

Also another disadvantage is that many alumni may not be a member of this group



**Approach 3 : Keyword search**

This is done by searching keywords like ”Jadavpur University Department of information Technology” or “Jadavpur University IT department” in the search box and then scraping the results one by one



**6.5 How the app works**

**Technologies used:**

Python,Selenium,BeautifulSoup

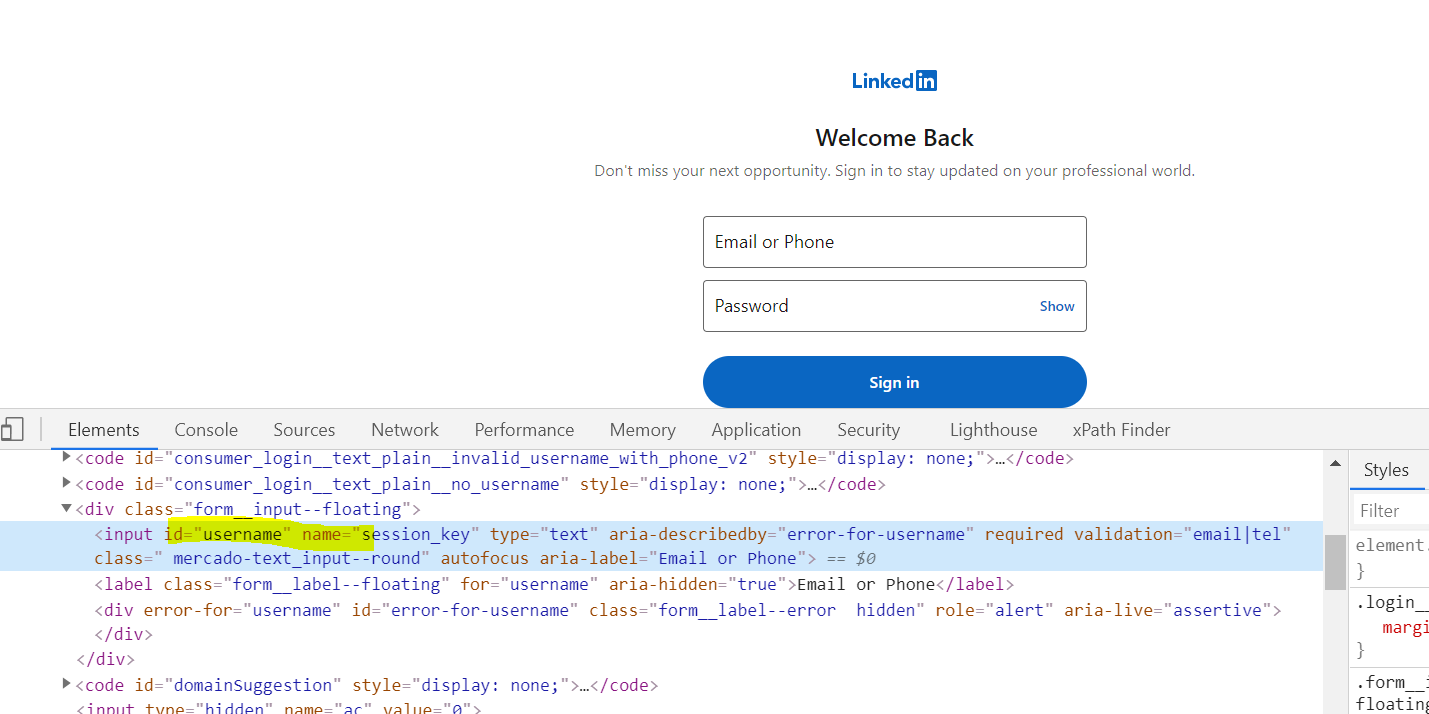
Python was used as the programming language as it had many useful feature also along with it BeautifulSoup, a Python library for pulling data out of HTML and XML files.

