University of Windsor

COMP8547-Advanced Computing Concepts

Project-FALL 2022

Real Estate Web Search Engine

Instructor: Dr. Mahdi Firoozjaei

University of Windsor

Team Crawlers - Roles

Richa Singh 110093085

- Spell Checker

Sreekanth Vinodkumar 110093577

- Searching based on Bedrooms
- Recent Search History

Tanmay Verma 110089806

- Hashing
- Sorting
- Listing

Venkata Naveen Varma Vegesna 110090483

- Crawling
- HTML to Text Conversion

Venkata Rahul Nittala 110094777

- Page Ranking

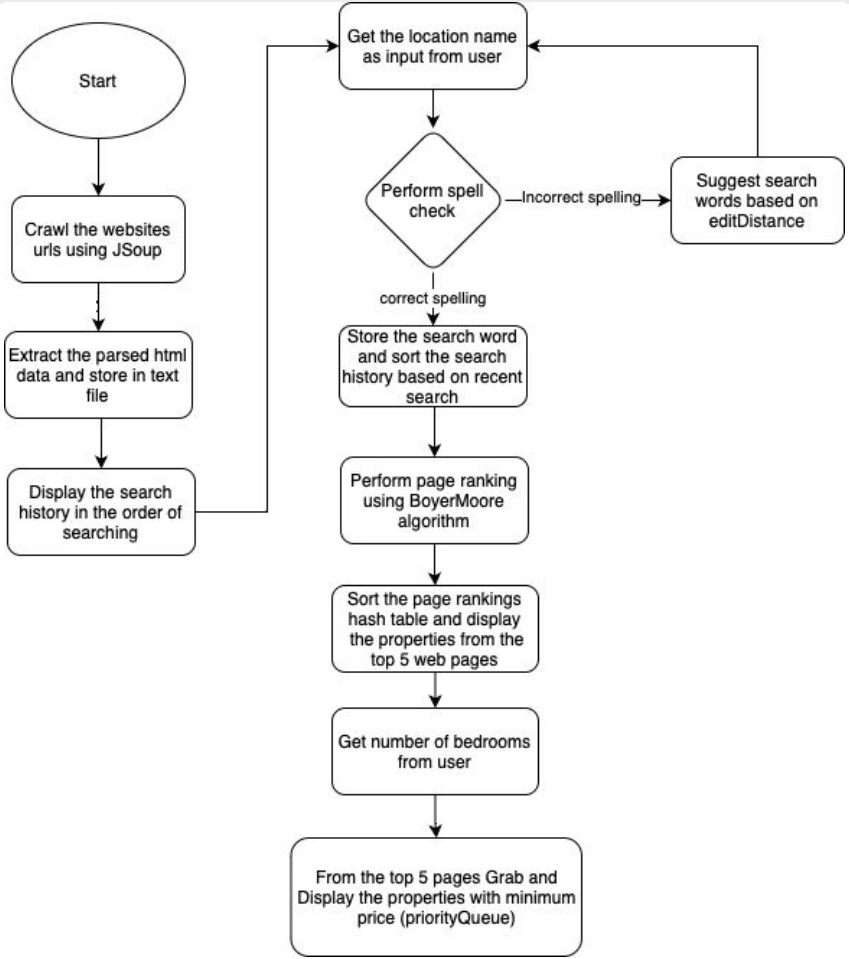
University of Windsor

Introduction

Real Estate search engine provides the listings of properties based on the property location, number of bedrooms and price. To achieve this goal our search engine will process through several web pages from the real estate website called “https://www.royallepagebinder.com/” to find matches to the user's search inputs.

University of Windsor

Workflow



University of Windsor

Features

University of Windsor

Crawling

Web Crawling is browsing the Web systematically to retrieve information without user intervention.

