Web Search Engine:

COMP8547 - Advanced Computing Concepts Final Project

Instructor: Dr. Mahdi Firoozjaei

This is where your searching begins

University of Windsor

Team Roles

Sahibjeet Singh (110123395)	*Luqmaan Shaik* (110122775)	*Parth Dangaria* (110123538)	*Hetansh Joshi* (110122332)	*Pratik Choudhari* (110073463)
• Searching	• SpellCheck	• SearchWord	• Ranking	• Crawling

Project Link : *https://github.com/ACC-Websearchengine-team/ACC-web-search-engine*

Click Here

University of Windsor

1. Introduction

What is a Web Search Engine?

2. Workflow Diagram

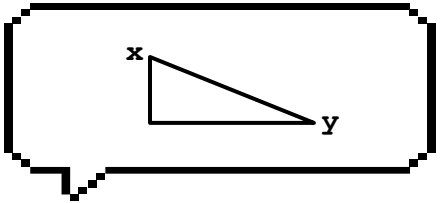
How does our Web Search Engine works?

3. Features Description

How this feature works?

4. Demo

University of Windsor



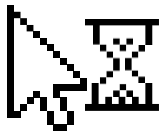
01

Introduction

What is Search Engine ?

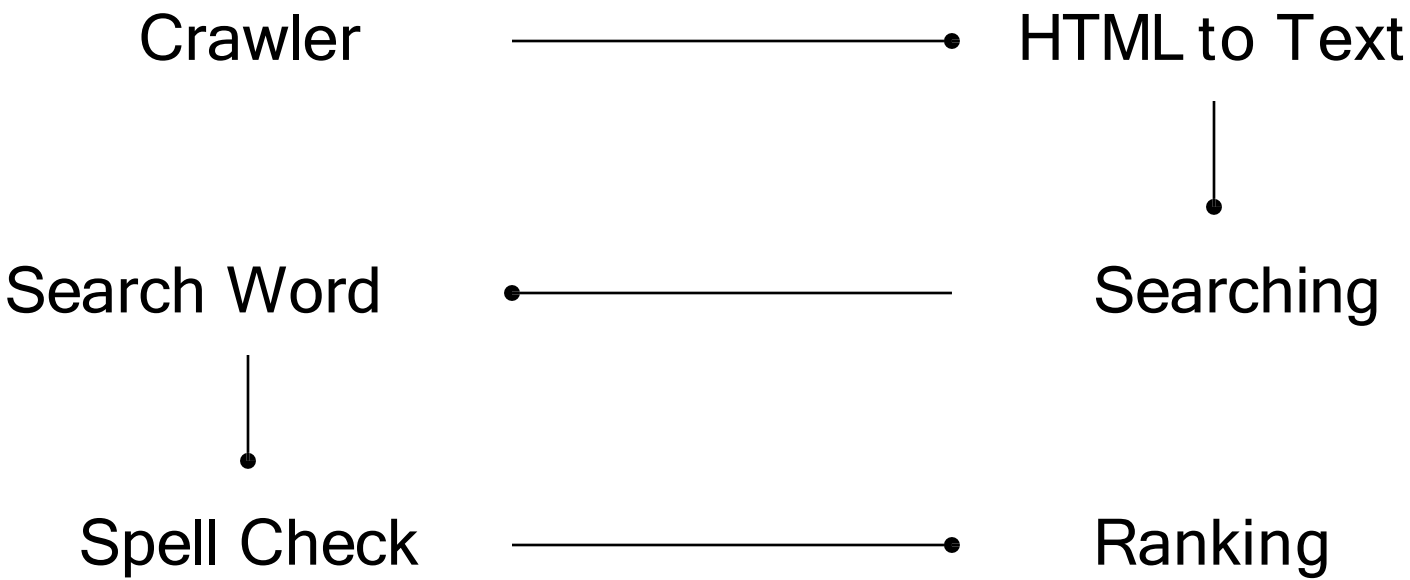
An application created specifically to do web searches is known as a search engine.

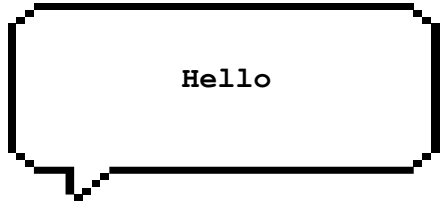
Search engine will look at many web pages to find matches to the user's search inputs. It will return results ranked by relevancy and popularity by the search engine.



