

Website Summary Project

Summer Internship 2016

Sulav Poudyal, Computer Sciences Department
May 2017

Internship Report
In completion of course credit requirements for ENGR3000
Texas Tech University

Contents:

1. R&D Group, University Library, Texas Tech University	3
2. Website Summary Project	5
a. Description	6
b. Tools Used	9
c. Setting Up Cluster	13
d. Scanner Code	15
3. Things Learned	37
4. Connection to Academic Study	37
5. Conclusion	39
6. Attachments	
• Research & Development Summer Project Internship 2016 (position description)	40

1. R&D Group, University Library, Texas Tech University

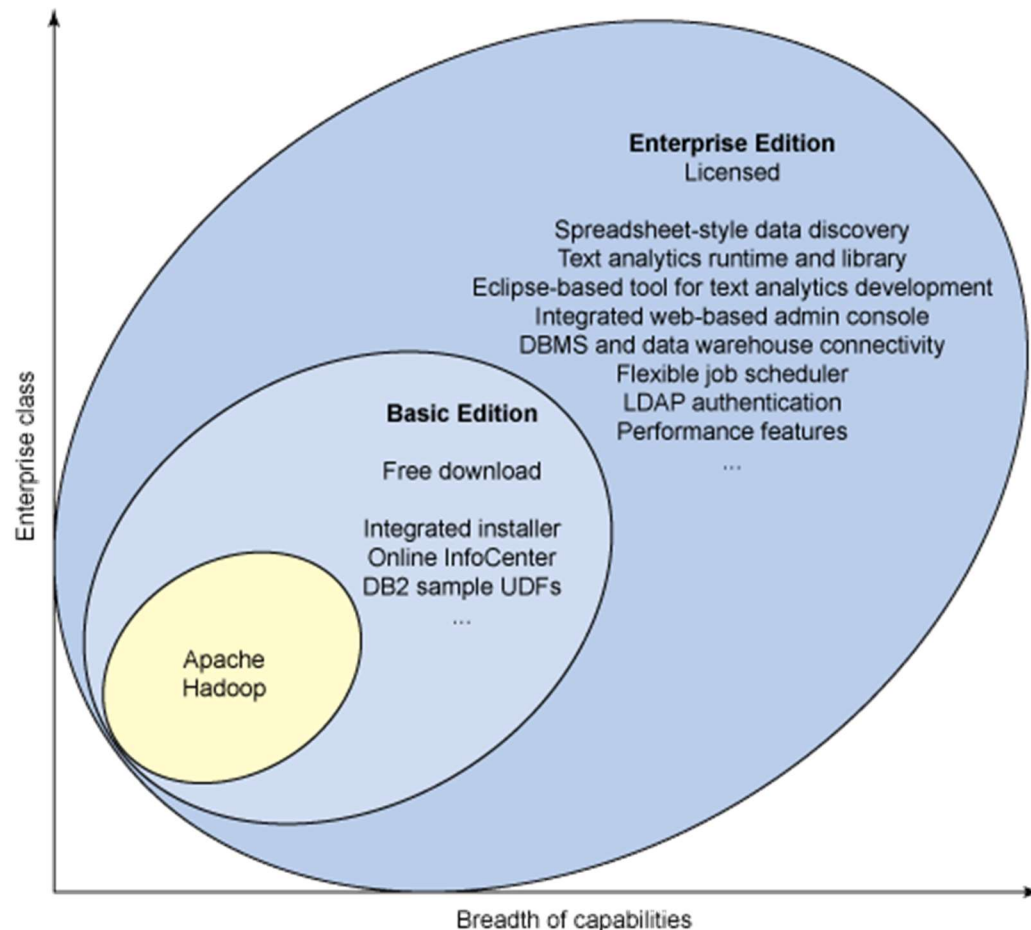
The University Library consists of the Main Library, Southwest Collections, Architecture Library, and the Old Barn Building (Conference Center and Artist in Residence). Over 50 faculty librarians are present throughout the Texas Tech campus Library system along with more than 110 staff positions and 165 student assistants.

