News Event Website Capstone Report

by

Joseph Acanfora, Briana Crabb, and Jeff Morris

Client: Mohammed Faraq

Professor
Dr. Edward A. Fox
College of Engineering Department of Computer Science

May 2015
Abstract

Dr. Edward A. Fox
College of Engineering Department of Computer Science

News Event Website

by Joseph Acanfora, Briana Crabb, and Jeff Morris

This report gives a detailed overview of the archiving services developed by the News Event Detection Group in the Virginia Tech Multimedia Capstone. Our group developed a framework to help facilitate the collection of large amount of news data for the project sponsor, Mohamed Magdy Gharib Farag.

The paper will begin by giving the reader an overview of the project background and the importance of our final product. Following the overview are several sections on the design stages of our product. These sections include our system overview and architecture. Then we discuss the data, component and interface design procedures that took place. The paper then covers the developmental stages of the project. We discuss our implementation and prototyping plan as well as the testing procedures and results we used on the product.

Next the paper will instruct how the user is to conduct a query through our web interface. Following the user manual is a developer’s manual documenting in full detail the tools and procedures used to create our product. These tools include PHP, Apache, and Flask. This developer manual will assist any individual looking to analyze the product or make future modifications.

Finally, the paper addresses the lessons the group has learned while working on this project. This includes some of the challenges that were faced, solutions to those challenges, as well as future changes that could be made.
# Contents

Abstract 1

List of Figures 5

1 Introduction 6
   1.1 Overview 6
   1.2 Project Background 6
   1.3 Purpose 6
      1.3.1 Scope 7

2 System Overview 8
   2.0.2 Software Usage 8

3 System Architecture 9
   3.1 Architectural Design 9
   3.2 Decomposition Description 11
   3.3 Design Rationale 11

4 Data Design 12
   4.1 Data Description 12

5 Component Design 13
   5.1 Web Interface 13
   5.2 Starting Twitter Script 13
   5.3 Start the web crawling Script 13
   5.4 Final Email Notification 14

6 Human Interface Design 15
   6.1 Overview of User Interface 15
   6.2 Screen Images 15
   6.3 Screen Objects and Actions 15

7 Implementation Plan 18
   7.1 Team Roles 18
      7.1.1 Joe Acanfora 18
      7.1.2 Jeff Morris 18
      7.1.3 Briana Crabb 18
   7.2 Major Milestones 19
12 Lessons Learned

12.1 Time line & Schedule ........................................ 35
12.2 Problems and Solutions ....................................... 35
    12.2.1 Cloud9 and Email Complications ...................... 35
    12.2.2 calling scripts non locally ................................ 36
    12.2.3 deciding what backend structure/ framework would be easiest to implement ..................................... 36
    12.2.4 Executing Scripts from Flask ......................... 36
12.3 Future Work .................................................. 36

13 Acknowledgments ................................................. 37

A Appendix .................................................................. 38
    A.1 Scripts .......................................................... 38
List of Figures

3.1 System Design .................................................. 10
4.1 Twitter Database Wire Diagram .................................. 12
6.1 Web Design Screenshot ............................................. 16
6.2 TwapperKeeper Results Page ........................................ 16
6.3 Twitter Login Page .................................................. 16
6.4 ProjFocusedCrawler Results Page ................................. 17
6.5 Screen Objects and Actions Table ................................. 17
7.1 Gant Chart weeks 1 through 8 ................................. 20
7.2 Gant Chart weeks 9 through end ................................. 21
8.1 YourTwapperKeeper Screen shot .................................... 23
8.2 Curl Example Screen shot .......................................... 24
8.3 Email Screen shot ................................................... 25
9.1 Web Front End Screenshot ........................................... 27
9.2 YourTwapperKeeper Output ........................................ 28
9.3 YourTwapperKeeper Detailed Output ............................... 28
Chapter 1

Introduction

1.1 Overview

The overall objective of this project is to build a website front end to help the client automate the web archiving process for their big data doctorate research. The user of our product should be able to type in keywords for a news event, hit submit, and our product will call the necessary scripts to start the process of crawling Twitter and news websites to archive data pertaining to a specific news event. The group will be responsible for creating a web interface for a user to input the necessary key words and start the archiving process. We will create two scripts: one to start the YouTwapperKeeper script to begin the twitter crawl, and the other script will start the news scraping script. After the process has been initiated by our scripts we will notify the user via email when the job is complete.