- Scrape and parse HTML from a URL.
- Crawling n number of URLs and adding the valid URLs to the HashSet and storing the html data into a “.html” file.

```java
public class Crawler {

    static HashSet<String> uniqueLinks = new HashSet<String>();
    private static int max_pages = 50;

    // Get links from the given url website
    public static void crawl(String url, int page_count)
    {
        try {
            if(!uniqueLinks.contains(url) && uniqueLinks.size() < max_pages) {
                if(page_count>0) {
                    // Add the url into the hashSet
                    uniqueLinks.add(url);
                    // download the html data and store as html files, from the webpage url
                    String currentUrl = url;
                    Document doc = Jsoup.connect(currentUrl).get();
                    String html = doc.html();
                    String filePath = "src/resources/htmlFiles/";
                    // Using regular expression to format the url
                    String fileName = currentUrl.replaceAll("[^a-zA-Z0-9_-]", "") + ".html";
                    File tmpFile = new File(filePath);
                    boolean fileExists = tmpFile.exists();
                    // Check if file exists or not
                    if(!fileExists) {
                        // Write the formatted data into a html file.
                        BufferedWriter out = new BufferedWriter(new FileWriter(filePath + fileName, true));
                        out.write(url + " " + html);
                        out.close();
                    }
                }
                page_count++;
                // Using Jsoup library to scrape and parse HTML from a URL
                Document document= Jsoup.connect(url).get();
                Elements linkpage= document.select("a[href^=\"https://www.royallepagebinder.com/residential-properties/page\"]");
                // Loop through each page link in the webpage
                for(Element page: linkpage)
                {
                    crawl(page.attr("abs:href"), page_count);
                }
            }
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    }
```

University of Windsor

HTML to Text

For each html web-page file extracted,

- Using Jsoup library, extract all property details from a web-page.
- Format the property details extracted.
- And store in a text file to make further processing easy.

```java
public class Convert_to_text {

    // Fetch html page files and scrap the data from it in required format and store it in a text file.
        public static void convert_html_to_text()
        {
            System.out.println("\nFetch and convert all .html files to .txt files");
            try {
                String source_path = "src/resources/htmlFiles/";
                String dest_path = "src/resources/textFiles/";
                File dir = new File(source_path);
                String files_list[] = dir.list();
                String address, price, type;

                // For each file, fetch property details
                for(String filename: files_list) {
                    File currentFile = new File(source_path + filename);
                    // Using Jsoup to parse the data in the file
                    Document doc = Jsoup.parse(currentFile, "UTF-8");
                    // Fetching all the property listings in that file
                    Elements property_listings = doc.getElementsByClass("listing_quick_info");
                    // For each property,format the data as required
                    for(Element property: property_listings) {
                        // Fetch data
                        address = property.getElementsByClass("address").text();
                        price = property.getElementsByClass("price").text();
                        type = property.getElementsByClass("type").text();
                        // Format data
                        address = address.replace(",", "");
                        price = price.replace("$", "");
                        price = price.replace(",", "");
                        type = type.replace("|", "");
                        String print_value = address+","+price+","+type;
                        String fileName = filename.replace(".html", ".txt");
                        File tmpFile = new File(dest_path);
                        boolean fileExists = tmpFile.exists();
                        // Check if file exists or not
                        if(!fileExists) {
                            // Write the formatted data into a text file.
                            BufferedWriter out = new BufferedWriter(new FileWriter(dest_path + fileName, true));
                            out.write(print_value + "\n");
                            out.close();
                        }
                    }
                }
                System.out.println("\nAll files converted to .txt!!\n");
            }
            catch(Exception e) {
                e.printStackTrace();
            }
        }
```

University of Windsor

Page Ranks

Page Rank is used to determine the rank of web page based on the comparison of the search term entered by the user in all the web pages.

