Alexander Mosiychuk
Thursday: Cmpt-220
Project Two Milestone

ABSTRACT

This paper describes the implementation of a web crawler to assess the offensiveness of a website. Specifically, the paper provides a description of the program's primary components, and a summary of how the methods interact. Moreover, the paper explains the potential sources for user error.

INTRODUCTION

To many individuals, the internet is a cornerstone of their everyday life, providing access to previously unattainable resources and information. However, alongside the idealistic access to almost boundless information and social media, the internet also hosts sites many would consider offensive, such as sites dedicated to potentially racist and otherwise crude humor. Furthermore, individuals may be unable to perform a preliminary evaluation of the site with any degree of accuracy. My project aims to fix this issue, and attempts to provide a way to perform a preliminary evaluation of an inputted web address. Specifically, for this assignment, I have worked to develop a WebCrawler program that determines the risk of a website being offensive.

Detailed System Description

As previously mentioned, the primary goal of my webcrawler is to produce a risk assessment of a website's offensiveness. To accomplish my program consists of three primary components. Specifically, an array of input keywords, a counter based algorithm which compares the keywords to the contents of a desired website, and a series of if statements to output the appropriate statements.

The primary data field that lies at the core of my program is the single dimensional array keywords, which as the name suggests stores user inputted string keywords. Specifically, the user initially inputs the intended amount of keywords, and the system then will prompt for the corresponding amount. Afterwards, a selection sort algorithm, namely *stringSort*, for the string is invoked to sort the array, allowing for a binary search to be used in the comparison component of the program. Afterwards, the system asks for the URL address of the website the webcrawler will assess.

To assess the risk of offensivity, my webcrawler implements a counter based system. In particular, the program counts the matches of the keyword array elements with the website's contents. To accomplish the task, the method read, which returns an integer value, is invoked within the *crawler* method, using a binary search to scan the contents of the array. The number of matches is then returned to the crawler method which increments a counter value that then is passed to the main method. Lastly, the counter is passed through a series of if statements that output an appropriate risk assessment that is dependent on the value of counter.

UML Diagram

Proj2
length:int keyword: String[] counter: int
stringSort(list: String[]): void read(inpUrl: String, key: String[]): int crawler(startingUrl: String, key: String[]): int GetSubURLs(urlString: String): ArrayList

User Manual

Overall the webcrawler is relatively simple to operate, requiring the user to input an integer value for the array length, the keywords for comparison, and the URL of the site the user wants to evaluate. In order to proceed with the program the user must input the user must input an integer value. If not the system will simply reproduce the prompt for an appropriate length value. Moreover, the user must input a valid URL address otherwise the crawler will be unable to operate and implement its algorithms. Otherwise, there do not exist many points of user induced runtime errors.

Conclusion

In summation, the program utilizes a counter based system to assess a website's offensiveness. The user has a degree of control over the program, allowing the program to be adaptive to a wide variety of offensive subject matter without the need for an excessively large keyword array. Moreover, the use of binary search improves the speed of the *read* method, improving the efficiency of the program.