1.2 Project Background

The digital archive database is a Virginia Tech research initiative to explore the use of natural language processing in conjunction with Hadoop and other data analysis tools. The database’s purpose is to collect large amounts of articles about natural disasters and other news events, and to use use software tools to quickly summarize those events.

1.3 Purpose

This document outlines the architecture and system design of the news event reporting project. It will highlight the features being used to automate the scripting process. The
primary audience of the documentation will be the graduate student sponsor, Professor Fox, and any potential future uses of the website. This document may be consulted if any future changes or modifications need to be made. This document will allow the user to quickly understand how to replicate and use this system to capture news event data.

1.3.1 Scope

The goal of this project is to help streamline the news event archiving process. We are automating the scripts that collect the news articles and complies them in an archive. This projects benefits anyone interested in archiving news events. The end user will no longer have to sift through and type in or select features multiple times. They will be able to type in the necessary info once and let the scripts run, thus saving the user time and reducing the likelihood of input errors.
Chapter 2

System Overview

The project performs the necessary functions to automate the scripts to start archiving news events. We have broken the project into four phases: web design and form entry, initiation of twitter crawling script, and initiation of web archive script, and finally, email notification back to the user. The four phases can be seen in more detail in figure 3.1. Phase 1 in the top block named web interface. This interface is described in depth in chapter 6, Human Interface Design. Phase 2 is the backend for the website that takes in all the input from the forms and launches the later 2 phases. Phase three involves the launching of the two scripts screen in the ellipses of the same figure. The final phase is the email reporting service, which will communicate to the user the final result of their query.

2.0.2 Software Usage

Several languages are used to implement this project due to the varying of technologies required to initiate the scripts and handle form input. The web interface is implemented in HTML and CSS using the Twitter Bootstrap Framework. The backend of the web interface uses PHP for the form input and launching YourTwapperKeeper. Python and flask are used to initiate, and display the output from, the ProjFocusedCrawler script.
Chapter 3

System Architecture

3.1 Architectural Design

The project is based around a PHP backend web server. The input is passed into this system from the html form and then the server then manages the running of each Python script that will generate the necessary output. After the output is gathered the PHP server will be able to send an email to the original runner of the script. The ProjFocusedCrawler script is managed by a separate flask server. This server receives HTTP get requests from the PHP server instructing it to initiate the script and display the output to the users browser.
Figure 3.1: System Design
3.2 Decomposition Description

The Web Interface is described in more detail in chapter 6. The purpose of this interface is to allow the user to input information into a form, which will be communicated to the PHP backend. The backend collects and processes all the information that comes from the web front end and launches the YourTwapperKeeper script. The PHP server allows initiates a http/get request to the Flask backend. The python flask server then initiates the ProjFocusedCrawler script. These scripts perform the data collection and storage, the links to the source code of these scripts are listed in Appendix A. The email reporting service sends a summary of the results generated from the two scripts to the user upon competition of the data collection.

3.3 Design Rationale

We chose to have two backend servers in two different languages because each web service is tailored specifically to launching, and handling the output from, each of their respective data crawling and collection scripts. YourTwapperKeeper is a PHP script, making it more easily fit with a PHP backend. ProjFocusedCrawler is a python script, meaning that it can be more easily modified and handled from a python backend. Flask was chosen over other python backend options like bottle and django because of its extensibility, rapid development cycle and flexibility.
Chapter 4

Data Design

4.1 Data Description

There are two unique types of data that will be the outputs from the scripts that we will need to store. First, the data coming from the wayback machine are html files that will be stored using a file tree structure. Second is the twitter data from 'YourTwapperKeeper,' which we will store using a database.

<table>
<thead>
<tr>
<th>Tweet Data</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>tweet_id</td>
<td>int</td>
<td>PK</td>
</tr>
<tr>
<td>time</td>
<td>timestamp</td>
<td>N</td>
</tr>
<tr>
<td>author</td>
<td>char(50)</td>
<td>N</td>
</tr>
<tr>
<td>location</td>
<td>char(150)</td>
<td>N</td>
</tr>
<tr>
<td>tweet</td>
<td>varchar(145)</td>
<td>N</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hashtags</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>int</td>
<td>PK</td>
</tr>
<tr>
<td>tweet_id</td>
<td>int</td>
<td>N</td>
</tr>
<tr>
<td>hashtag</td>
<td>varchar(145)</td>
<td>N</td>
</tr>
</tbody>
</table>

Figure 4.1: Twitter Database Wire Diagram
Chapter 5

Component Design

5.1 Web Interface

The web interface is designed using Bootstrap to create a professional and streamline page for user input. There are areas for the user to insert necessary keywords to describe a new event. These include the name of the event, type of event, the year, urls associated with the event and a file containing a list of seed urls that are used as initial targets for the ProjFocusedCrawler script. The user will click a submit button and then be notified via email when the data collection has been completed. In addition to the front end there is a backend that parses the users input and makes sure the correct variables are passed into each script. We are using PHP for this.

