INDRAPRASTHA INSTITUTE *of*
INFORMATION TECHNOLOGY
DELHI

Department
of
Computer Science Engineering

CSE508|Information Retrival

Rajiv Ratn Shah

Final Review

Group: 27
Members: Anand (2020280)
Apoorva Agrawal (2020426)
Vaidik Kumar (2020346)

Web Crawling- IIITD Resources

Anand (2020280)
(Student) B. Tech
IIITD

Vaidik Kumar (2020346)
(Student) B. Tech
IIITD

Apoorva Agrawal (2020426)
(Student) B. Tech
IIITD

Rajiv Ratn Shah
Professor
IIITD

***Abstract*— Searching for and locating information within documents, online databases, and the internet are all part of information retrieval. A web crawler is a computer program or software that methodically and automatically browses the Internet and downloads web content. Web crawlers often utilize one of three types of crawling techniques: distributed, focused, and general-purpose crawling. The main aim of this research was to create a simple, fast and precise method to find desired resources that IIITD provide online. Now that we have implemented that, found the solution to the main problem that we were having and also made a search engine for searching easly. Search engines employ web crawlers, which are computer programs, to automatically gather the data they require from the internet by the user's specified parameters. Web crawlers must move extremely quickly because of the volume of data and resources that IIITD makes available online. A number of earlier studies have explored the process of extracting information from a web document that must be taken into account from both perspectives, including the structure of the web page and the amount of time required. Therefore, in this research we will be creating a custom web crawler which will only crawl though IIITD website and provide the result based on the user's query and using the custom web crawler algorithm used.**

I. INTRODUCTION

The Internet has emerged as the most significant information source due to the network's explosive expansion in information. Research on extracting relevant knowledge from vast amounts of data is quite popular. Nonetheless, the primary objective of those studies is to gather pertinent data from the Internet through crawling web pages. Thus, researchers suggested web crawlers to crawl web pages efficiently. Web crawler applications have existed for as long as the Internet. These are brief computer programs commonly referred to as "bots," "ants," or "worms," to download web pages, extract hyperlinks from them, compile a list of their URLs, and add them to a local database (Chakrabarti, 2003). Search engines are their most popular uses, and every internet user is familiar with them in this form. In 1998, Google's owners released a widely cited academic paper on crawler programming, one of the earliest works demonstrating how successful search crawling and indexing can be on the web. There are many types of web crawlers but we will be focusing on the keyword focused web crawler.

II. PROBLEM STATEMENT

During the course of college, one has to visit college website for different resources (link to library, ERP, know course policy). But no link provides for upper mentioned things and many more.

III. MOTIVATION

Based on real life scenario- while searching for various links like library, ERP or fee payment not able to find it. The webpage of IIITD resources only provide link to course description and regulation. No mention of link to onl8ine library ort any other resource. Spending so much time on IIITD website not still not able to find link to student resources.

IV. OBJECTIVE

1) To provide the most accurate and precise result using the custom algorithm.
2) To identify the errors and optimize the custom algorithm made.
3) Identify the feasibility of the web crawler in large scale magnitude.
4) Create a simple search engine or portal. 5) Create a simple UI.
6) Sort and select what all will be the stop words and also what all to include in the web crawler.

V. NOVELTY

1. One may face problems due to low bandwidth or storage.

2. Too much information being crawled or extracted.

3. Some URLs which needs VPN connection will not able accessible.

4. No changes can be made to the web crawler code without the previous knowledge about it.

5. Needs perpetual maintenance.

VI. LITERATURE REVIEW

Possibly the largest level study of Web page change was performed by Fetterly et al. . They crawled 151 million pages once a week for 11 weeks, and compared the modification across pages. Like Ntoulas et. al. , they found a relatively small amount of change, with 65% of all page pairs remaining exactly the same. The study furthermore found that past change was a good judge of future change, this page length was correlated with change, and that the top level domain of a page was correlated with change. Describing the amount of change on the Web has been of significant interest to researchers. While we only had a small database and limited resources. But still depending upon the resources invested we have been able to make progress and get that desired result accordingly that that we have studied in the different research papers and we are still make optimizations.

