Name: CyberSecurityWebScraper Program Documentation
Author: Nathanael Fuller
Author: Bryan Kline
Class: CS445
Date: May 3, 2018

Software Package Contents:

CyberSecurityWebScraper Program Documentation.pdf
readme

`CyberSecurityWebScraper.py`
`systemInitialization.py`
`WebScraper.py`
`Filter.py`
`TemporaryDatabase.py`
`Clear.py`

`makefile`
`main.cpp`
`Dictionary.h`
`Dictionary.cpp`
`Accumulator.h`
`Accumulator.cpp`

`dictionary.txt`
`exclusionDictionary.txt`

Program Overview

CyberSecurityWebScraper is a terminal-based tool for Linux developed for cyber security professionals in order to automate the process of checking for alerts, notifications, and news related to cyber security vulnerabilities, threats, and attacks. The user builds and runs the program, the content of 16 cyber security related websites are scraped, their contents are checked against a dictionary of cyber security terms, and any matches are pushed into a SQLite database for subsequent perusal. The content added to the database includes security related headlines, their URLs, time stamps corresponding to the time the site was scraped, as well as a score related to the number of matches a headline had against the cyber security dictionary. Additionally, the program checks the database contents in order to determine new words that can be added to the security dictionary by checking the frequency of the appearance of uncommon and non-grammatical words as well as their proximity to dictionary key terms in the headlines.

The program consists of four major phases, the first being the initialization phase wherein the SQLite database is created and the `updateDictionary` program is compiled. Before these steps are taken however the current working directory is inspected in order to determine if these steps have already been taken from a previous build and execution of the program, in which case either initialization step is not taken. The initialization phase is executed with the `systemInitialization.py` script which is written in Python. The next phase of of the program is the scraping and filtering of website

content, the population of the database with the filtered content, and the creation of a temporary file containing all content from the database, include from previous passes of the system, as a raw file for use by the next phase of the program. The web scraping phase of the program is accomplished by the `WebScraper, Filter,` and `TemporaryDatabase` classes, which are all written in Python. The next phase of the program involves stepping through all scraped content present in the database, inspecting the non-dictionary, uncommon, and non-grammatical terms present in the content, determining if they are relevant and if so adding them to the dictionary. The `updateDictionary` program consists of a `makefile` and `main.cpp, Dictionary.h, Dictionary.cpp, Accumulator.h`, and `Accumulator.h` files, all of which are written in C++. The final phase of the program involves deleting temporary files created by the program, which is implemented in the `Clear` class and written in Python. All programming units described above are executed in the program main driver, `CyberSecurityWebScraper.py`, which is written in Python.

Program Details

The main driver of the program, `CyberSecurityWebScraper.py`, executes the the `systemInititalization.py` script which makes a system call to determine if the database has been created by inspecting the contents of the current working directory, if not then the Python SQLite API is used to create a database called `securityHeadlines.db`. The script also inspects the directory for object files of the appropriate name to determine if the C++ program has been compiled, if not then it builds the program with a system call. Next, `CyberSecurityWebScraper.py`, creates a `WebScraper` object and uses it to scrape 16 cyber security websites. The `WebScraper` class creates a CSV file and then calls 16 class methods which each open a connection with a website through the Python web scraping library BeautifulSoup. Once the connection is open, the content from the site is pulled into the program and parsed according to how the HTML on that particular site is organized, and headlines and URLs are pulled into the program and formatted, a time stamp is generated, and they are written to the CSV file.

Once the CSV file containing the raw content from all the websites is generated, a `Filter` object is created and used to filter the content in the main driver. The `Filter` class creates several threads in order to parallelize the work it does as much of it is not depended upon other steps. These threads read in the CSV file, read in the dictionary, look for matches between the raw content and the dictionary, assign a score to each based on how many matches are found, and inspect the database to determine the primary key of the last pass of the program. After those threads have terminated, the matches are inspected in order to remove duplicates, they are sorted on the basis of their scores, and then those that remain are assigned primary keys based on the last primary key in the database, and they are then written to the database. Once this is complete the main driver creates a `TemporaryDatabase` class object which reads in the database and writes it out to a temporary file to be used by the C++ program which updates the dictionary.

