School of Computer Science

Masters in Applied Computing (M.A.C)

Subject Code: COMP – 8547

Subject Name: Advanced Computing Concepts

Professor: Dr. Olena Syrotkina

Report Submission

by

Status 200

Chandini Priya Kondru – 110094181

Valiant Dmello – 110089739

Darshil Patel – 110091604

Naiket Patel - 110089315

Krupal Rudani - 110092052

Variant: - Top Mobile Analysis

The idea of the project is to analyze and understand different mobile phone offerings and find the best one according to the user's criteria.

1. Choose a website that cells mobile phones (e.g., the source.ca).
2. Specify the details of your search (keyword, company, etc.)
3. Find the best offerings in the selected criteria.

Introduction: -

The project uses a mobile seller website to crawl through web and use different algorithms and data structures to give web page suggestion as per input keywords.

Group members and their respective tasks: -

Name	Features
Darshil Patel	Web Crawler HTML Parser
Chandini Priya kondru	Spell Checking Search Frequency
Valiant Dmello	Word Completion Code Integration
Naiket Patel	Inverted Indexing Frequency Count
Krupal Rudani	Data Validation using Regex Page Ranking

Web Crawler: -

In web crawler, we have selected www.bestbuy.ca

For crawling, we are giving three input parameters: - Depth, URL and Regular Expression

It will start with depth 0 and it will store urls in visited sites HashSet.

Then function will get called recursively with incrementing depth for all the urls in the parent webpage and it goes till the max depth.

The initial url we used: https://www.bestbuy.ca/en-ca/category/best-buy-mobile/20006

Max Depth: 30

Regex: "https://www.bestbuy.ca/en-ca(//w+)*"

We use regex on every links to check if the web crawler is not crawling out of the main website.

Output: - We got 1123 urls after crawling [1]

WebCrawler.java

```
public static void webCrawl(int depth, String url, HashSet<String> visitedPages, int maxDepth, String regex) {
    if(depth<=maxDepth) {
        Pattern pattern = Pattern.compile(regex);    //regex to let crawl only in bestbuy website
        Matcher matcher = pattern.matcher(url);
        if(matcher.find()) {
            Document doc = request(url, visitedPages);
            if(doc!=null) {
                for(Element link: doc.select("a[href]")) {
                    String nextLink = link.absUrl("href");
                    if(!visitedPages.contains(nextLink)) {  //if page not already visited
                        webCrawl(depth++, nextLink, visitedPages, maxDepth, regex);     //recursively crawl in each link with
                    }
                }
            }
```

Output: -

```
link: https://www.bestbuy.ca/en-ca/category/best-buy-mobile/20006
link: https://www.bestbuy.ca/en-ca/about/best-buy-
business/bltfad9143fefc09dc6
link: https://www.bestbuy.ca/en-ca
link: https://www.bestbuy.ca/en-ca/collection/black-friday/413517
link: https://www.bestbuy.ca/en-ca/collection/pokemon-scarlet-
violet/407459
link: https://www.bestbuy.ca/en-ca/event/gift-ideas/bltcffbb89dd264d25c
link: https://www.bestbuy.ca/en-ca/collection/laptops-on-sale/46082
link: https://www.bestbuy.ca/en-ca/collection/christmas-trees-
decorations/264141
link: https://www.bestbuy.ca/en-ca/collection/best-buy-outlet/113080
link: https://www.bestbuy.ca/en-ca/category/computers-tablets/20001
link: https://www.bestbuy.ca/en-ca/category/office-supplies-ink/30957
link: https://www.bestbuy.ca/en-ca/category/tv-home-theatre/20003
link: https://www.bestbuy.ca/en-ca/category/audio/659699
link: https://www.bestbuy.ca/en-ca/category/cameras-camcorders/20005
link: https://www.bestbuy.ca/en-ca/category/car-electronics-gps/20004
link: https://www.bestbuy.ca/en-ca/category/appliances/26517
link: https://www.bestbuy.ca/en-ca/collection/unlocked-phones-on-
sale/245148
link: https://www.bestbuy.ca/en-ca/category/smart-home/30438
link: https://www.bestbuy.ca/en-ca/category/home-living/homegardentools
```

HTML Parser: -

In HTML Parser, we are using output of web crawler and taking all the urls as input.

Iterating through each url, we are calling JSOUP Library for extracting webpages content to text.

This text is stored in the individual text files for each url.

We are using naming convention as 0.txt, 1.txt,….and so on.