VII. WORK PLAN OR EVALUATION

- **Zeroth Review**: We went through several research papers and understood the exact process of approaching our problem step by step. A draft along with a presentation was also made for review zero.

- **First Review**: For the first review we will try and test our custom-made algorithms for different datasets.

- **Final Review**: For the final review we will come to the conclusion that our custom-made web crawler was successful in delivering the desired results.
-

VIII. METHODOLOGY

The goal of this research is to develop a web crawler that will help students in finding IIITD- Resources easily. To achieve the goal, we will make our own custom algorithm-based web crawler. For this we will use external libraries like: BeautifulSoup4 for HTML parsing and NLTK Porter Stemmer for stemming.

Crawler Object

- **seed_url**: The starting URL used to initialize the crawler
- **url_frontier**: A queue of pages to be crawled
- **robots_txt**: A dictionary containing directories that are either "Allowed" or "Disallowed"
- **visited_urls**: A dictionary containing visited links, their page titles, and a unique Document ID.
- Document IDs are assigned by hashing the contents of a page
- **duplicate_urls**: A dictionary containing DocumentIDs and arrays of the pages that produce that ID.
- **outgoing_urls**, broken_urls, and graphic_urls: Lists of various types of URLs picked up by the crawler
- **words**: A dictionary containing DocumentIDs and the valid words found in a page
- **frequency_matrix**: A 2D array of terms and their frequencies

Crawler Algorithm

First, we will add the seed URL to the query URL frontier. Now while the queue isn't empty, we will pop the next URL from the queue. Now if the URL meets the requirement of the file, we will "Visit" the page using python's URL library. Now we will parse the contents of the page using Beautiful Soup. Then we will use regex to store the valid "words" of a page. Then store the necessary metadata about the page. Then add valid, unvisited links found in <a> tags to the queue. This crawler is considerate in that it examines the file before accessing a page. Also, it is relatively durable because it is resistant to duplicate and broken pages, both detected after the crawler is done. With the help of the NLTK Porter stemmer, each distinct word is separated into its term and document frequencies.

Search Engine Implementation

The search engine class inherits the original WebCrawler class and has the additional functionality.

1. Import/export and index from disk.
2. Contains documents clusters.
3. Computes the cosine similarity between documents.
4. Processes user queries
5. Returns results of user queries.
6. Thesaurus expansion of user queries.

The process of selecting 5 random leaders and assigning followers to the leaders with least Euclidean distance results in the clustering of the technique. This clustering technique works fine, but it has a tendency to concentrate documents around a small number of leaders.

The SearchEngine class also deals with user inquiries and stopwords are removed. Each non-space character in a word must come after an alphabetic character. The final

character of a word can be special or alphabetic and can also be numeric.

The phrase frequency vectors from user inquiries are then compared to each document. This weighting system performs a decent job of balancing the frequency of phrases within and between documents, taking care to not give terms that appear frequently an excessive amount of priority while giving rarer terms a greater significance.

The user is only provided with documents that have a cosine similarity score higher than 0. After expanding the search using thesaurus, the search engine resumes its search if it discovers less than three pertinent documents.

Finally,.25 is added to the score of documents whose titles contain search phrases. In general, I believe the search engine does an excellent job of providing the user with pertinent results.

```
------------------------------------------------------------------------
Seed URL: " http://techtree.iiitd.edu.in/viewDescription/filename?=CSE102 "
Page limit: 20
Stop words: stopwords.txt
------------------------------------------------------------------------
Beginning crawling...

robots.txt: Disallowed['http://techtree.iiitd.edu.in/viewDescription/filename?=CSE102'] Allowed[]

1. Visiting: /~CSE102/


Done crawling.
------------------------------------------------------------------------
Visited URLs: 1

Outgoing URLs:
  +  http://techtree.iiitd.edu.in/viewDescription/filename?=CSE102
  +

Broken URLs:


Graphic URLs:

Duplicate URLs:

------------------------------------------------------------------------

Building Term Frequency matrix...

Most Common Stemmed Terms:

Term                      Term Frequency        Document Frequency
----                      --------------        ------------------
1. syllabus                            2                         1
------------------------------------------------------------------------

Complete frequency matrix has been exported to tf_matrix.csv


Goodbye!
```

Baseline result above. And final review below

IX. RELATED WORK

1. K. Zhang et al. developed heuristic rules to discover URLs. Koppula et al. tries to mine rules of different URLs with similar text.