The main driver then issues a system call to execute the C++ program `updateDictionary` which executes its main driver. In the `updateDictionary` driver a `Dictionary` object is created which reads in the files containing the database content, the dictionary of security terms, and an exclusion dictionary, the latter containing common and grammatical words which should not be used to update the dictionary if they are found to be in the database content. The `Dictionary` object then parses the database content by breaking the content into individual whitespace and punctuation delimited strings and then removes common and grammatical terms from the content by comparing it with the exclusion

dictionary. The `Dictionary` class has an `Accumulator` class object as a data member which it then uses to build an accumulator for the scores for each term. The parsed and pared down database content is then passed to the `Accumulator` class which creates a data structure of `KeyTerm` structs which hold a string for the term as well as an int for the score. The accumulator maintained by the `Accumulator` class holds all terms where each row is a scraped headline and each column is a term, the dictionary is iterated through and all dictionary terms are assigned a score which marks them as already in the dictionary. The `Accumulator` class then calculates the scores of the the terms in the `Accumulator` by looking for dictionary key terms nearby and by enumerating the frequency of the terms in other headlines. If a term is near a dictionary key term in the headline or if it occurs elsewhere in another headline then its score is incremented. All terms in the accumulator have a threshold applied to them which is some proportion of the highest score in the accumulator, 0.3 being used here. The terms with scores below threshold are removed and those that remain in the accumulator are then added to another data structure from which duplicates are removed, the terms are added to the dictionary, and the `updateDictionary` program terminates.

Finally, after the dictionary is updated, the program main driver creates a `Clear` class object which cleans up the current working directory by deleting all temporary files created for the program during its execution. The database is now ready for the cyber security professional to consult in order investigate and follow up on security vulnerabilities, threats, and attacks.

Programming Units

The following is organized by file, class, and class method, wherein each method is described in detail.

CyberSecurityWebScraper.py

Class: None

systemInitialization.py

Class: None

WebScraper.py

Class: WebScraper

Methods:

Name:	__init__
Description:	WebScraper class default constructor which opens the CSV file which the raw scraped content is written to
Parameters:	None
Return:	None

Name:	parseText

Description: WebScraper class method which takes in a string and removes commas and semicolons
Parameters: Takes in a string which has commas and semicolons removed from it
Return: Returns the input string with commas and semicolons removed

Name: removeWhitespace
Description: WebScraper class method which takes in a string and removes whitespace from it, including tabs, newlines, and carriage returns
Parameters: Takes in a string to have whitespace removed from it
Return: Returns the input string with whitespace removed

Name: getTimeDate
Description: WebScraper class method which gets the time and date and makes a time stamp from it
Parameters: None
Return: Returns a time stamp as a string

Name: pushRawHeadline
Description: WebScraper class method which takes in strings which are the headline, URL, and time stamp and writes them to the CSV file which holds the raw scraped content
Parameters: Takes in strings corresponding to the headline, URL, and time stamp to be written to the CSV file
Return: None

Name: crawlUSGovAnnouncements
Description: WebScraper class method which scrapes https://www.us-cert.gov/announcements
Parameters: None
Return: None

Name: crawlUSGovAlerts
Description: WebScraper class method which scrapes https://www.us-cert.gov/ncas/alerts
Parameters: None
Return: None