- Read the downloaded web page data from the text files
- Search for the word occurrence in each file by comparing with the search term using the Boyers Moore algorithm
- Storing the count of word occurrence in the hashmap

```
public class Wordranks {
    Map<String, Integer> ranks= new HashMap<String, Integer>();   //hashmap to store the rank of word and page

    /**
     * Get the rank of the pages and the frequency of words based on the search key passed
     */
    public Map<String,Integer> getPageRank(String searchKey) {
        searchKey=searchKey.toLowerCase();          //converting the input to lower cases
        File directory=new File("src/resources/textFiles");
        File filesList[] = directory.listFiles();   //get the list of files from the directory
        /**
         *  From the list of all files read each file convert to the lower case.
         */
        for(File file : filesList) {
            String line = "";
            int c=1;
            FileReader fr = null;
            try {
                fr = new FileReader(file.getPath());
            } catch (FileNotFoundException e1) {
                e1.printStackTrace();
            }
            Scanner scanFile=new Scanner(fr);
            while(scanFile.hasNext()) {
                line=line+" "+ scanFile.nextLine();
            }
            line=line.toLowerCase();        //convert the content to the lower case after read from file
            line=line.trim();               //remove the spaces
            int startIndex=0;
            /**
             *  For each file use the Boyers Moore algorithm for searching the word count.
             */
            for(int index=0;index<=line.length();index=index+startIndex+searchKey.length()) {
                BoyerMoore bm= new BoyerMoore(searchKey);
                startIndex=bm.search(line.substring(index));
                if(startIndex+index<line.length())
                {
                    ranks.put(file.getName(), c);   //store the web page name in a hash map
                    c= ranks.get(file.getName())+1; //update the frequency of the word occurrence
                }
            }
            scanFile.close();
        }
```

Page Sorting

Page Sorting is used to sort the pages based on the ranking of the pages for the search location.

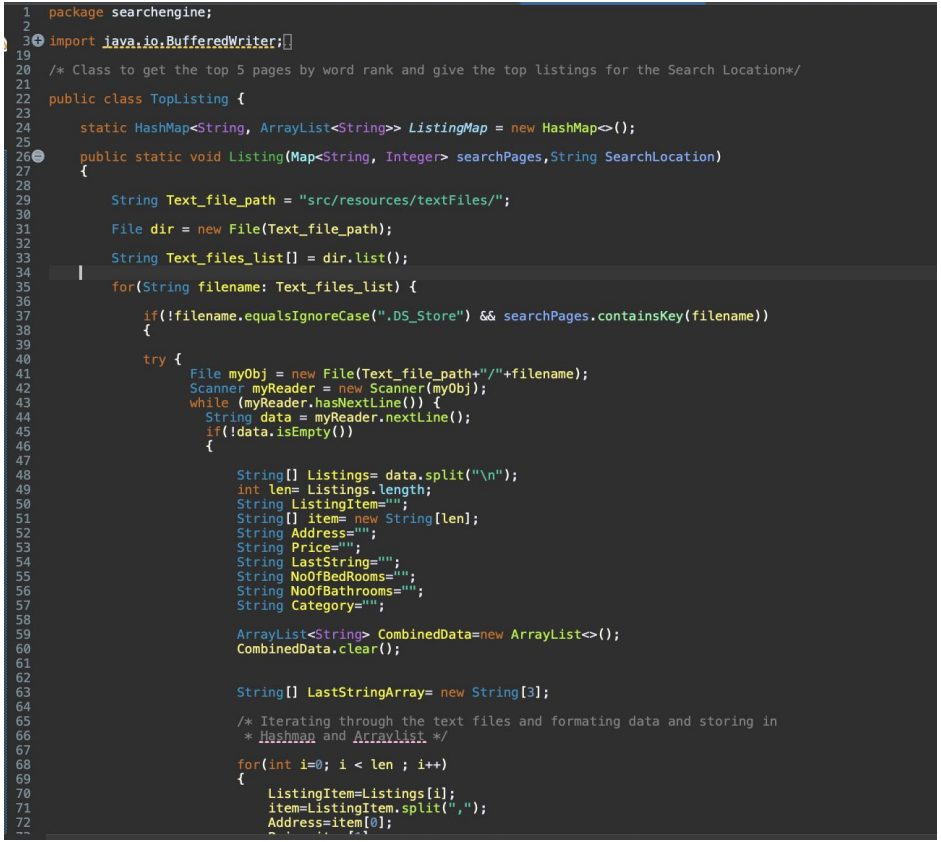
- It utilizes the Stream API of Java for sorting in descending order by using the comparingByValue() method and it returns a comparator that compares the values.
- Page Sorting class also gives top 5 pages out of the list which has the highest ranking.

```java
package searchengine;

import java.util.Comparator;

public class PageSorting {

    public static List<Entry<String, Integer>> SortMap(Map<String, Integer> webpages) {

        List<Entry<String, Integer>> list = null;

        if (!webpages.isEmpty()) {

            System.out.println("\nPages after sorting based on no. of occurences : ");

            /* Using Stream API of Java 8 for sorting Hash map values.
             * In java 8, Map.Entry class has static method comparingByValue()
             * to help in sorting by values. This method returns a Comparator
             * that compares Map.Entry in natural order on values.*/

            Stream<Map.Entry<String, Integer>> sorted = webpages.entrySet().stream()
                    .sorted(Map.Entry.comparingByValue(Comparator.reverseOrder()));

            System.out.println("Top 5 pages with the best results are : -");

            list = sorted.toList();

            for (int k = 0; k < 5; k++) {
                System.out.println(
                        "Page Name : " + list.get(k).getKey() + "\nNo. of Occurences : " + list.get(k).getValue());

            }

        }

        return list; // Returning the sorted list of the top 5 pages with highest word ranking
    }
}
```