Data scraped was in python collection objects and later dumped on to excel files or stored in a document database (Mongodb)

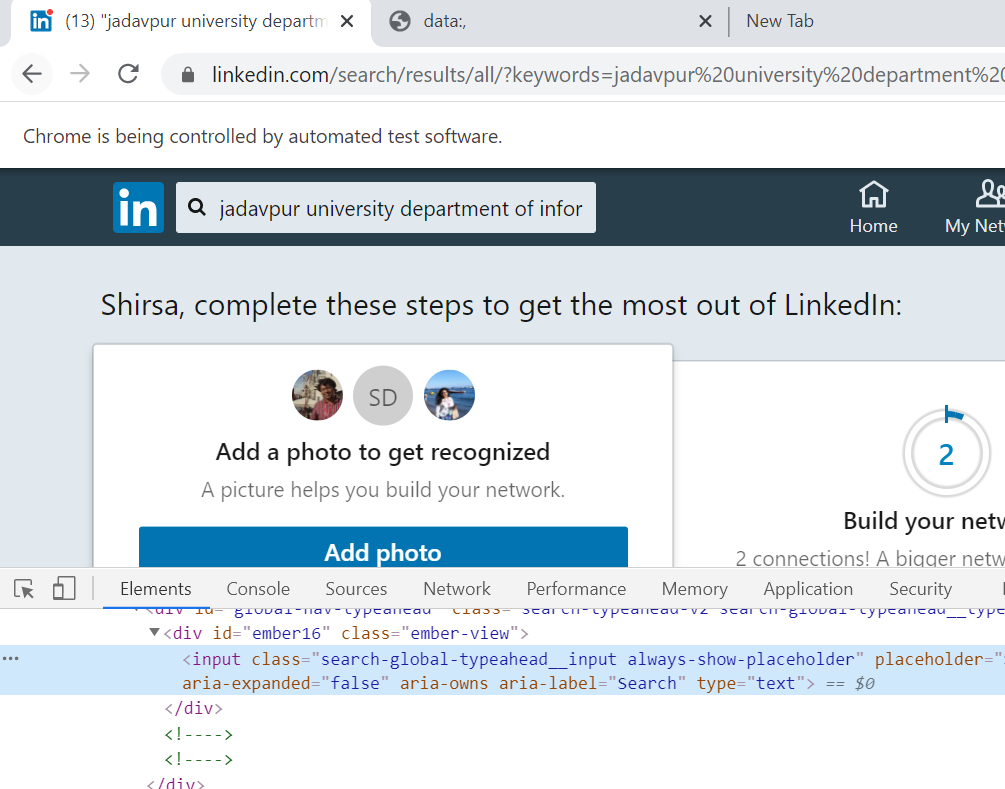
The whole process was automated using selenium webdriver which controls the browser emulating normal user activity

**Step1 :** Automated Login using Selenium webdriver

The username and password element is identified and values are passed by selenium web driver in an automated fashion