Name: crawlUKGovAlerts
Description: WebScraper class method which scrapes https://www.ncsc.gov.uk/index/alerts-and-advisories
Parameters: None

```
Return:         None

Name:           crawlCanadaAlerts
Description:    WebScraper class method which scrapes
                https://www.publicsafety.gc.ca/cnt/rsrcs/cybr-
                ctr/index-en.aspx#al
Parameters:     None
Return:         None

Name:           crawlReuters
Description:    WebScraper class method which scrapes
                https://www.reuters.com/news/archive/
                cybersecurity
Parameters:     None
Return:         None

Name:           crawlSecurityAffairs
Description:    WebScraper class method which scrapes
                http://securityaffairs.co/wordpress/
Parameters:     None
Return:         None

Name:           crawlHackerNews
Description:    WebScraper class method which scrapes
                https://thehackernews.com
Parameters:     None
Return:         None

Name:           crawlThreatNews
Description:    WebScraper class method which scrapes
                https://threatpost.com/blog/
Parameters:     None
Return:         None

Name:           crawlComputerWorldSecurity
Description:    WebScraper class method which scrapes
                https://www.computerworld.com/category/security/
Parameters:     None
Return:         None

Name:           crawlTechNewsSecurity
Description:    WebScraper class method which scrapes
                https://www.technewsworld.com/perl/section/tech-
                security
Parameters:     None
Return:         None
```

```
Name:           crawlDarkReading
Description:    WebScraper class method which scrapes
                https://www.darkreading.com/attacks-breaches.asp
Parameters:     None
Return:         None

Name:           crawlHackerCombat
Description:    WebScraper class method which scrapes
                https://hackercombat.com/
Parameters:     None
Return:         None

Name:           crawlSCMagazine
Description:    WebScraper class method which scrapes
                https://www.scmagazine.com/news/section/6343/
Parameters:     None
Return:         None

Name:           crawlSecurityMagazine
Description:    WebScraper class method which scrapes
                https://www.securitymagazine.com/topics/2236-
                cyber-security-news
Parameters:     None
Return:         None

Name:           crawlSecurityWeek
Description:    WebScraper class method which scrapes
                https://www.securityweek.com/cybercrime
Parameters:     None
Return:         None

Name:           crawlReddit
Description:    WebScraper class method which scrapes
                http://www.reddit.com/r/cybersecurity
Parameters:     None
Return:         None

Name:           run
Description:    WebScraper class method which acts as the primary
                interface to the class which calls all class
                methods to scrape 16 different cyber security
                resource sites and writes the scraped headlines,
                URLs, and a time stamp for each one, to a CSV
                file
Parameters:     None
Return:         None
```

Filter.py

Class: Headline

Methods:

Name:	__init__
Description:	Headline class parameterized constructor which builds a Headline object from the parameters passed in
Parameters:	Takes in ints corresponding to the primary key and score and strings corresponding to the headline, URL, and time stamp
Return:	None

Name:	getPrimaryKey
Description:	Headline class method which returns the primary key
Parameters:	None
Return:	Returns the primary key

Name:	getHeadline
Description:	Headline class method which returns the headline
Parameters:	None
Return:	Returns the headline

Name:	getURL
Description:	Headline class method which returns the URL
Parameters:	None
Return:	Returns the URL

Name:	getTimeDate
Description:	Headline class method which returns the time stamp
Parameters:	None
Return:	Returns the time stamp

Name:	getScore
Description:	Headline class method which returns the score
Parameters:	None
Return:	Returns the score

Name:	setPrimaryKey
Description:	Headline class method which sets the primary key

Parameters: Takes in a primary key which will be assinged to the corresponding class data member
Return: None

Name: printHeadlineObject
Description: Headline class method which prints the class data members
Parameters: None
Return: None

Class: Filter

Methods:

Name: __init__
Description: Filter class default constructor which opens the CSV and dictionary files
Parameters: None
Return: None

Name: databaseCommit
Description: Filter class method which takes in a Headline object, gets its data members, opens a connection with the SQLite databsse and pushes the data from the object into the database
Parameters: Takes in a Headline object and a boo corresponding to whether or not the class should print the object for diagnostic purposes
Return: None

Name: readInDatabase
Description: Filter class method which opens a connection with the SQLite database, reads in all the data from it from previous passes into a data structure if there are any entries in the database
Parameters: None
Return: None

Name: getLastPrimaryKey
Description: Filter class method which opens a connection with the SQLite database and gets the primary key from the last entry in it if there are any so as to start the data it pushes into the database on the current pass from the correct primary key
Parameters: None
Return: None