5.2 Starting Twitter Script

To automate the YouTwrapperScript, we pull the data the user has input into our front end. This includes the event name, type, any associated hashtags, and keywords. The scraping is done with the provided script and will run in the background of a server collecting data for a period of time. This will be later collected by whomever originally ran the script.

5.3 Start the web crawling Script

The automation of the web crawling script requires the PHP server to communicate with the python Flask server. The seed file from the front end is passed to the Flask
server which then initializes the focus crawler script. The resulting list of URLs is then displayed on the website.

5.4 Final Email Notification

Upon completion of the two crawling scripts a PHP script is called to send an email to the user notifying them of the completion of the process. The users email address will be received from the input via the front end.
Chapter 6

Human Interface Design

6.1 Overview of User Interface

The user will only interact with the main webpage of the screen. They will type in the necessary keywords into the webpage. If they are missing keywords and error message will occur. This is to ensure the user enters all the information that is needed to automate our scripts. The user will hit a submit button to start the process. Then the user is free to leave the webpage. Once the process is complete the user will receive an email notification, provided he or she entered a valid email address into our front end webpage.

6.2 Screen Images

The figures in this section show the current state and design of the application. Figure 6.1 shows the entry point for the user, where the enter the data about their query and their email address to receive the results and report. This figure represents the 'Web Interface' block from figure 3.1.

6.3 Screen Objects and Actions

The figures in this section describe the actions that correspond to each aspect of the corresponding screenshot in section 6.2. Figure 6.5 has more details for figure 6.1.
Figure 6.1: Web Design Screenshot

Figure 6.2: TwapperKeeper Results Page

Figure 6.3: Twitter Login Page
**Figure 6.4:** ProjFocusedCrawler Results Page

**Figure 6.5:** Screen Objects and Actions Table

<table>
<thead>
<tr>
<th>Event type</th>
<th>The user must select from a pull down menu the category of the event they are trying to archive.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keywords</td>
<td>The user must type in keywords connected to the event separated by commas or blank-spaces.</td>
</tr>
<tr>
<td>Hashtags</td>
<td>The user must enter the hashtags they want searched on twitter pertaining to the event, including the pound symbol.</td>
</tr>
<tr>
<td>URLs</td>
<td>The user will type in the urls connected to news events they already collected.</td>
</tr>
<tr>
<td>Date</td>
<td>The user must select the date the event occurred using a calendar pull down feature.</td>
</tr>
<tr>
<td>Email</td>
<td>The user will type in his or her valid email address to receive email notification the process is complete.</td>
</tr>
<tr>
<td>Submit</td>
<td>The user will click submit to begin the archiving process.</td>
</tr>
</tbody>
</table>
Chapter 7

Implementation Plan

7.1 Team Roles

7.1.1 Joe Acanfora

Joe Acanfora will be in the role of chief python scripter. The responsibilities for this role includes being in charge of the execution of the python scripts coming from the input front end, and to build the flask python backend to manage projFocusedCrawler script.

7.1.2 Jeff Morris

Jeff Morris will be in the role of Front End Operative and PHP scripting specialist. The responsibilities for this role include creating the front end web application that users will interact to the service with. The PHP scripting specialist aspect of this position includes the being responsible for the YourTwapperKeeper script.

7.1.3 Briana Crabb

Briana will be in the role of Communications Operative. The communications operative is responsible for all external communications with the client and building a personal relationship with the client to effectively impart the progress and current state of the project as well as relaying feedback from the client back to the rest of the team. Other responsibilities include keeping detailed notes of all interactions with the client and to build the report generation and automated email scripts that will be sent to users. Briana will be in the best position to build the email report scripts because of her close
contact with the client and insider knowledge of the information they will need in said report.