HTMLParser.java

```
public static String htmlParser(String url) throws IOException {
    Document doc = Jsoup.connect(url).get();     // connect to url
    String text = doc.title().concat(doc.body().text());    //Extract content from site to String text
    return text;
}
```

Search Frequency: -

1. In this user will be entering the word/words how many ever they want to get a fine search on the bestbuy.ca
2. Then I will be searching whether the word is present or not if not will add the word and add the count as 1. If it is present, we will be incrementing the count of word by 1.
3. I am using the hash map as it doesn't allow duplicate keys and, I will be storing the count of the words, word as value and keys respectively returning the entire hash map of search history.

   Example:

   {mobile=2, Samsung=1}

   mobile and Samsung are keys(words)

   2,1 are count stored as values.

Search.java

```
//Takes the word for every iteration.
public static void searchUpdate(String word) {

    //checks if word is not first in the list so that we directly add count to it.
    if(searchMap.keySet().size()>0) {
            // if the word is already present we increment the count.
            if (searchMap.containsKey(word)) {
                    searchMap.put(word, searchMap.get(word)+1);
                }
                else {
                    searchMap.put(word, 1);
                }
            }
    // if the word is first one we directly put the count as 1
        else {
            searchMap.put(word, 1);
        }
    }
```

Output: -

```
The input words are :
[mobile, samsung, mobile, mobile, iphone, 22]
Search History : {22=1, samsung=1, mobile=3, iphone=1}
```

Inverted Indexing and Frequency Count: -

- In inverted indexing, we are iterating through each text file.
- We are using HashTable and storing string as a key and inside it using ArrayList of integers as value to store frequency of each word at the index of the files i.e. index 0 for 0.txt, index 1 for 1.txt
- For storing frequency, opening each file and reading each words.
- If word comes for the first time then it will be added in the HashTable and its frequency will be one. If same word is repeated one more time then the frequency of the word will be incremented or else new word will be added in to the HashTable.

InvertedIndexing.java

```
for(int i =0; i<fileNames.length; i++) {//iterate through all .txt files
    BufferedReader reader = new BufferedReader(new FileReader(dir+fileNames[i]));   //read file
    String text = "", line;
    String link = reader.readLine();    //website link that is stored in first line of .txt file
    while((line = reader.readLine())!=null) {
        text = text + line;         //add all text from file to text String
    }
    for(String word : text.split(" ")) {    //iterate throught each word
        word = word.toLowerCase().replaceAll("[^a-zA-Z0-9\\s]", "");
        if(word!="") {
            if(invertedIndex.containsKey(word)) {   //if word exist in map
                ArrayList<Integer> list = invertedIndex.get(word);
                int freq = list.get(i);
                list.set(i, freq+1);    //increament frequency
                invertedIndex.put(word, list);

            }else {
                ArrayList<Integer> list = new ArrayList<Integer>(Collections.nCopies(fileNames.length, 0));
                list.set(i,1);  //in list make frequency of word for ith index of list 1
                invertedIndex.put(word, list);  //Add word with list
            }
        }
    }
```

Word Completion: -

The output of Inverted Index is used as input for this function.

All the words stored from inverted indexing were used to create a Trie.

Each node of Trie stores a character, a Boolean and a SortedMap to store all children.

Sorted Map is a part of java collection frameworks that implements the red black tree, this helps store children in sorted order.

The inputs words are checked in the Trie.

The first Alphabetically available word in the trie is provided form word completion.

WordCompletion.java

```
class Node{
    public char c;  //node character
    public boolean isWord;  //to check if word completes at this node
    public SortedMap<Character,Node> children;  //SortedMap of character and nodes for the children of this node

    public Node(char c) {   //constructor  for Node
        this.c = c;
        isWord = false;
        children = new TreeMap<>();
    }
}

public class Trie {
    private static Node root;   //root node for Trie

    //Trie Constructor
    public Trie() {
        root = new Node('\0');  //root node empty node
    }

    //method to add node to trie
    public void add(String word) {
        Node curr = root;   //assign current node to root node
        for(int i=0; i<word.length(); i++) {    //iterate throught each character of word
            char c = word.charAt(i);
            if (curr.children.get(c)==null) {   //check if node for char c exists
                curr.children.put(c, new Node(c));  //create node if doesnt exit
            }
            curr = curr.children.get(c);    //travel to next child of the char of word
        }
        curr.isWord = true;
    }
```

```
//method to return same string if word exists in trie, return completed word if trie finds one, else return empty string
public String checkTrie(String word) {
    Node node = getNode(word);  //get node where the word end
    if(node!=null && node.isWord) { //word found in trie
        return word;
    }
    if(node!=null) {    //incomplete word found in trie
        String s = getWord(node);   //get complete word
        return word+s;  //return complete word
    }
    return "";  //if trie can find the input string
}
```