I performed my Summer 2016 Internship responsibilities in the Research and Development (R&D) Group at the Main Library of the University Library of Texas Tech University (TTU). The group is located on second floor of the University Library main building, on the west side of Digital Media Studio. The Library R&D Group is part of the Library Technology Management Services Department (LTMS) of the Library, which has 30 full-time employees. In addition, there are also from 3 to 7 student assistants. The campus Central Information Technology Division supplies centralized network services for the Library, but the Library handles its own support, installation and maintenance of the Library's computer systems (approximately 250 public computers and 125 staff computers, various servers and clusters) and technology services as well as building control systems (phone, elevator, emergency, and server room). Along with the in-house systems the Library also maintains electronic journal subscriptions, electronic databases, repositories, the online catalog—all of which involve various combinations of remote and local hardware—and a number of internal processing systems which connect with campus systems for student affiliations (such as EZproxy). Within LTMS are represented the following sections: Digital Resources, Public Operations, Informatics Lab, 3D Animation Lab, Computer Support, Programming, and R&D. Each of the groups reports to the Associate Dean for Library Technology Management Services.

LTMS also works closely with TOSM (Technology Operations Server Management), which is part of Central Information Technology. TOSM physically houses some Library servers, but the Library maintains the operating systems and their applications. TOSM is also responsible for server backup.

The TTU Library has just over 200 wireless access points with a capacity of up to 8,000 simultaneous connections. So far peak load has been around 2,500 with

approximately 10,000 visitors in foot traffic per day. The Library digitizes and makes available to patrons major portions of its collections and has about 225 TB disk storage for such purposes. The Library has its own Server Room located on the First Floor, where it houses racks of computers. One is devoted to a cluster of 14 Dell R5400 systems, eight of which is involved in my task to build a BigInsights cluster. Another rack houses in-house systems for elevator, telephone, emergency use and the Research Data Management test system. The Library collaborates with Central Information Technology and the Office of the Vice President of Research to maintain the 100 TB iRODS system which is used for campus Data Research Management.



IBM BigInsights Basic and Enterprise Editions

(Source: <https://www.ibm.com/developerworks/data/library/techarticle/dm-1110biginsightsintro/index.html>)

The R&D Group has had staff numbers ranging from two to six over time. Its functions are to explore and investigate innovative technologies, including operating systems, software, hardware and tools. The Group currently maintains several research server clusters running Red Hat Linux. It is a member of the IBM Academic Initiative, which gives it access to commercial versions of IBM software, documentation and training materials. Some of the server clusters are running IBM BigInsights, which is an expansive toolset of applications used in Big Data, including database and data warehouse functions, text analytics, and interfaces for web crawling and social media. BigInsights uses the Hadoop distributed file system. Two faculty members who are part of the R&D Group write scholarly articles for publication. The direction of the R&D Group depends on the needs of the LTMS area. It contributes to the Library Knowledge Base of technical materials, conducts campus surveys, and explores technologies such as Raspberry Pi, 3D printing and modeling, and virtual reality.

The TTU Library is currently reviewing a 30-million-dollar-plus expansion proposal which would renovate space, offer increased offsite storage, include more access areas for students and classroom, add an auditorium with seating capacity up to 1,000 spaces, and combine stack levels by removing stack level flooring to create large study spaces. In addition, the TTU Library is considering submitting a proposal for establishing a School of Library and Information Science, which will bring graduate students onto campus and coordinate with related departments on campus. If this proposal moves forward, the first students would attend in the Spring of 2019.

2. Website Summary Project

The Website Summary Project originated from the Library's need to frequently examine many of the same websites over time and look for new information about what was changing at those websites. The number of pages involved and the number of

websites makes manual methods both tedious and cost prohibitive. In addition, the same websites would need to be checked periodically for new information and changes. For example, the Association of College and Research Libraries (ACRL) (<http://www.ala.org/acrl/aboutacrl>) is one grouping of libraries that is of interest to TTU librarians, since many of the innovations in librarianship will be announced in those web pages. The top 100 ACRL libraries list is of particular interest in this regard. In the Website Summary Project, a *website* is defined to be *all the URLs that can be reached from a starting URL*. For example, the TTU Library main URL is <http://www.library.ttu.edu> and the website that can be reached from this starting URL includes approximately 1,500 web pages. The total number of web pages in the top 100 ACRL libraries represents a significantly larger number of pages. A web crawl of the complete TTU Library website can approach multiple gigabytes of data. The Website Summary Project is based on the idea of storing periodic web crawls of a given website. The crawl data can be checked for specific content and the website can be reviewed from time to time for changes.