### 7.2 Major Milestones

<table>
<thead>
<tr>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/30/15</td>
<td>Retrieve Scripts for review</td>
</tr>
<tr>
<td>2/6/15</td>
<td>Develop Webpage to initiate process</td>
</tr>
<tr>
<td>3/20/15</td>
<td>Finish Script to begin Article Crawling Process</td>
</tr>
<tr>
<td>4/3/15</td>
<td>Finish Script to get Log files</td>
</tr>
<tr>
<td>4/17/2015</td>
<td>Finish script that communicates with the wayback machine</td>
</tr>
<tr>
<td>4/24/2015</td>
<td>Finish email notification and start final write up</td>
</tr>
<tr>
<td>5/8/2015</td>
<td>Finish user manual and documentation</td>
</tr>
</tbody>
</table>
### NewsEvent Project Schedule

**Multimedia and Hypertext Capstone**

**Project Lead:** Joe Acanfora  
**Project Start Date:** 2/2/2015 (Monday)  
**Today’s Date:** 3/4/2015 (Wednesday)  
**Display Week:** 1

<table>
<thead>
<tr>
<th>WBS</th>
<th>Task</th>
<th>Lead</th>
<th>Start</th>
<th>End</th>
<th>Cal Days</th>
<th>% Done</th>
<th>Work Days</th>
<th>Days Done</th>
<th>Days Left</th>
<th>Week 1</th>
<th>Week 2</th>
<th>Week 3</th>
<th>Week 4</th>
<th>Week 5</th>
<th>Week 6</th>
<th>Week 7</th>
<th>Week 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.3</td>
<td>Host Web Site</td>
<td></td>
<td>Mon 3/9/15</td>
<td>Wed 3/11/15</td>
<td>3</td>
<td>0%</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Project Management</td>
<td>John</td>
<td>Mon 2/2/15</td>
<td>Tue 5/19/15</td>
<td>107</td>
<td>77</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.1</td>
<td>Retrieve Scripts to Review</td>
<td></td>
<td>Mon 2/2/15</td>
<td>Mon 2/2/15</td>
<td>1</td>
<td>100%</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.2</td>
<td>Documentation</td>
<td></td>
<td>5/8/2015</td>
<td>Thu 5/14/15</td>
<td>7</td>
<td>0%</td>
<td>5</td>
<td>0</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Scripting</td>
<td>Joe</td>
<td>Fri 2/13/15</td>
<td>Mon 4/27/15</td>
<td>7</td>
<td>52</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.1</td>
<td>Finish Script for Twitter Data pull</td>
<td></td>
<td>Fri 2/13/15</td>
<td>Thu 2/19/15</td>
<td>7</td>
<td>0%</td>
<td>5</td>
<td>0</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.2</td>
<td>Finish Script for Article Crawling</td>
<td></td>
<td>Fri 2/20/15</td>
<td>Wed 3/18/15</td>
<td>27</td>
<td>0%</td>
<td>19</td>
<td>0</td>
<td>27</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.3</td>
<td>Finish Script to call to wayback machine</td>
<td></td>
<td>Thu 3/18/15</td>
<td>Sat 4/4/15</td>
<td>17</td>
<td>0%</td>
<td>12</td>
<td>0</td>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.4</td>
<td>Script Testing</td>
<td></td>
<td>Mon 4/6/15</td>
<td>Sun 4/26/15</td>
<td>21</td>
<td>0%</td>
<td>16</td>
<td>0</td>
<td>21</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Reporting Service</td>
<td>Briana</td>
<td>Wed 4/1/15</td>
<td>Mon 5/4/15</td>
<td>34</td>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.1</td>
<td>Create email report template</td>
<td></td>
<td>Wed 4/1/15</td>
<td>Tue 4/7/15</td>
<td>7</td>
<td>0%</td>
<td>5</td>
<td>0</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.2</td>
<td>Create Script to Populate Report</td>
<td></td>
<td>Wed 4/8/15</td>
<td>Fri 5/1/15</td>
<td>24</td>
<td>0%</td>
<td>18</td>
<td>0</td>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 7.1:** Gant Chart weeks 1 through 8
### NewsEvent Project Schedule

**Multimedia and Hypertext Capstone**

**Project Lead:** Joe Acanfora  
**Project Start Date:** 2/2/2015 (Monday)  
**Today’s Date:** 3/4/2015 (Wednesday)  
**Display Week:** 9