Top Listing

Top Listing takes the input as the Top 5 pages sorted by the page sorting feature and then reads the text files for those top 5 pages.

- It then processes that data to format, standardise and filter the data.
- Once filtered the data is stored in a Hashmap which takes Address of the listing as the Key and then Price, No. of bedroom, No. of Bathrooms and Category in a Arraylist and then passes the arraylist as the value for the corresponding key.
- This gives the data for the top listings from top 5 pages for the search location and also passes that further for searching the best deal.

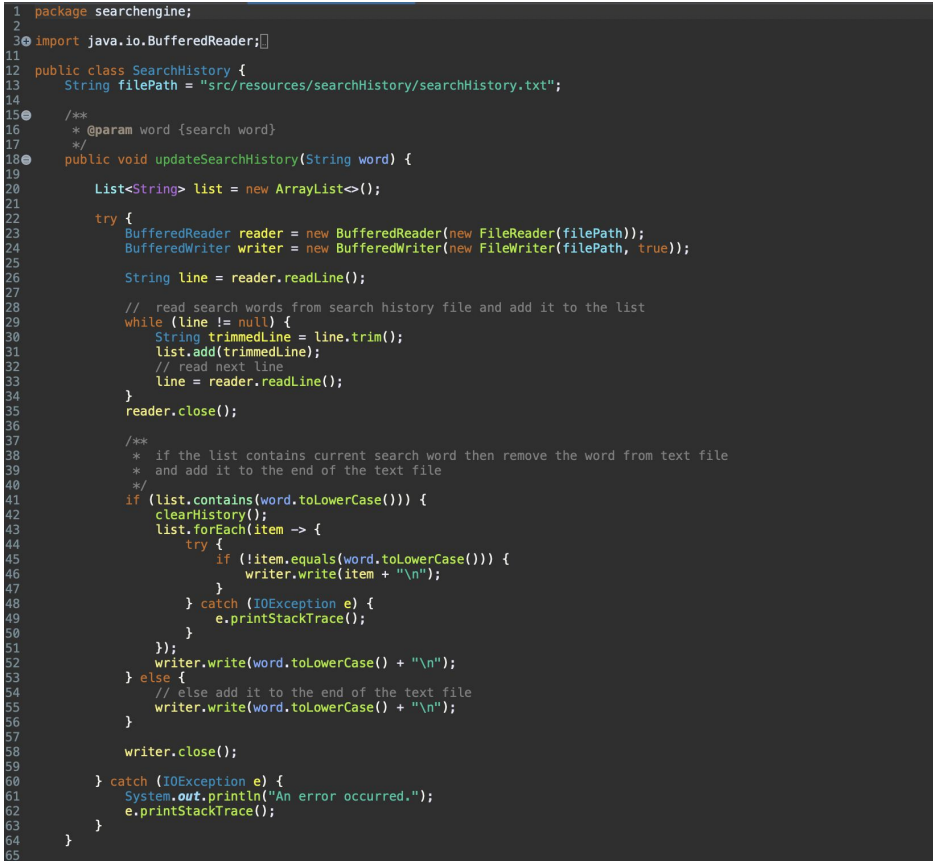


```java
package searchengine;

import java.io.BufferedWriter;

/* Class to get the top 5 pages by word rank and give the top listings for the Search Location*/

public class TopListing {

    static HashMap<String, ArrayList<String>> ListingMap = new HashMap<>();

    public static void Listing(Map<String, Integer> searchPages,String SearchLocation)
    {

        String Text_file_path = "src/resources/textFiles/";

        File dir = new File(Text_file_path);

        String Text_files_list[] = dir.list();

        for(String filename: Text_files_list) {

            if(!filename.equalsIgnoreCase(".DS_Store") && searchPages.containsKey(filename))
            {

            try {
                  File myObj = new File(Text_file_path+"/"+filename);
                  Scanner myReader = new Scanner(myObj);
                  while (myReader.hasNextLine()) {
                    String data = myReader.nextLine();
                    if(!data.isEmpty())
                    {

                        String[] Listings= data.split("\n");
                        int len= Listings.length;
                        String ListingItem="";
                        String[] item= new String[len];
                        String Address="";
                        String Price="";
                        String LastString="";
                        String NoOfBedRooms="";
                        String NoOfBathrooms="";
                        String Category="";

                        ArrayList<String> CombinedData=new ArrayList<>();
                        CombinedData.clear();


                        String[] LastStringArray= new String[3];

                        /* Iterating through the text files and formating data and storing in
                         * Hashman and Arraylist */

                        for(int i=0; i < len ; i++)
                        {
                            ListingItem=Listings[i];
                            item=ListingItem.split(",");
                            Address=item[0];
```