University of Windsor

Core Modules

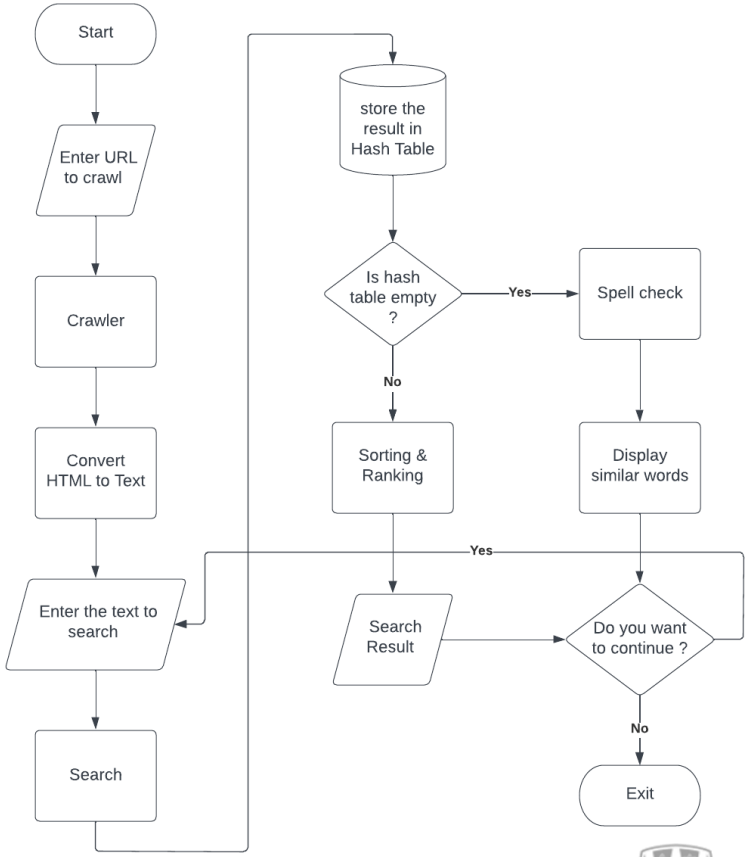




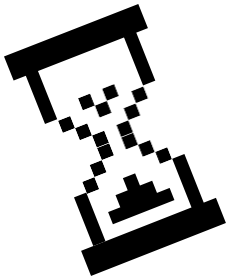
02

Workflow

FLOW CHART



University of Windsor



03

Features

University of Windsor

Crawler

Crawls a URL and adds all other links found in webpage to a hashset

```java
public static void crawlURL(String url) {
    try {
        Document pageContent = Jsoup.connect(url).get();

        // add url only after HTML can be fetched
        visitedLinks.add(url);

        // regex for web url matching
        String pattern = "^((https?://)|(www\\.))[-a-zA-Z0-9+&@#/%?=~_|!:,.;]*[-a-zA-Z0-9+&@#/%=~_|]";
        System.out.println("\nParsing: " + pattern);

        String tmpURL = "";
        // iterate over all anchor tags with href attribute using a css selector
        for (Element anchorTags : pageContent.select("a[href]")) {
            // get the value of href attribute from anchor tag
            tmpURL = anchorTags.attr("abs:href");
            System.out.println(tmpURL);

            // anchor tage with no matching url pattern
            if (!Pattern.matches(pattern, tmpURL)) {
                System.out.println("\nFound URL: " + tmpURL + " => unknown");
            }
            // already visited links
            else if (visitedLinks.contains(tmpURL)) {
                System.out.println("\nFound URL: " + tmpURL + " => ignored because visited");
            }
            // add valid links to crawl
            else {
                visitedLinks.add(tmpURL);
                System.out.println("\nFound URL: " + tmpURL + " => added to crawl list");
            }
            tmpURL = "";
        }
    }
    // catch exception when jsoup can not connect to website
    catch (org.jsoup.HttpStatusException e) {
        System.out.println("\nURL: " + url + " => blocked, not crawled");
    } catch (IOException e) {
        System.out.println("\nURL: " + url + " => I/O error, not crawled");
    }
}
```