#### Gant Chart weeks 9 through end

<table>
<thead>
<tr>
<th>WBS</th>
<th>Task Description</th>
<th>Lead</th>
<th>Start</th>
<th>End</th>
<th>Cal Days</th>
<th>% Done</th>
<th>Work Days</th>
<th>Days Done</th>
<th>Days Left</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.3</td>
<td>Host Web Site</td>
<td></td>
<td>Mon 3/9/15</td>
<td>Wed 3/11/15</td>
<td>3</td>
<td>0%</td>
<td>3</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>Project Management</td>
<td>John</td>
<td>Mon 2/2/15</td>
<td>Tue 5/19/15</td>
<td>107</td>
<td>77</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.1</td>
<td>Retrieve Scripts to Review</td>
<td></td>
<td>Mon 2/2/15</td>
<td>Mon 2/2/15</td>
<td>1</td>
<td>100%</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2.2</td>
<td>Documentation</td>
<td></td>
<td>5/8/2015</td>
<td>Thu 5/14/15</td>
<td>7</td>
<td>0%</td>
<td>5</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>Scripting</td>
<td>Joe</td>
<td>Fri 2/13/15</td>
<td>Mon 4/27/15</td>
<td>74</td>
<td>52</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.1</td>
<td>Finish Script for Twitter Data</td>
<td></td>
<td>Fri 2/13/15</td>
<td>Thu 2/19/15</td>
<td>7</td>
<td>0%</td>
<td>5</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>3.2</td>
<td>Finish Script for Article Crawl</td>
<td></td>
<td>Fri 2/20/15</td>
<td>Wed 3/18/15</td>
<td>27</td>
<td>0%</td>
<td>19</td>
<td>0</td>
<td>27</td>
</tr>
<tr>
<td>3.3</td>
<td>Finish Script to calls to Wayback</td>
<td></td>
<td>Thu 3/19/15</td>
<td>Sat 4/4/15</td>
<td>17</td>
<td>0%</td>
<td>12</td>
<td>0</td>
<td>17</td>
</tr>
<tr>
<td>3.4</td>
<td>Script Testing</td>
<td></td>
<td>Mon 4/8/15</td>
<td>Sun 4/28/15</td>
<td>21</td>
<td>0%</td>
<td>15</td>
<td>0</td>
<td>21</td>
</tr>
<tr>
<td>4</td>
<td>Reporting Service</td>
<td>Briana</td>
<td>Wed 4/1/15</td>
<td>Mon 5/4/15</td>
<td>34</td>
<td></td>
<td>24</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.1</td>
<td>Create email report template</td>
<td></td>
<td>Wed 4/1/15</td>
<td>Tue 4/7/15</td>
<td>7</td>
<td>0%</td>
<td>5</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>4.2</td>
<td>Create Script to Populate Report</td>
<td></td>
<td>Wed 4/8/15</td>
<td>Fri 5/1/15</td>
<td>24</td>
<td>0%</td>
<td>18</td>
<td>0</td>
<td>24</td>
</tr>
</tbody>
</table>

**Figure 7.2:** Gant Chart weeks 9 through end
Chapter 8

Prototyping Report

8.1 Background: Changes in structure

In the past we initially stated that we would be developing the backend using Python for the ease of use for starting the ForwardCrawler script which was already written in Python. Over time we have discovered that this tradeoff of potential ease of use vs time was not paying off as we had expected. The team came to a conclusion to stop development of the Python backend and start over the development with a PHP/Apache based webserver as our backend. This has dramatically increased our development speed due to the groups familiarity of PHP.

8.2 Current state of Development

8.2.1 Cloud 9

For development purposes we immediately looked for an online platform that the entire group could simultaneously develop on at once. In the beginning the group was split between PythonAnywhere and Cloud9. Over time with the belief that Python wasn’t going to be our backend language of choice we decided to go for Cloud9 for its free access and the ability to have multiple users edit a file at once. An additional factor which was a major issue for us was that Cloud9 was free of cost for relatively small teams which perfectly suited our team. There were a few drawbacks with the Cloud 9 platform which we will discuss later but overall the system has been flawless.
8.2.2 PHP Backend

The PHP backend allows for ease of use for transferring data from the front end to the backend through the use of POST and GET HTTP requests. In addition PHP allows us to develop quickly and easily through the use of Apache which has allowed us to install addons such as PHP Curl quickly and effortlessly. In addition Apache allows for ease of use of MySQL which we later discovered useful for the initial set up of the YourTwapperKeeper.

