Report: Text Extraction and Cleaning

Team Member: Ankita, Narayan, Shravan, Shadab, Suraj

Objective:

The objective of this project is to perform text extraction and cleaning using Java programming. The main tasks involve extracting text from input sources and applying specific rules to clean the text, removing special characters, handling punctuation marks, adjusting capitalization, and eliminating unnecessary spaces and lines.

Requirements:

Libraries and files required for the Java project:

- java.io.*
- javax.xml.parsers.DocumentBuilderFactory
- javax.xml.parsers.DocumentBuilder
- org.w3c.dom.Document
- org.w3c.dom.NodeList
- org.w3c.dom.Node
- java.nio.file.Files
- java.io.FileWriter

Cleaning rules for Java project:

1. If any special character other than punctuation marks is found then it will be removed completely from the text.
   An array of characters to be removed fully:
   char[] removable = {'#', '@', '$', '%', '^', '*', '~', '`', '+', '=', '_', '|', '/', '>', '<', '(', ')', '{', '}', '[', ']', '\'}

2. Other punctuation marks will be retained or removed as per the following conditions:
   An array of punctuation marks: char[] punctuations = {'!', ':', ';', '"', '\'', '?', '.', ','}
   a. If a punctuation mark is between two consecutive letters or digits then it is completely removed and letters or digits are concatenated.
   b. If two or more punctuations are consecutive then the last punctuation mark will be considered and the rest are ignored.

c. If after a character there is a punctuation mark followed by a space then it will be retained.

3. Numbers and letters will be retained as per the following rules:
   a. If a digit is between two consecutive letters then it will be completely removed and letters will be concatenated.
   b. If a letter is between two consecutive digits then it will be completely removed and digits will be concatenated.
   c. If space is followed by digit and digit is followed by letter then remove the digit completely.
   d. If letter is followed by digit and digit is followed by punctuation mark then remove digit completely.

4. Capitalization of the First letter of the sentence as per the following rules:
   char[] delimiters = {'.', '?'};
   a. The starting letter of the document will be capitalized.
   b. If delimiters are followed by one space then ignore the space. (implement carefully)
   c. If there is a delimiter before the letter then capitalize the letter.

5. Cleaning of unnecessary spaces and lines will be done as per the following conditions:
   a. Consecutive two or more than two spaces will be terminated to one space.
   b. Consecutive two or more than two lines will be terminated to one line.