2. Another technique like Meta search that collect topic related web pages and combines returned result to be the seed URLs.
3. According to M. Yuvarani et al., there are two processes in operating a centred crawler. The first step is to look for the starting URLs and identify the user requirements. The crawling technique is the second phase. The crawler uses the most relevant URLs in the queue as a route selection technique.

4. While our work will be based on the customization of the web crawler so that it can only be used for the IIITD-resource purpose and its method will be different (mentioned in the methodology).

```
$ python SearchEngine.py
------------------------------------------------------------------------
|    II Search Engine                                                  |
|                                                                      |
|    [0] Exit                                                          |
|    [1] Build Index                                                   |
|    [2] Search Documents                                              |
------------------------------------------------------------------------
Please select an option: 1
------------------------------------------------------------------------
Would you like to import the index from disk? (y/n) n

Seed URL: http://techtree.iiitd.edu.in/viewDescription/filename?=CSE121
Page limit: 60
Stop words: Input/stopwords.txt
Thesaurus: Input/thesaurus.csv

Beginning crawling...


1. Visiting: /~CSE121/ (course discription- IIITD)
2. Visiting: /~CSE121/http://techtree.iiitd.edu.in/
3. Visiting: /~CSE121/https://iiitd.ac.in/
4. Visiting: /~CSE121/http://techtree.iiitd.edu.in/techtree
5. Visiting: /~CSE121/https://iiitd.ac.in/academics/resources
6. Visiting: /~CSE121/http://techtree.iiitd.edu.in/contact
7. Visiting: /~CSE121/http://techtree.iiitd.edu.in/credits

Index built.
------------------------------------------------------------------------
Would you like to see info about the pages crawled? (y/n) y

Pages crawled: 1
Pages indexed: 43
Visited URLs: 7

Outgoing URLs:
  +  http://techtree.iiitd.edu.in/
  +  https://iiitd.ac.in/
  +  http://techtree.iiitd.edu.in/techtree
```

X. FINAL REVIEW RESULT

The implementation of the custom web crawler using the mentioned algorithm is successful. The intended results have been obtained.