Output: -

```
please enter the keyword to search :
samsun
inputs = samsun
"word" samsun doesnt exist in dictionary using "samsung" instead
You want to search again. Please type N to exit and enter Y key continue
```

Spell Checking: -

1. In spell checking, we are using all the text files as an input.
2. Iterating through each text files, we will be creating inverted hash index where we have words as keys and values as array list of count occurred in each text file.
3. But for spell checking we will be taking the words from inverted hash map which are keys and try to form dictionary of words.
4. So, now the user enters the word we will be comparing with every word we formed in dictionary and if the word isn't present then we calculate the edit distance of two words.
5. Then we will be comparing whose edit distance is 1 and then suggest the respective word to the user allowing him to re-enter the word.
6. So, here I used 2d array for calculating the edit distance between the words.

Example:

User enters mobie

Spell checking will be displaying movie, mobile as suggestions to the user.

SpellChecking.java

```
private static ArrayList<String> dictionaryCreation(HashMap<String, ArrayList<Integer>> invertedIndex) {
    ArrayList<String> arr = new ArrayList<String>();
    //iterating through each and every word and adding it into the dictionary so that we have the dictionary of words.
    Iterator iter = invertedIndex.entrySet().iterator();
    while (iter.hasNext()) {
        HashMap.Entry entry = (HashMap.Entry) iter.next();
        String key = (String) entry.getKey();
        arr.add(key);
    }
    return arr;
}
```

```
public static ArrayList<String> wordCorrectionSuggestion(String word,
        HashMap<String, ArrayList<Integer>> invertedIndex) {
    ArrayList<String> wordCorrector = new ArrayList<String>();

    try {
        ArrayList<String> temp = new ArrayList<String>();
        int i = 0;
        int distance = 0;
        //Storing the dictionary of words as array list.
        temp = dictionaryCreation(invertedIndex);

        if (!temp.contains(word)) {

            for (i = 0; i < temp.size(); i++) {
                // Finding the difference in distance between the words
                distance = SpellChecking.editDistance(word, temp.get(i));
                // Adding a alternate word if the distance is equal to 1
                if (distance == 1) {
                    wordCorrector.add(temp.get(i));
                }
            }

        }
    } catch (Exception e) {
        System.out.println(e);
    }

    //returning word corrector after correcting it
    return wordCorrector;
```

Output: -

```
please enter the keyword to search :
mobie
inputs = mobie
"word" mobie doesnt exist in dictionary
[movie, mobile]
You mean the above words? Let's try re-enter
please enter the keyword to search :
```

Page Ranking: -

For Page ranking, we are using maximum heap as a data structure.

We are iterating through each file and counting words and along with storing its frequency we are also storing index of that file.

We use Java Collections PriorityQueue which implements heap data structure.

Then in getTopNPages method, we are using poll() method of priorityqueue which returns the root node of heap.