8.2.3 YourTwapperKeeper Server

We assumed the YourTwapperKeeper script will be run on an independent server so we set up an additional server on Cloud 9 to replicate this process. In order to completely replicate this we created a Twitter account and app associated with in order to get oAuthenticate to work with the Twitter API. We then set up the server by creating the necessary database and tables which then allowed the website to be run. We initially had and continue to have issues dealing with oAuthentication while using our Twitter Account. We have verified our setup of the server with our sponsor and continue to make efforts to track down the issue. We currently have traced it to an HTTP request so we suspect there might be an issue with a library not being allowed by Cloud 9.

8.2.4 PHP Curl Initialization

In order to interface with the remote YourTwapperKeeper server we needed a tool which would allow us to interact with remote servers and webpages. There were a handful of
tools available to us through Python that would allow us to Crawl a website but the most prominent of those available was Curl which was ideal for passing information to other websites through the POST HTTP Request method. Curl also allowed for us to handle the data which it would receive through a variety of methods. Figure 8.2 shows how we tested Curl by grabbing the website at the domain (www.example.com) and displayed it as the background of our backend webpage. Once we have the YourTwapperKeeper oAuthentication to work we can login to Twitter and then send the data entered by the user to the YourTwapperKeeper site using this tool.

8.2.5 Forward Crawler

The Forward Crawler script is by far the more difficult of the two scripts that we have had to deal with. After some analysis it clearly shows itself as a Classifier in Python that uses NLTK but still hides how it is managed and run. To run this script we discovered a PHP feature that allows us to run bash commands from within Python. This was the main reason why we wanted to avoid PHP in the first place but fortunately there was support for what we needed to do.

8.2.6 Email

We needed a verification email to send to the user when their request has been processed and which also contained links to their running scripts. Fortunately PHP has a built in function called mail() which allowed for us to email the user through the SMTP mail server built into Apache. We designed and implemented a HTML email with the
Bootstrap layout that we later will rebuild with internal CSS which will be sent to the user. Figure 8.3 was what we came up with. Unfortunately for us during the testing of this script we were unable to get the email to be received on our end. After debugging and discovering that the email was being confirmed to being sent to the server we found out that Cloud 9 blocks the mail() function in PHP to prevent free accounts from becoming spam servers. The code that we have written has been tested on other Apache based servers and will work once we move off the Cloud 9 system.

8.3 Moving Forward

We plan to test the YourTwapperKeeper on another Apache server to test whether there is an internal PHP library being blocked which is preventing it from running. In addition we plan to understand and begin writing a script for the Forward Crawler Python Script.

8.4 April 8th Update

8.4.1 YourTwapperKeeper

After not being able to successfully log any entries into the yourTwapperKeeper due to an unknown glitch we decided that there was potentially an issue with cloud 9 (the service we were using for the server) that was blocking a library that the script was depending upon. Instead we uploaded it onto a server that one of our members maintains which is run by Bluehost. After a few hours of debugging we were successfully able to launch the YourTwapperKeeper on that server and were able to login to the script using the TwitterOAuth. Then after successfully logging into the system we were able to add fields for the YourTwapperKeeper script to monitor. There was one issue with the server that was whenever the YourTwapperKeeper attempted to run a monitoring process for each field the server would automatically kill that subprocess for a reason we have yet to discover. Following the success of the YourTwapperKeeper server we went back to the phpCurl to see if we could login to Twitter as practice before trying to tackle the Twitter OAuth. We successfully accomplished this through PHP Curl.
8.4.2 Moving Forward

Our next goal on this front is to successfully login via the Twitter Oauth and make entries for the yourTrapperKeeper only through the PHP Curl script.

8.5 April 21st Update

8.5.1 ProjFocusedCrawler

It was determined to run the ProjFocusedCrawler script from a flask backend hosted on pythonanywhere. Because this script is running as a separate service, http/get requests have to be used to communicate between the Flask backend and the PHP backend.
Chapter 9

Testing Report

9.1 Project Status

9.1.1 Web Front End

Enter your Query here: http://babs.dlib.vt.edu/twitter/index2.PHP Once you click submit you will automatically be redirected to the YourTwapperKeeper Page.

![Web Front End Screenshot](image)

Figure 9.1: Web Front End Screenshot

9.1.2 YourTwapperKeeper

YourTwapperKeeper can be seen here: http://babs.dlib.vt.edu/twitter/index.php
9.1.3 Email Reporting Service

The basic outline of the email report has been built and tested. The emails successful send a summary report to the user with a link to where they can view their results.