Implementation Details:

```java
/*
This code is for extracting the texts from xml and text files and
cleaning the data as per defined rules.

Authors: Ankita, Suraj, Shadab, Shrawan, Narayan
Date: 4 December, 2023
*/

// Importing necessary modules
import java.io.*;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.DocumentBuilder;
import org.w3c.dom.Document;
import org.w3c.dom.NodeList;
import org.w3c.dom.Node;
import java.nio.file.Files;
import java.io.FileWriter;


public class TextCleaner{
    // Main method
    public static void main(String[] args) throws IOException {

        String path = "file.txt";
        File file = new File(path);
        FileWriter fw = new FileWriter("output.txt");   //writing to output file
        String fileextension = extension(path);
        String text = "";
        if (fileextension.equals(".txt")){
            // txt file extraction
            text = readFileAsString(file);
        }
        else if (fileextension.equals(".xml")){
            text = XMLParser(file);    // Extracting text from XML file
        }
        if (text.equals("")){
            System.out.println("There is no text to clean, please enter valid text.");
        }
        else{
            Cleaner obj = new Cleaner();
            String cleanedText = obj.clean(text);
            for (int i = 0; i < cleanedText.length(); i++){
                fw.write(cleanedText.charAt(i));
            }
            fw.close();
        }
    }

    // Identify extension
    static String extension(String path){
        return path.substring(path.indexOf('.'));
    }
```

```java
    // txt extraction
    public static String readFileAsString(File file) throws IOException {
        return new String(Files.readAllBytes(file.toPath()));
    }


    // XML Parser Method
    public static String XMLParser(File file){
        try{
            DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
            DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
            Document doc = dBuilder.parse(file);
            doc.getDocumentElement().normalize();

            NodeList nodeList = doc.getElementsByTagName("*"); // Get all elements

            StringBuilder textContent = new StringBuilder();
            for (int temp = 0; temp < nodeList.getLength(); temp++) {
                Node node = nodeList.item(temp);
                if (node.getNodeType() == Node.ELEMENT_NODE) {
                    textContent.append(node.getTextContent()).append("\n");
                }
            }
            return textContent.toString();
        }
        catch (Exception e) {
            e.printStackTrace();
        }
        return "";
    }
}


// Other cleaner class
class Cleaner{
    char[] removable = {'#', '@', '$', '%', '^', '*', '~', '`', '+', '=', '_', '|', '/', '>', '<', '(', ')', '{', '}', '[', ']', '&'};
    char[] punctuations = {'!', ':', ';', '?', '.', ','};
    char[] spaces = {'\n', ' '};
    char[] delimenaters = {'.', '?'};

    // Methods to clean the data
    public String clean(String text){
        String t5 = text;
        for(int i = 0; i < 3; i++){
            String t1 = removeSpecChar(t5);
            String t2 = updatePunct(t1);
            String t3 = removeDigits(t2);
            String t4 = capitalize(t3);
            t5 = removeSpaces(t4);
        }
        return t5;
    }
```

```java
    public String removeSpecChar(String text) {
        String updatedText = "";
        for (int i=0; i<=text.length()-1; i++){
            char c = text.charAt(i);
            boolean found = isCharInArray(c, removable);
            if (found){
                continue;
            }
            else{
                updatedText += c;
            }
        }
        return updatedText;
    }


    // Method to remove unncessary punctuations
    public String updatePunct(String text){
        String updatedText = "";
        for (int i=0; i<=text.length()-1; i++){
            char c = text.charAt(i);
            boolean found = isCharInArray(c, punctuations);
            if (found){
                if (i==0){
                    continue;
                }
                else if(i==text.length()-1){
                    updatedText += c;
                }
                else{
                    char prev = text.charAt(i-1);
                    char next = text.charAt(i+1);
                    boolean p = isCharInArray(next, punctuations);
                    if (Character.isLetter(prev) && Character.isLetter(next)){
                        continue;
                    }
                    else if (Character.isDigit(prev) && Character.isDigit(next)){
                        continue;
                    }
                    else if (Character.isDigit(prev) && Character.isLetter(next)){
                        continue;
                    }
                    else if (Character.isLetter(prev) && Character.isDigit(next)){
                        continue;
                    }
                    else if (p){
                        continue;
                    }
                    else if (prev == ' ' && next == ' '){
                        continue;
                    }
                    else if((Character.isLetter(prev) || Character.isDigit(prev)) && next == ' '){
                        updatedText += c;
```

```
                    }
                    // in doubt unnecessary
                    else if(!(Character.isLetter(prev)) && Character.isLetter(next)){
                        if(c==','){
                            updatedText = updatedText + c + " ";}
                    }
                    else{
                        updatedText += c;
                    }
                }
            }
            else{
                updatedText += c;
            }
        }
        return updatedText;
    }

    // Combining spaces.
    public String removeDigits(String text){
        String updatedText = "";
        int len = text.length();
        for (int i=0; i<=text.length()-1; i++){
            char c = text.charAt(i);
            if (i == 0 && Character.isDigit(c) && Character.isLetter(text.charAt(i + 1))){
                continue;
            }
            else if(i == 0){
                updatedText +=c;
            }
            else if (i == len - 1 && Character.isDigit(c) && Character.isLetter(text.charAt(i - 1))){
                continue;
            }
            else if(i == len -1){
                updatedText +=c;
            }
            else{
                char prev = text.charAt(i-1);
                char next = text.charAt(i+1);
                boolean p = isCharInArray(next, punctuations);
                if (Character.isDigit(c) && Character.isLetter(prev) && Character.isLetter(next)){
                    continue;
                }
                else if(Character.isLetter(c) && Character.isDigit(prev) && Character.isDigit(next)){
                    continue;
                }
                else if (prev == ' ' && Character.isDigit(c) && Character.isLetter(next)){
                    continue;
                }
                else if (next == ' ' && Character.isDigit(c) && Character.isLetter(prev)){
                    continue;
                }
                else if (Character.isLetter(prev) && Character.isDigit(c) && p){
```

```
                    continue;
                }
                else{
                    updatedText += c;
                }
            }
        }
        return updatedText;
    }

    public String capitalize(String text){
        String updatedText = "";
        int len = text.length();
        for (int i=0; i<=text.length()-1; i++){
            char c = text.charAt(i);
            if (i==0 && Character.isLetter(c)){
                updatedText += Character.toUpperCase(c);
            }else if (i == len -1 && Character.isLetter(c)){
                updatedText = updatedText + c + ".";
            }
            else if (i == 0){
                updatedText += c;
            }
            else{
                char prev = text.charAt(i-1);
                boolean p = isCharInArray(prev, delimenaters);
                if (c == ' ' && p){
                    continue;
                }
                else if ( i >= 2 && isCharInArray(text.charAt(i - 2), delimenaters) && prev == ' ' && Character.isLetter(c)){
                    updatedText = updatedText + " " + Character.toUpperCase(c);
                }
                else{
                    updatedText += c;
                }
            }
        }
        return updatedText;
    }

    public String removeSpaces(String text){
        String updatedText = "";
        int len = text.length();
        for (int i=0; i<=text.length()-1; i++){
            char c = text.charAt(i);
            boolean p = isCharInArray(c, spaces);
            if (i == 0 && p){
                continue;
            }else if(i == len - 1 && p){
                continue;
            }
            else if(i == len -1){
                updatedText +=c;
```

```
265                 updatedText +=c;
266             }
267             else{
268                 char next = text.charAt(i+1);
269                 if (c == ' ' && next == ' '){
270                     continue;
271                 }else if(c == '\n' && next == '\n'){
272                     continue;
273                 }else{
274                     updatedText += c;
275                 }
276             }
277         }
278         return updatedText;
279     }
280 
281     // ------------ helper method----------
282     public boolean isCharInArray(char c, char[] arr){
283             boolean found = false;
284             for (char x : arr) {
285                 if (x == c) {
286                     found = true;
287                     break;
288                 }
289             }
290             return found;
291     }
292 }
```

Result:

Input→ “hey, how are you? buddy. are you do2ne w8ith pro&j@ct. tell m#e as soon a5 s I need to make ,, .. report*? thank you.”
Output→ “Hey, how are youbuddyare you done with projct. Tell me as soon a s I need to
make report? Thank you.”

Conclusion:

The implemented text extraction and cleaning process successfully achieved the defined objectives. However, there were some inaccuracies observed in the cleaning process, possibly due to certain exceptional cases not being accounted for. Further refinement and handling of edge cases might be required to enhance the accuracy and robustness of the text-cleaning algorithm.

The project successfully fulfills the objectives of text extraction and cleaning in Java, but further refinement is recommended to address the observed inaccuracies.
In the future, our team would be happy to improve the cleaning
part.