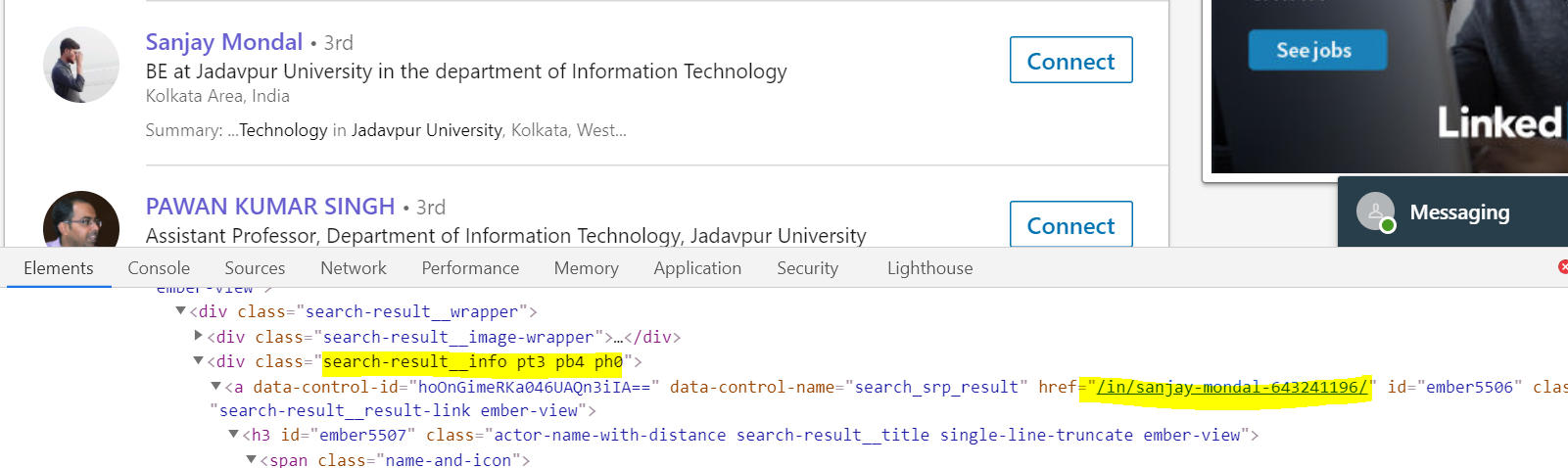
**Step 2:**



get the id of thse search-box and bind search text “Jadavpur university department of technology” (one of the many search string that were tried)with the search box

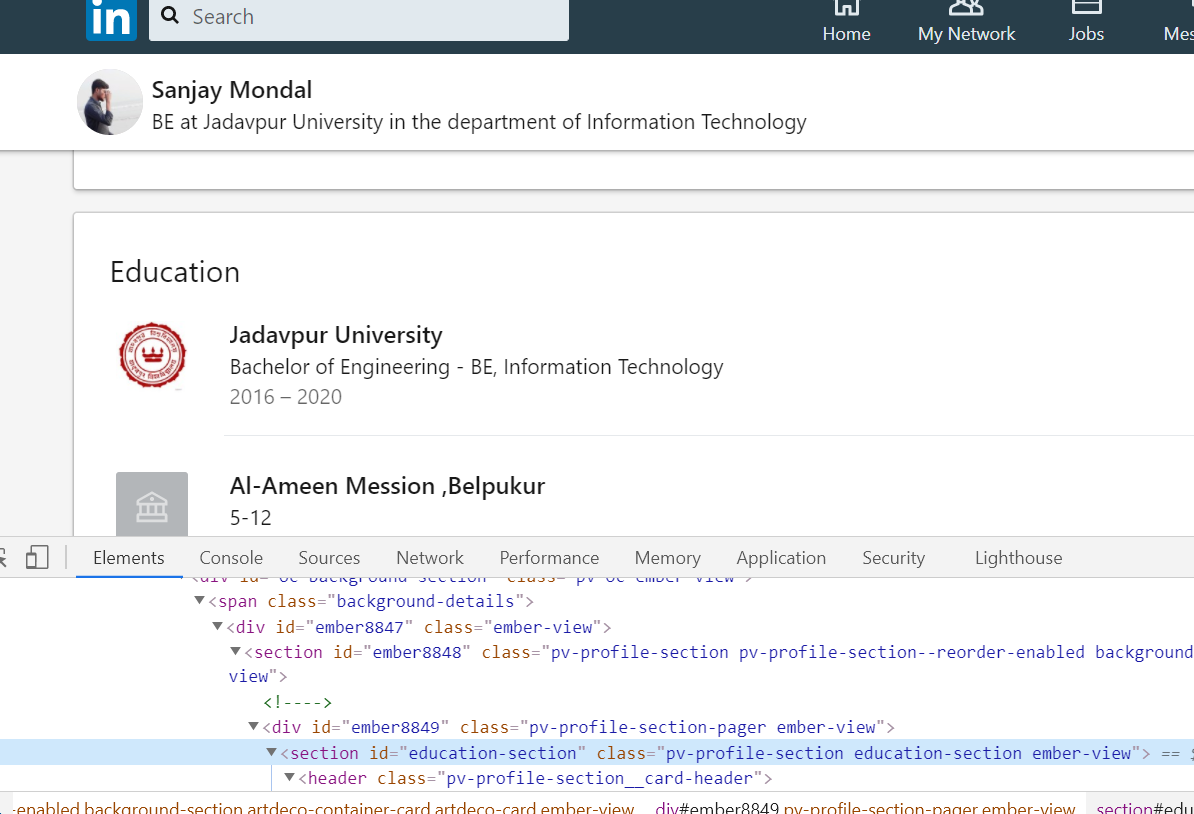
**Step3 :**

For example it loops through the search result page looking for “div” elementof class “class="search-result\_\_info pt3 pb4 ph0" and then navigating the DOM tree it finds the href element which contains link to profile page of target profile