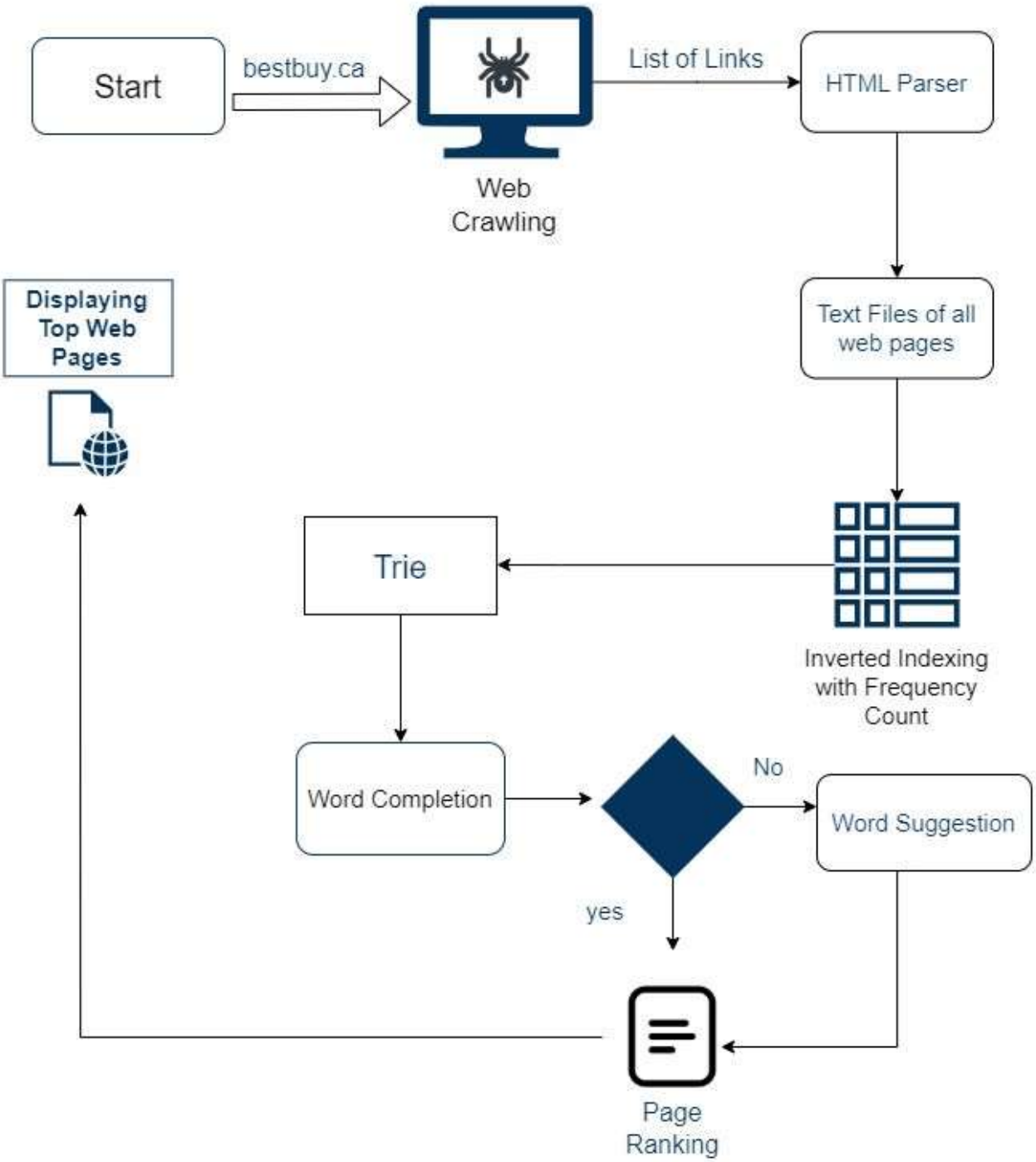
PageRanking.java

```
public static PriorityQueue<Pair<Integer, Integer>> pageRank(String [] words, HashMap<String,ArrayList<Integer>> invertedIndex) {
    PriorityQueue<Pair<Integer, Integer>> pageRank = new PriorityQueue<>((a, b) -> b.getKey() - a.getKey());    //Maximum heap to store pa
    for(int i=0; i<invertedIndex.get(words[0]).size(); i++) {   //iterate through inverted Indexes for each page
        int score = 0;
        for(String word: words) {
            word = word.toLowerCase().replaceAll("[^a-zA-Z0-9\\s]", "");
            score =  score + invertedIndex.get(word).get(i);    //update score
        }
        pageRank.add(new Pair<Integer, Integer>(score, i)); //get score for each page and add to heap
    }
    return pageRank;
}

public static void getTopNpages(int n, PriorityQueue<Pair<Integer, Integer>> pageRank, String fileNames[], String dir) throws IOException
    for(int i=0; i<n; i++) {    //iterate n times
        Pair<Integer, Integer> pair = pageRank.poll();  //get root and delete from heap
        int score = pair.getKey();  //score of the root node
        int fileIndex = pair.getValue();    //file index

        BufferedReader reader = new BufferedReader(new FileReader(dir+fileNames[fileIndex]));
        String link = reader.readLine();    //read link stored on first line of txt file
        System.out.println("Rank "+(i+1)+" of page : "+link+" score : "+score);
    }
}
```

Project Workflow: -



References: -

[1]
"Simple Web Crawler in 50 Lines of Java Code!," *www.youtube.com*. https://www.youtube.com/watch?v=wrFXBV4MwvI&ab_channel=CodingWithTim (accessed Nov. 24, 2022).
[2]
"Edit Distance | DP-5," *GeeksforGeeks*, Jul. 06, 2011. https://www.geeksforgeeks.org/edit-distance-dp-5/
[3]
"Trie Data Structure Implementation (LeetCode)," *www.youtube.com*. https://www.youtube.com/watch?v=giiaIofn31A&ab_channel=MichaelMuinos (accessed Nov. 24, 2022).