Name: readFileContents
Description: Filter class method which opens the CSV file holding the raw data from the site scraped, reads the contents into a class data structure and makes each item lower case
Parameters: None
Return: None

Name: readDictionary
Description: Filter class method which opens the dictionary file and reads in its contents into a class data structure
Parameters: None
Return: None

Name: findMatches
Description: Filter class method which moves through the CSV file contents in the local data structure and checks each item against the contents of the dictionary data structure and if there is a match then that item is added to a class data structure which keeps track of all scraped content with terms present that are found in the dictionary, also scores each item based on the number of matches it has against the dictionary
Parameters: None
Return: None

Name: filterMatches
Description: Filter class method which moves through all the matches found between the database contents and the dictionary and removes all duplicates to prevent repeated content in the database
Parameters: None
Return: None

Name: removePunctuation
Description: Filter class method which takes in a string and returns the string with all punctuation removed
Parameters: Takes in a string which is to have its punctuation removed
Return: Returns the input string with all punctuation removed

Name: matchScore

Description: Filter class method which determines the total score for a headline based on the number of matches it has in the dictionary
Parameters: Takes in a headline and a dictionary term, splits the headline by white space to break it into words, then steps through the words and checks them against the dictionary term to determine the score for that headline
Return: Returns the score for the headline passed in as a parameter

Name: assignKeys
Description: Filter class method which determines the last primary key in the database and moves through the sorted matches and assigns each one a primary key
Parameters: None
Return: None

Name: orderMatches
Description: Filter class method which moves through the class data structure holding all the matches of headlines with the dictionary and sorts them by the score they received corresponding to the number of matches found
Parameters: None
Return: None

Name: commitMatches
Description: Filter class method which steps through the sorted matches and adds each one to the database
Parameters: None
Return: None

Name: run
Description: Filter class method which acts as the primary interface for the class which opens separate threads to read in all files into local data structures and to find matches between the scraped content and the dictionary, gets the next primary key to use, filters and sorts the matches, and pushes the matches into the database
Parameters: None

Return: None

Name: printMatches
Description: Filter class method which prints all matches to the screen for diagnostic purposes
Parameters: None
Return: None

TemporaryDatabase.py

Class: TemporaryDatabase

Methods:

Name: __init__
Description: TemporaryDatabase class default constructor which creates the temporary database file to write to
Parameters: None
Return: None

Name: readInDatabase
Description: TemporaryDatabase class method which opens a connection with the SQLite database and reads its contents into a local data structure
Parameters: None
Return: None

Name: writeOutDatabase
Description: TemporaryDatabase class method which moves through the class data structure which holds the contents of the database and writes them to a file
Parameters: None
Return: None

Name: run
Description: TemporaryDatabase class method which acts as the main interface through which the class methods are accessed which reads in the database and writes its contents to a raw file
Parameters: None
Return: None

Clear.py

Class: Clear

Methods:

Name: __init__
Description: Clear class parameterized constructor
Parameters: Takes in bools corresponding to whether the database, temporary database, and CSV file should be deleted upon termination of the program
Return: None

Name: cleanUp
Description: Clear class method which checks the class data members to determine if the database, temporary database, and CSV file should be deleted upon termination of the program
Parameters: None
Return: None

Name: deleteDatabase
Description: Clear class method which deletes the database from the current working directory
Parameters: None
Return: None

Name: deleteTempDatabase
Description: Clear class method which deletes the temporary database from the current working directory
Parameters: None
Return: None

Name: deleteCSV
Description: Clear class method which deletes the CSV file from the current working directory
Parameters: None
Return: None

makefile

Class: None

main.cpp

Class: None

Dictionary.h/Dictionary.cpp

Class: Dictionary

Methods:

Name: Dictionary
Description: Dictionary class parameterized constructor
Parameters: None
Return: None

Name: ~Dictionary
Description: Dictionary class destructor
Parameters: None
Return: None

Name: readInDatabase
Description: Dictionary class method which opens and reads in the database into a local data structure
Parameters: None
Return: Void