HTML to Text

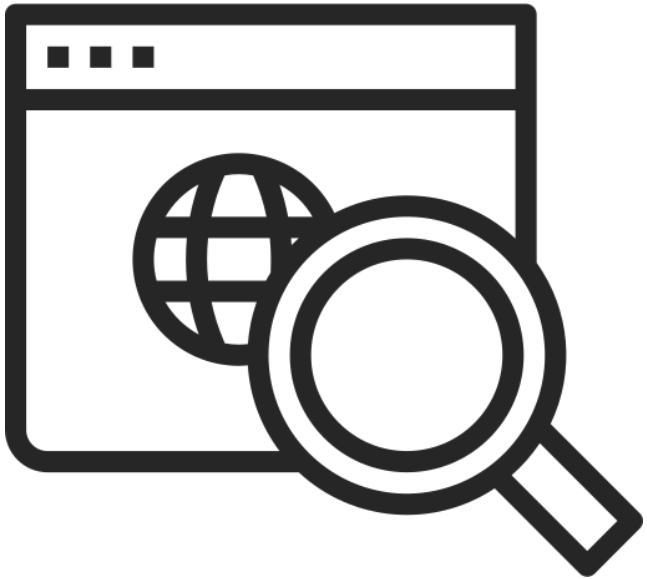
Gets the HTML body of URL visited and saves them as txt file in **assets/textFiles** folder

```
public static void extractTextFromHTML() {
    try {
        String txt, currentURL;
        String filePath = System.getProperty("user.dir") + Constant.FILE_PATH;
        Iterator<String> itr = visitedLinks.iterator();
        while (itr.hasNext()) {
            currentURL = itr.next();
            try {
                Document document = Jsoup.connect(currentURL).get();
                txt = document.text();
                String docTitle = document.title().replaceAll("[^a-zA-Z0-9_-]", "") + ".txt";
                BufferedWriter out = new BufferedWriter(new FileWriter(filePath + docTitle, true));
                out.write(currentURL + " " + txt);
                out.close();
            } catch (org.jsoup.HttpStatusException e) {
                System.out.println("\nURL from page: " + currentURL + " => blocked, not crawled");
            }
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

University of Windsor

Searching

- Takes input from user to search
- Uses Boyer Moore algorithm to search from all the test files generated
- Counting instances of the text in each file and store it using hashing.



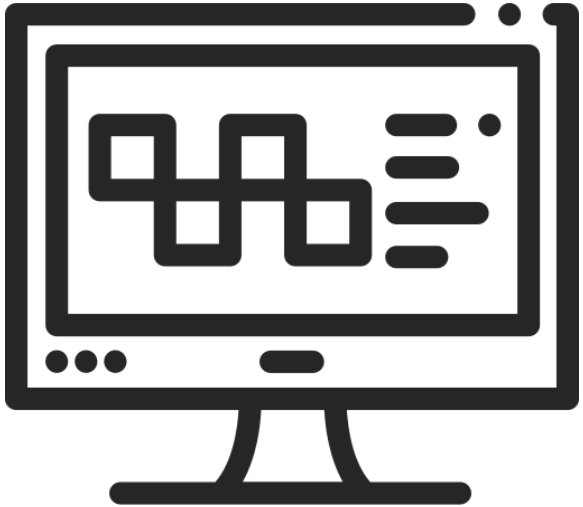
Searching

- Iterate words through all the files present.
- Consider each file as a set of characters and try to match using Boyer Moore Algorithm.

```java
package WebSearchEngine;

public class BoyerMoore {
    private final int R;     // the radix
    private int[] right;     // the bad-character skip array

    private char[] pattern;  // store the pattern as a character array
    private String pat;      // or as a string

    // pattern provided as a string
    public BoyerMoore(String pat) {
        this.R = 100000;
        this.pat = pat;

        // position of rightmost occurrence of c in the pattern
        right = new int[R];
        for (int c = 0; c < R; c++)
            right[c] = -1;
        for (int j = 0; j < pat.length(); j++)
            right[Character.toLowerCase(pat.charAt(j))] = j;
    }

    // return offset of first match; N if no match
    public int search(String txt) {
        int M = pat.length();
        int N = txt.length();
        int skip;
        for (int i = 0; i <= N - M; i += skip) {
            skip = 0;
            for (int j = M-1; j >= 0; j--) {
```

Search Word

- A hash table is used to store the results from searching.
- File names are stored as key and count of occurrences in that file are stored as value.
- Further this hash table is used for sorting and ranking.



University of Windsor

Search Word

- Hash table will save all the records from search operation using separate chaining method.
- Hash table will increase its capacity once 75% of the table is filled.

```java
public static int wordSearch(String word, File filePath) throws IOException {
    int count = 0;
    StringBuilder data = new StringBuilder();
    try {
        BufferedReader reader = new BufferedReader(new FileReader(filePath));
        String str = null;

        while ((str = reader.readLine()) != null) {
            data.append(str);
        }
        reader.close();
    } catch (Exception e) {
        System.out.println("Exception:" + e);
    }
    // Find position of the word
    String text = data.toString();

    int offset = 0, loc =0;

    while (loc <= text.length()) {
        offset = SearchEngine.search1(word, text.substring(loc));
        if ((offset + loc) < text.length()) {
            count++;
            System.out.println("\n" + word + " is at position " + (offset + loc) + "."); // printing the position of the word
        }
        loc += offset + word.length();
    }

    // If the word is found, print the file name where it is found
    if (count != 0) {
        System.out.println("-----------------------------------------------");
        System.out.println("\nWord found in " + filePath.getName());
        System.out.println("-----------------------------------------------");
    }
    return count;
}