University of Windsor

Search History

- Save search words to database each time when user searching a word.
- Returns search words based on recent searches.



```java
package searchengine;

import java.io.BufferedReader;

public class SearchHistory {
    String filePath = "src/resources/searchHistory/searchHistory.txt";

    /**
     * @param word {search word}
     */
    public void updateSearchHistory(String word) {

        List<String> list = new ArrayList<>();

        try {
            BufferedReader reader = new BufferedReader(new FileReader(filePath));
            BufferedWriter writer = new BufferedWriter(new FileWriter(filePath, true));

            String line = reader.readLine();

            //  read search words from search history file and add it to the list
            while (line != null) {
                String trimmedLine = line.trim();
                list.add(trimmedLine);
                // read next line
                line = reader.readLine();
            }
            reader.close();

            /**
             *  if the list contains current search word then remove the word from text file
             *  and add it to the end of the text file
             */
            if (list.contains(word.toLowerCase())) {
                clearHistory();
                list.forEach(item -> {
                    try {
                        if (!item.equals(word.toLowerCase())) {
                            writer.write(item + "\n");
                        }
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                });
                writer.write(word.toLowerCase() + "\n");
            } else {
                // else add it to the end of the text file
                writer.write(word.toLowerCase() + "\n");
            }

            writer.close();

        } catch (IOException e) {
            System.out.println("An error occurred.");
            e.printStackTrace();
        }
    }

```

University of Windsor

Best deal based on number of bedrooms

- Passing filtered data to priority queue. Filtered data contains building details in a specified location that the user has looked for.
- Generating priority queue based on price.
- Priority queue contains list of objects with details of building like address, price, number of bedrooms, and number of washrooms.
- User will be able to get plot with best price based on the number of bedrooms.

```
    /**
     *
     * @param bedroomNo {number of bedrooms}
     * @return String {details of best deal}
     */
    public static String bestDeal(int bedroomNo) {
        while (!buildingPriorityQueue.isEmpty()) {
            Buildings building = buildingPriorityQueue.poll();


            if (bedroomNo == building.bedrooms) {
                return "Address: " + building.address + ",\nPrice: $" + building.price + ",\nNumber of bedrooms: "
                        + building.bedrooms + ",\nNumber of washrooms: " + building.washrooms;

            }

        }
        return "Not able to find a residence with given number of bedrooms";
    }
}
```

University of Windsor

Spell Checking

- A spell checker is a feature that is used to check the misspellings of the text.
- For performing spell check in our project we are using Edit Distance algorithm.
- Creating dictionary from the web pages.
- Using the dictionary to calculate the edit distance

```java
package searchengine;

import java.util.Random;

public class Sequences {

    public static int editDistance(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];
    }
}// class
```

University of Windsor

DEMO

Github Link: https://github.com/venkata-naveen-varma-v/ACC_Project_Team_Crawlers

University of Windsor

Thank You