Name: readInDictionary
Description: Dictionary class method which opens and reads in the dictionary into a local data structure
Parameters: None
Return: Void

Name: readInExclusionDictionary
Description: Dictionary class method which opens and reads in the exclusion dictionary into a local data structure
Parameters: None
Return: Void

Name: run
Description: Dictionary class method which acts as the primary interface through which the program is accessed, it calls the necessary Dictionary and Accumulator class methods in order to read in the data from files, to parse and filter the data, to push the data into the accumulator data structure and the then assign scores to the terms, and finally to update the dictionary
Parameters: None
Return: Void

Name: parseDatabase
Description: Dictionary class method which moves through the content scraped as several long strings and breaks them down into words and removes whitespace and punctuation and pushes them into the data structure which holds the parsed content from the database
Parameters: None
Return: Void

Name: excludeTerms
Description: Dictionary class method which steps through the parsed database data structure and removes terms if they are found in the exclusion dictionary
Parameters: None
Return: Void

Name: searchExclusionDictionary
Description: Dictionary class method which takes in a term as a parameter and steps through the exclusion dictionary to check whether it is a term which should be excluded
Parameters: Takes in a string which search for in the exclusion dictionary
Return: Returns a bool corresponding to whether or not the input string is found in the exclusion dictionary

Name: diagnosticPrint
Description: Dictionary class method which prints to the screen all class data structures for diagnostic purposes
Parameters: None
Return: Void

Accumulator.h/Accumulator.cpp

Class: Accumulator

Methods:

Name: Accumulator
Description: Accumulator class default constructor
Parameters: None
Return: None

Name: buildAccumulator
Description: Accumulator class method which takes in the parsedDatabase and the dictionary and builds a a data structure containing KeyTerm structs which hold all the terms in the scraped content, if content in the data structure is found in the dictionary then it is assigned a score signifying it is a dictionary term, otherwise it gets initialized to a score of zero
Parameters: Takes in a two dimensional vector of string which holds the parsed content from the sites scraped and a vector of strings which is the dictionary
Return: Void

Name: accumulate
Description: Accumulator class method which moves through the data structure containing KeyTerm structs holding the parsed content from the sites scraped and accumulates each item's score by determining whether there are dictionary terms proximal to it and whether the item is found elsewhere in the structure
Parameters: None
Return: Void

Name: frequencyScore
Description: Accumulator class method which moves through the data structure containing the scraped content and enumerates a score based on the frequency with which that item has occurred
Parameters: None
Return: None

Name: thresholdAccumulator
Description: Accumulator class method which moves through the accumulator and removes all values which are some threshold proportion of the highest scored item in the accumulator
Parameters: None
Return: Void

Name: termUpdated
Description: Accumulator class method which takes in a candidate term for addition to the dictionary and moves through the terms to be added to the

	dictionary and checks if the candidate is already to be added so as to avoid adding duplicates
Parameters:	Takes in a string which is a candidate addition to the dictionary
Return:	Returns a bool corresponding to whether or not the input string has already been added to the structure which holds the new additions to the dictionary

Name:	updateDictionary
Description:	Accumulator class method which writes the terms to be added to the dictionary to the dictionary file
Parameters:	None
Return:	Void

Name:	dictionaryCheck
Description:	Accumulator class method which takes in the dictionary and a string and returns a bool signaling whether or not the string is found so that the items in the accumulator can be assigned proper scores
Parameters:	Takes in a vector of strings which is the dictionary and an input string which is being sought in the dictionary
Return:	Returns a bool corresponding to whether or not the input string is found in the dictionary

Name:	toUpper
Description:	Accumulator class method which takes in a string and converts all lower case chars in the string to upper case so that new dictionary entries are upper case for tracking purposes
Parameters:	Takes in a string to be converted to upper case
Return:	Returns a string which is the input string in upper case

Name:	diagnosticPrint
Description:	Accumulator class method which prints the screen the various data members and structures in the class for diagnostic purposes
Parameters:	None
Return:	Void