9.1.4 ProjFocusCrawler

ProjFocusedCrawler is still in the development phase. An approach and design have been finalized with the client and writing, and testing of the code are under way. ProjFocusedCrawler will use a python Flask web application that will call the startCrawler script using a subprocess. The resulting output will be redirected to the webpage for the user to see as the script makes progress and emits its output.
9.2 Testing

9.2.1 Testing Method

The group preformed usability testing with the project client. The client was able to easily enter the required information into our front end page. He submitted the form to make the first of the two scripts run. He mentioned adding a second page to display the output of the webcrawler URLs. He also requested the group to look into an issue with the twitter username not displaying along the newly created twitter archive. Besides these minor changes/ fixes the client was happy with the product. The product was simple and easy to use and produced the desired results.

Stress testing (sometimes called torture testing) is a form of deliberately intense or thorough testing used to determine the stability of a given system or entity. It involves testing beyond normal operational capacity, often to a breaking point, in order to observe the results.

A stress test was performed by several group members. Each member of the team created several different searches to be performed. The program was able to create a new process to run. However we determined that only three news events will be crawled at a time. The email chain worked for all the events created (show joes email screen shot and my email screen shot). We also input nonsense/ useless data into our front end. (see image of the ones with jeffs useless input) These useless input did not crash the program, they just produced no results (see image).

9.2.2 Testing Result

The testing results showed our product was easy for the consumer to use. It also demonstrated that our scripts accurately call the provided scripts to produce the desired automation of webcrawling.
Chapter 10

User Manual

10.1 Introduction

To run our python scripts that call our automated scripts the user must go to our webpage http://babs.dlib.vt.edu/twitter/index2.php. It is recommended the user use any browser that supports bootstrap. Bootstrap is compatible with all modern browsers (Chrome, Firefox, Internet Explorer, Safari, and Opera). The user does not need any other software installed; he or she only needs to enter the required data into the input fields to perform their desired query. Below describes the input fields.

10.2 How To Execute Query

10.2.1 Event Type

This is a required field. The user can select from 4 specified classifications of events or they can choose other.

10.2.2 Keywords

This is a required field. The user must enter the keywords they would like to use for their query. These words will be used to search through twitter and the web. They are entered using a space as a delimiter.
10.2.3 Description

This is a required field. The user must enter a few words to describe their query. The words are entered using a space as a delimiter. The description does not affect the query.

10.2.4 Hashtags

This is an optional field. If the user knows hashtags that exist in correlation with their event they can input these to assist with the twitter search. These must be entered with the # symbol with a space as a delimiter.

10.2.5 Date

This is an optional field. If the user knows the date an event occurred he or she can enter the date to assist with the query. The date should be formatted as MM/DD/YYYY.

10.2.6 Email

If the user wants an email notification he or she may enter a valid email address. The user will receive an email update to an email entered in this field.

10.2.7 Positives

This a required field. The user must input a text file that contains a list of initiate URLs where projFocusedCrawler will start.

10.2.8 Submit

Once all the necessary fields have been filled out the user will click this button to start their query. Two pages will be brought up showing the results of the twitter and web crawl.
Chapter 11

Developer Manual

11.1 Configuring Project Focused Crawler

11.1.1 Introduction

The ProjFocusedCrawler script must be run with python 2.7, however the flask application that will display the output of that script to the browser requires python 3.4, to avoid Unicode errors and for the necessary text streaming tools available in python 3 +.

11.1.2 Files

1. All files from the ProjFocusedCrawler git repository which is linked to in the appendix.

2. flaskapp.py: This file is the web application that launches the script and displays its output to the browser. Here you may have to modify the change directory command to point to the location of the ProjFocusedCrawler script.

3. CrawlWeb.py: This file should be placed in the ProjFocusedCrawler/src directory. It is the main script used to the launch the startCrawl function.

11.1.3 Required packages

These are the python packages required to be able to run this web application without errors.