The need for the Website Summary Project came about from an earlier review of 3D animation lab implementations which are housed at American universities. The TTU Library was planning to implement a 3D Animation Lab and was looking for examples of other 3D animation lab sites to initiate contact with those universities for possible discussion, and to help determine how its design of a 3D animation lab should take place. The Library was interested in what tools were being used, what policies govern how the lab is run, and what kinds of training materials and reference materials would be the most effective. The Library R&D team informed me about how difficult it was to do web searching of the term “3D” and how many misleading search hits could occur. The result was that staff performed a high number of manual web searches with manual review of the web search hits.

a. Description

My initial role in the Website Summary Project was to develop some application tools for processing website crawl data. The training for my role began when I was a Student Assistant in the R&D Group where I learned Microsoft Visual Basic in the Visual

Studio framework and learned about using ADO methods for working with a Microsoft Access database. I also learned Red Hat Linux, versions 6 and 7, Enterprise Editions. My coding projects take place in a Windows 10 environment and the BigInsights software runs on a Linux cluster. We use a smaller two-server test cluster to develop procedures and training for using the software, and I worked on configuring a larger eight server cluster for production processing of web crawl data. The eventual goal of the Website Summary Project is to develop a system which can allow users to initiate their own website requests and receive reports on the results of those requests.

Although TTU Library staff routinely work with the websites of other libraries, the tools which are being developed are more general and allow for the crawling and summary of information for a wide range of websites. The tools in the BigInsights web crawl allow specifying how much data is to be retrieved, by indicating the depth of the crawl and specifying how many pages should be retrieved at each level. Crawler data is produced in the Hadoop file system running on BigInsights and then downloaded to Windows for my processing using tools which are part of BigInsights and manage the transfer from Hadoop. Apart from the programming, database work, hardware and cluster configuration, installation and testing, I also worked on several research areas.

Approximately half of my 20 hours per week during the Summer Internship was devoted to research. This included readings in Natural Language Processing and learning about the Raspberry Pi. I examined articles, blogs and books on these topics. There was also some opportunity to explore Virtual Reality, 3D Printing and Modeling, though these were on a more limited basis as time permitted. The goal of the research portion of my job was to help me understand some of the general issues in processing natural language and create an awareness of techniques that are used by others working in this area. The work with the Raspberry Pi was to increase my understanding of computer hardware, especially by configuring the hardware on the board to perform various tasks such as enabling Wi-Fi or getting a camera to work with the board.

Report writer

The primary goal of the Website Summary Project is to produce a report writer which will output the principal contents of the website. The reports will give the most frequent to least frequent term and allow the user to mark the specific terms. The user can also define specific items that should be included in the report, regardless of frequency. The reports will also provide the context for terms which will give the user a better understanding of the content. The report gives the user a summary of the website and allows the user to avoid the need to go to specific individual web pages, which overall represents saving a lot of time.

Ultimately we want to automate website searches for any term in any website. Currently, there are no tools that can summarize the website this way. Google searches for the term but does so based on the keywords of the website. It does not give a summary of most frequent terms.

The stages in production for the Website Summary Project are these in chronological order:

- Configure BigInsights Test System
- Generate Sample Web Crawls for code development
- Write Scanner code which derives TOKENS for Access database
- Build Access database
- Implement control for Synonyms
- Write report writer which uses Access database
- Configure and test 8-server cluster of BigInsights for faster web crawls
- Develop file management procedures to handle accumulated data
- Develop user interface to support user self-service

Use of Synonyms

We plan to bring together synonyms of searched term so that related terms will be grouped together in the report. The databases we can use are Big Huge, Wordnet.

WordNet is a large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical

relations. The resulting network of meaningfully related words and concepts can be navigated with the browser. WordNet is also freely and publicly available for download. WordNet structure makes it a useful tool for computational linguistics and natural language processing.

WordNet superficially resembles a thesaurus, in that it groups words together based on their meanings. However, there are some important distinctions. First, WordNet interlinks not just word forms—strings of letters—but specific senses of words. Thus, words that are found in close proximity to one another in the network are semantically disambiguated. Second, WordNet labels the semantic relations among words, whereas the groupings of words in a thesaurus does not follow any explicit pattern other than meaning similarity. (<https://wordnet.princeton.edu/>)

Big Huge thesaurus is based on the WordNet database. It gives noun synonyms, verb synonyms where applicable and it also returns similar sounding and rhyming words. We want to use the database that can return result in text format instead of html. It will remove the need to filter the result for html tags.

We encountered some issues while trying to use the synonym database. We found that integrating the database with our program is a difficult task. We have two options, use API to communicate with online database or use a local database of synonyms. Some of the databases have to be purchased. Some allow certain number of queries per day.