Now after the making changes implementation of the search engine the searching of queries has been made easier.

```
------------------------------------------------------------------------
Building Term Frequency matrix... Done.


Complete frequency matrix has been exported to Output/tf_matrix.csv
------------------------------------------------------------------------

Would you like to see the most frequent terms? (y/n) y
------------------------------------------------------------------------
Most Common Stemmed Terms:

Term                      Term Frequency        Document Frequency
----                      --------------        ------------------
1. and                                10                         1
2. Students                           12                         1

------------------------------------------------------------------------

Beginning clustering...

Documents clustered. Would you like to see their clustering? (y/n) n
------------------------------------------------------------------------



Would you like to export this index to disk? (y/n) y
Exported to Output/exported_index.obj.

------------------------------------------------------------------------
|    IIITD Search Engine                                               |
|                                                                      |
|    [0] Exit                                                          |
|    [1] Build Index                                                   |
|    [2] Search Documents                                              |
------------------------------------------------------------------------
Please select an option: 2
------------------------------------------------------------------------
```

```
------------------------------------------------------------------

Please enter a query or "stop": Students
------------------------------------------------------------------------------
1.    http://techtree.iiitd.edu.in/viewDescription/filename?-CSE121

      "Students are able to read, interpret and write some basic mathematical notations."


2.    http://techtree.iiitd.edu.in/viewDescription/filename?-CSE121

      "Students are able to recognize and to construct examples of mathematical objects introduced during the course such as the sets and functions."


3.    http://techtree.iiitd.edu.in/viewDescription/filename?-CSE121

      "Students are able to develop several matematical models."

------------------------------------------------------------------

Please enter a query or "stop": stop

Goodbye!
```

```
from WebCrawler import WebCrawler
import pickle
import sys
import argparse
import numpy as np
import random
from sklearn.metrics.pairwise import euclidean_distances
from nltk.stem import PorterStemmer
import math
import csv
from textwrap import wrap


class SearchEngine(WebCrawler):
    def __init__(self, seed_url):
        super().__init__(seed_url)
        self.thesaurus = None  # {word: alternative}
        self.thesaurus_file = None
        self.clusters = None  # {leader: [ (followerN, distanceN) ]}
        self.N = None  # number of docs indexed
        self.df = None  # doc frequency for each term

    # parses a thesaurus file and sets attribute
    def set_thesaurus(self, thesaurus_file):
        thesaurus = {}

        try:
            with open(thesaurus_file) as csvfile:
                reader = csv.reader(csvfile, delimiter=',')
                for row in reader:
                    word = row[0]
                    alternatives = row[1:]
                    thesaurus[word] = alternatives

            self.thesaurus = thesaurus
            self.thesaurus_file = thesaurus_file
```

```
if __name__ == "__main__":
    search_engine = SearchEngine("http://techtree.iiitd.edu.in/viewDescription/filename?-CSE121")

    # handle command line arguments
    parser = argparse.ArgumentParser(description="Search Engine IIITD")
    parser.add_argument("-p", "--pagelimit", help="Maximum number of pages to crawl. (Required)", required=False, default="60")
    parser.add_argument("-s", "--stopwords",
                        help="Stop words file: a newline separated list of stop words. (Default is Input/stopwords.txt)", required=False, default="Input/stopwords.txt")
    parser.add_argument("-t", "--thesaurus",
                        help="Thesaurus file: a comma separated list of words and their synonyms. (Default is Input/thesaurus.csv)", required=False, default="Input/thesaurus.csv")

    argument = parser.parse_args()

    # set attributes based off arguments
    if int(argument.pagelimit) > 1:
        search_engine.set_page_limit(argument.pagelimit)

        if argument.stopwords:
            search_engine.set_stop_words(argument.stopwords)
        if argument.thesaurus:
            search_engine.set_thesaurus(argument.thesaurus)

        # show main menu to user
        search_engine.display_menu()
    else:
        print("Sorry. You must crawl a minimum of 2 pages. Otherwise, why would you need a search engine?")
```

```
from setuptools import setup

setup(
    name='Web-Crawler',
    version = '1.0',
    packages = [],
    url = 'https://github.com/dracaryop/27_FinalReview',
    license='',
    author='anand',
    author_email = 'anand20280@iiitd.ac.in',
    description = '',
    install_requires = ['bs4', 'nltk', 'scikit-learn', 'numpy']
)
```

XI. DIFFERENCE BETWEEN BASELINE AND FINAL REVIEW

1. Easier command line interface for executing the program.
2. The web crawler now clusters documents in addition to indexing them.
3. Minor bug fixes for the crawler.
4. The program now features a search engine allowing the user to query the page crawled. (Search engine implementation)
5. The index created from the crawler can now be imported and exported from the disk.
6. Getting better and faster result as mentioned in the above section.
7. The getting of result for the search query has been optimized more.

XII. CONTIBUTIONS

There are three members in our group and all the work will be equally distributed. This topic was selected after extensive discussion between the group members and also some help from the previous projects provided to on google classroom. All the 3 members have visited different research papers for reference. Two members will be involved in making the web crawler and search engine and one will be involved in the error checking, modifications, data correction, interface, correcting interface, fixing bugs and recommending things and giving references.

XIII. REFERENCES

[1] Dr Jatinder Singh Bal, Mini Singh Ahuja, Varnica (2014) Web Crawler: Extracting the Web Data

[2] Nemeslaki, András; Pocsarovszky, Károly (2011) Web crawler research methodology

[3] Md. Abu Kausar, V.S. Dhaka, Sanjeev Kumar Singh (2013) Web Crawler: A Review

[4] Gunjan H. Agre, Nikita V. Mahajan (2015) IEEE Keyword Focused Web Crawler

[5] Houqing Lu, Donghui Zhan, Lei Zhou, Dengchao He (2016) Improved Focused Crawler.

Note: Rest references that is not related to the research paper has been mentioned in the readme file. (Used for creating the web-crawler)

XIV. LINK

1. https://youtu.be/WBFuP7GMhrg