1. install python-pip
2. install nltk

3. Install Setuptools: http://pypi.python.org/pypi/setuptools

4. Install Numpy (optional): run sudo pip install -U numpy

5. Install NLTK: run sudo pip install -U nltk

6. Test installation: run python then type import nltk

7. download nltk stopwords

8. bash-$ python

9. >>> import nltk

10. >>> nltk.download()

11. downloader: d stopwords

12. pip install beautifulsoup4

13. pip install requests

14. yum install python-devel

15. yum install python-lxml

16. pip install readability

17. pip install readability-lxml

18. install standford ner from https://github.com/dat/pyner

19. pip install gensim

20. pip install sh

11.2 Configuring YourTwapperKeeper

11.2.1 Introduction

The developer must have Apache installed on their server, preferably the most recent version for security concerns. On top of this server the developer must install PHP 5.3.3 or newer.
11.2.2 Files

1. Index2.php: This is the file that the user enters the information about the type of query they want entered into yourTwapperKeeper and the type of file uploaded to the Focused Crawler. The file was constructed using Twitter Bootstrap version 3 and is currently configured to reject any invalid queries. When the submit button is clicked it sends data to the jeffScript.php via the POST http request.

2. jeffScript.php: This php file is the backend of the index2.php and receives the HTTP request from index2.php and then processes whether the data is valid. It also attempts to clean the data in case of any low level attacks are entered. The script will process the incoming file containing the positive words and upload it to a directory called uploads and rename the file to pos.txt. It then processes the query and starts a new YourTwapperKeeper entry based off of its parameters. To modify the emails:

3. index.php: This is the main page for the yourTwapperKeeper. This is the location where all of the entries are shown and their current status. From this page you're also able to delete or edit any existing queries or manually enter in new queries.
Chapter 12

Lessons Learned

12.1 Time line & Schedule

Sticking with the timeline was difficult because we had to change our backend architecture and server host locations several times during the implementation process. Finding the correct combination of web services and languages was a trial and error process that resulted in taking longer than expected. Once the proper tools were agreed upon the work continued at the planned pace.

12.2 Problems and Solutions

12.2.1 Cloud9 and Email Complications

When we first addressed this problem we set up a cloud9 server to use as our testing platform for all of our code. Initially we were successful in installing all of the required software onto the cloud9 platform but when it came to running our code we were introduced with unexpected issues with none of our code operating as it should. After numerous hours of debugging we discovered that the cloud9 platform blocked specific PHP functions from being called which prevented our scripts from being run. To solve this we realized we needed another platform to develop on that would give us more freerange. We initially tried to run it on a server that one of our team members owned but that opened up more issues with lack of SSH access into the server. Then one of our sponsors colleagues allowed us to run a virtual machine that we could all SSH into which provided us the ability to run the scripts without any issues.
12.2.2 calling scripts non locally

Our original strategy to tackle the approach to transfer the data from our server to the yourTwapperKeeper server was to use an external post push through PHP which would have allowed for a simpler transition and more flexibility. Unfortunately this introduced new complications mainly around not having the amount of access that is given externally versus a local PHP file which would have access to all of the local yourTwapperKeeper PHP files.

12.2.3 deciding what backend structure/ framework would be easiest to implement

Initially we decided upon a Python based backend due to what we initially thought would be mostly python scripts that we had to deal with. Once we discovered that yourTwapperKeeper was completely managed in PHP we switched to an Apache server and changed our development over to PHP. We later realized we also needed a Python script to handle the Project Focused Crawler so we added a Python based Flask server to handle the Python then send the results back to the PHP based Apache server which handled the Email.

12.2.4 Executing Scripts from Flask

Streaming the output of a python script in Flask was not immediately obvious. Even calling an external python script proved to be difficult. The eventual solution was to use a python package called ‘sh’, which allows the execution of bash shell commands. After executing the shell script a Flask Response function is used to display the output of the script in real time, by iterating through each line of the shell output and streaming it to the browser.

12.3 Future Work

fixing the user name login formatting the emails using different urls as the seed and pos parameters.

The output of ProjFocusedCrawler needs to be redirected into ”The Way Back Machine,” which is a script that iterates through the URLs and stores the textual data from the corresponding web pages.
Chapter 13

Acknowledgments

"Great mentorship is priceless." - Lailah Gifty

Special thanks to:

- Mohamed Magdy Garib Faraq, project sponsor - mmagdy@vt.edu
- Dr. Edward A Fox, class professor - eafox@vt.edu
- Sunshin Lee, external resource - sslee777@vt.edu
Appendix A

Appendix

A.1 Scripts

John Obrien III, yourTwapperKeeper, (2013),
GitHub repository, https://github.com/540co/yourTwapperKeeper

Doctoral Canidate M. Farag, ProjFocusedCrawler, (2015),
GitHub repository, https://github.com/mohamed83/ProjFocusedCrawler