// Finds strings with similar patterns and calls edit distance on those strings.
public static void findData(File sourceFile, int fileNumber, Matcher matcher, String p1)
        throws FileNotFoundException, ArrayIndexOutOfBoundsException {
    EditDistance.findWord(sourceFile, fileNumber, matcher, p1);
}
```

University of Windsor

Edit Distance

- Edit distance is calculated by measuring how many operations are required to convert one string into the other.
- Operations :
    - Insertion
    - Deletion
    - Replacement

Edit Distance

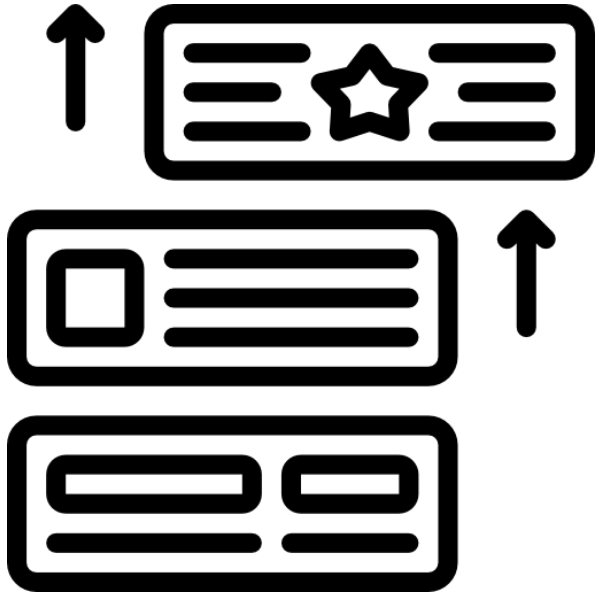
- All text files are tokenized and stored in a list.
- For every unique string in the list the input is matched and the word with least edit distance is returned as output.

```java
public static int findEditDistance(String word1, String word2)
{
    int len1 = word1.length();
    int len2 = word2.length();

    // len1+1, len2+1, because finally return dp[len1][len2]
    int[][] dp = new int[len1 + 1][len2 + 1];

    for (int i = 0; i <= len1; i++) {
        dp[i][0] = i;
    }

    for (int j = 0; j <= len2; j++) {
        dp[0][j] = j;
    }

    //iterate though, and check last char
    for (int i = 0; i < len1; i++) {
        char c1 = word1.charAt(i);
        for (int j = 0; j < len2; j++) {
            char c2 = word2.charAt(j);

            //if last two chars equal
            if (c1 == c2) {
                //update dp value for +1 length
                dp[i + 1][j + 1] = dp[i][j];
            } else {
                int replace = dp[i][j] + 1;
                int insert = dp[i][j + 1] + 1;
                int delete = dp[i + 1][j] + 1;

                int min = replace > insert ? insert : replace;
                min = delete > min ? min : delete;
                dp[i + 1][j + 1] = min;
            }
        }
    }
    return dp[len1][len2];
}
```

University of Windsor

Ranking

- Ranking is the order in which the indexed results appear on the result page
- Sorting operation is performed to get the ranking of the results



Ranking - Sorting

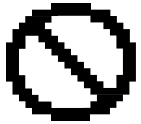
- Sorting the hash table according to the number of occurrences.
- Displaying top 5 file names having highest value in the hash table.

```java
public class Sorting {
    /**
     * Sorts web pages based on their occurrence using merge sort.
     *
     */
    public static void sortWebPagesByOccurrence(Hashtable<?, Integer> t, int occur) {
        // Organize the list of hashtable entries by sorting it.
        ArrayList<Map.Entry<?, Integer>> entryList = new ArrayList<>(t.entrySet());
        Collections.sort(entryList, new Comparator<Map.Entry<?, Integer>>() {
            public int compare(Map.Entry<?, Integer> o1, Map.Entry<?, Integer> o2) {
                return o1.getValue().compareTo(o2.getValue());
            }
        });
        //Reverse the sorted list to get it in descending order of occurrence
        Collections.reverse(entryList);
        // Display the sorted web page rankings if the 'occur' flag is not zero
        if (occur != 0) {
            System.out.println("\n-----------------------------Web Page Ranking-----------------------------\n");
            int k = 5; // number of top results to be shown
            int l = 0;
            System.out.printf("%-10s %s\n", "Sr. No.", "Name and Occurrence");
            System.out.println("------------------------------------------------");
            while (l < entryList.size() && k > 0) {
                System.out.printf("\n%-10d| %s\n", l + 1, entryList.get(l));
                l++;
                k--;
            }
            System.out.println("\n--------------------------------------------------------------\n");
        }
    }
}
```

University of Windsor

04

Demo



Thank You

University of Windsor