The class names of the div element have remain unchanged over the period this code 

**Step 4**

Having got the link a request is generated to go profile page and inspect the html elements “Education-Section” and “Experience-Section” the details such as stream,year of graduation,Branch ,current employers are collected



**Step 5:**

The data collected is then stored in excel file and store in aMongodb database

**Drawbacks**

* It is single threaded and the harvest rate is very slow .
* Profile pages being non uniform in structure many elements being not present (like for some profiles “Year of graduation” is missing while in some “Branch” to handle those exceptions complexity of code increased.
* There is always a chance of being blocked even after delaying request
* It is fetching all data every time instead of picking incremental data

7 **Future scope of work**

The web scraper can be made to be a muti threaded distributed crawler where from a main distributor thread which dynamically spawn child threads having their own frontier queues and url list to visit .Also batches to refresh data could be implemented to refresh only the data of interest.

Also ML techniques like classification can be used to make the crawler more intelligent with the data already scraped

And also similar attempt can be tried with other OSN like Facebook

**8.Conclusion**

In this paper we discussed basic of crawler,how the technology evolved over time and taxonomy of crawlers.We studied what are the guiding principles of designing robust and scalable crawler .We studied what architecture of distributed architecture should be like in keeping with the design principles,what is freshness and age and strategies how local data will be synchronized with the latest updates of webisites.

Then we discussed about OSN,how data is collected from theem and what are challenges and ethics of an OSN crawler,

Finally we discussed the demo app that was built to scrape Linkedin data and the different approaches and their pros and cons

**References**

1. A Brief History of Web Crawlers :

Seyed M. Mirtaheri, Mustafa Emre Dinc¸turk, Salman Hooshmand, ¨ Gregor V. Bochmann, Guy-Vincent Jourdan School of Electrical Engineering and Computer Science University of Ottawa, Ottawa, Ontario,

2. Cho, & Junghoo, & Garcia-Molina, & Hector, Hector. (2003). Effective Page Refresh Policies for Web Crawlers. ACM Transactions on Database Systems. 28. 10.1145/958942.958945.

3. Erlandsson, F., Nia, R., Boldt, M., Johnson, H., Wu, S F. (2015)

Crawling Online Social Networks.

In: Network Intelligence Conference (ENIC), 2015 Second European (pp. 9-16).

<http://dx.doi.org/10.1109/ENIC.2015.10>

 4 . [Improving the Performance of Focused Web Crawlers](http://www.intelligence.tuc.gr/~petrakis/publications/BaPeMi09.pdf)[[1]](https://qoqlinks.com/), Sotiris Batsakis, Euripides G. M. Petrakis, Evangelos Milios, 2012-04-09

5. Focused crawling: a new approach to topic-specific Web resource discovery

Soumen Chakrabarti a , Martin van den Berg b , Byron Dom c a Computer Science and Engineering, Indian Institute of Technology, Bombay, 400076, India b FX Palo Alto Laboratory, 3400 Hillview Ave, Bldg 4, Palo Alto, CA 94304, USA c IBM Almaden Research Center, 650 Harry Rd, San Jose, CA 95120, USA

6. Tae Jun Kim and Han- Joon Kim, 2017. Machine Learning-Based Topical Web Crawler: An Ensemble Approach Incorporating Meta-Features. Journal of Engineering and Applied Sciences, 12: 4651-4656.

DOI: [10.36478/jeasci.2017.4651.4656](http://dx.doi.org/10.36478/jeasci.2017.4651.4656)