We want our report writer to read the database and select report results based on the user input. The report writer will look for the synonyms on the database for the user's input. Then it will display the summary containing the user's input and their synonyms.

b. Tools Used

Web crawler Tools

Since we were working with websites we looked at various web crawler tools. A web crawler is used for web indexing/web spidering the World Wide Web. Most search

engines use this technique to perform a web search. The tools that are available for web crawl are HTTrack, Nutch, Crawler4j, IBM BigInsights and a number of others. We decided to go with IBM BigInsights.

IBM BigInsights

IBM BigInsights together with IBM Open Platform with Apache Spark and Apache Hadoop provide a software platform for discovering, analyzing, and visualizing data from disparate sources. BigInsights is a collection of value-added services that can be installed on top of the IBM Open Platform with Apache Spark and Apache Hadoop, which is the open Hadoop foundation. BigInsights is used to understand and analyze unstructured information. It can be used to work with big as well as small data. BigInsights helps data scientists, application developers, and administrators in an organization to build and deploy custom analytics to get meaningful information from data. IBM BigInsights package contains value-added services like Cluster management, Hive, HBase, Oozie, Flume, HDFS, YARN, MAP/Reduce, Zookeeper, Big SQL, Text Analytics, BigSheets, Big R and many others. (see: https://www.ibm.com/support/knowledgecenter/SSPT3X_4.2.0/com.ibm.swg.im.infosphere.biginsights.product.doc/doc/c0057605.html).

BigInsights also has tutorials for many of its services. IBM provides good documentation and videos to get started with the BigInsights. IBM BigInsights is provided by IBM's Academic Initiative program. It helps educators to teach market-ready skills. And it provides no-charge or discounted access to IBM software, courseware, hardware and technology resource. It also provides opportunities to build skills for students through student competitions; see for example: <https://developer.ibm.com/academic/programdetails/>.

BigInsights was developed as part of the IBM JumpStart program and has been around for a number of years. The first version released was 1.1 on 26th May 2011. The version we used was 3.0.0.2 which works with Red Hat 6 and appears functional with Red Hat 7, though it has many problems. It allows us to do more than just web crawl. It provides organized structure for retrieving web data. It also allows to select the number

of web pages to crawl. It provides choice of export formats. It delivers the result as a file with html as output.

We are trying to work with content of a webpage without the html tags so we needed to process the web crawl data. A web crawl returns the content of a webpage along with the html tags. We decided to use Microsoft Visual Basic to filter the web crawl data.

Microsoft Visual Studio

Microsoft Visual Studio was downloaded from Dreamspark. It is now called Microsoft Imagine. Historically many of the tools were available for library staff from Microsoft Imagine. The Library maintains full subscription to Microsoft Imagine. It provides software and tools for students through that subscription.

Microsoft Visual Basic was chosen because it is highly adapted to working with text strings. It provides ease of working with certain database products. It is simple to use and master compared to other programs. Its latest version is available in Microsoft Imagine for download. So, for these reasons it was chosen.

Microsoft Access characteristics and why it was chosen by the R&D Team:

- It is easy to use with Microsoft Visual Studio.
- Availability of Microsoft Access database software
- Ease of installation
- Availability of data inspection and data manipulation tools.
- Compatibility of various releases with earlier versions.
- It provides effective report writing and can be configured to produce complex reports
- It allows easy import and export of data
- It is highly documented and has numerous excellent tutorials available

Hardware

The initial approach was exploratory. We were interested in running on Red Hat Linux 7 Enterprise, but some questions remained about compatibility with IBM BigInsights. The Library R&D Team had hit some incompatibility problems on Red Hat 7 with earlier versions of the IBM product, and there was still some uncertainty whether these problems remained even after several updates of Red Hat 7 Enterprise. As a result, I tested initially on a small cluster of just two machines to determine what versions we should run for the larger production cluster. My exploration and work with the R&D Team eventually showed that we should continue to use Red Hat 6 Enterprise version due to remaining instabilities on Red Hat 7.

Some examples of the problems when using Red Hat 7 Enterprise version with BigInsights:

- A wide range of Java programming exceptions which did not occur under Red Hat 6 Enterprise version.
- BigInsights installer would report back not finding folders which it creates during the installation process while using Red Hat 7 Enterprise. These problems did not show up using Red Hat 6 Enterprise. The user does not create these folders and there is no obvious method to prevent them from happening under Red Hat 7 Enterprise; the errors simply go away using the earlier Red Hat 6 version.
- It should be noted that Red Hat 6 Enterprise is still fully supported by Red Hat and is the primary version used by most commercial users of the Red Hat products.

We are configuring a cluster of 8 nodes for IBM BigInsights. Initially the primary role of the cluster is to perform web crawls efficiently. In an earlier test, a full crawl of the TTU Library web site could take several days. We expect the time to be reduced to under a couple of hours with the 8 node cluster.

The servers we used were Dell R5400s. Its specifications are below:

- Processors: Dual-Core (6MB L2 cache) & Quad-Core (2X 6MB L2 cache) Intel Xeon Processors. 45nm architecture with Intel Virtualization Technology and Intel Trusted Execution Technology (TXT)

- Chipset: Intel 5400 chipset, which supports the latest generation of multi-core Intel® Xeon Processors, advanced ECC memory, and scalable industry standard graphics and storage options.
- Networking : Broadcom® 5754 NetXtreme 10/100/1000 Gigabit Ethernet controller.
- Key features of the Intel5400 Chipset on the Dell Precision R5400 workstation
Support for 64-bit Intel Xeon quad-core and dual core processors. Support for up to 32GB2 ECC system memory DDR2 fully buffered DIMM 667MHz ECC memory and quad channel memory channels 1333MHz front side bus Serial ATA hard drive support with host based RAID 0, 1 Quad Channel DDR2 FB DIMM Memory Quad channel FB DIMM memory delivers outstanding memory bandwidth and performance. PCI-Express (PCI-E) Bus Dual x16 Gen 1 graphics slots provide support for high performance industry standard graphics cards.
- Graphics: The Dell Precision R5400 offers a range of graphics cards from entry 2D to high performance OpenGL 3D with up to 768MB3 of graphics memory. Offerings include the NVIDIA® NVS 290, NVIDIA Quadro® FX 570, FX1700, FX3700, FX4600 and ATI® FireGL® V3600. Through comprehensive multi-monitor support, the R5400 graphics options from NVIDIA and ATI (AMD) enable workstation users who want additional real-estate to connect up to four displays (directly attached) and get a range of multi-monitor functionality that can help make them more productive. Up to two monitors can be remotely attached via the FX100 Remote Access Solution (R5400 remote access host card and user portal).
(See <http://site.pc-wholesale.com/manuals/Dell-Precision-R5400.pdf>)

c. Setting up the cluster

We first built the test cluster where we could install the software and test it on two nodes before building the cluster of 8 nodes. We installed Red Hat 6.9 on the nodes

and set up the subscription for it. We connected two nodes by secure shell. Secure shell allows two computers to connect and share files. Then we installed BigInsights 3.0.0.2 on them. We ran into many issues while installing BigInsights. The passwordless ssh connection was the main issue we encountered. After the installation of BigInsights, it is to be tested by doing a web crawl.

For the main cluster, we first set up the cluster in the server room of the university library. It is located at the ground floor. There were two server racks present. One of the racks is used for the library elevator system and emergency system. The other was empty and we stacked our servers in it. Then we installed Red Hat on each of the servers and set up the subscription. Red Hat subscription allows the download of latest updates and to information regarding Red Hat system. The servers had portable drives and IT department in the library had a cloning device to clone disk drives. We used that device, Kangaru, to clone the drives. There were two Kangaru machines, one with the capacity to clone 4 drives and another with a capacity to clone 14 drives. To do a full clone, it would take approximately 2.5 hours. We used the machine with higher capacity to clone our drives. BigInsights was installed on test cluster and is being tested. Installation of BigInsights will begin on main cluster after the testing is complete.

Test Cluster:

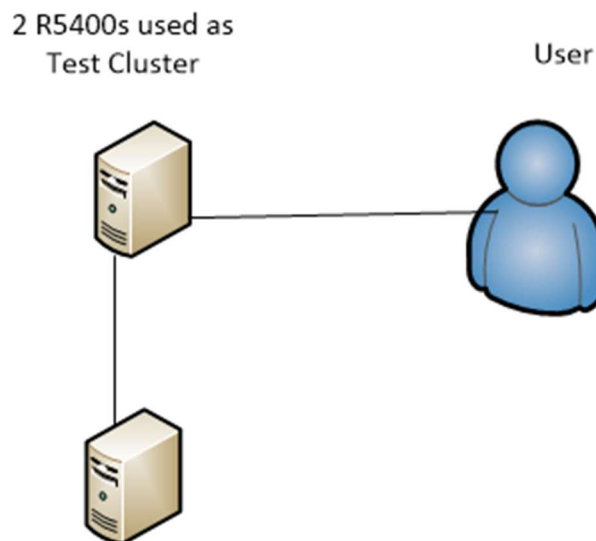


Figure: BigInsights Test Cluster

Main Cluster:

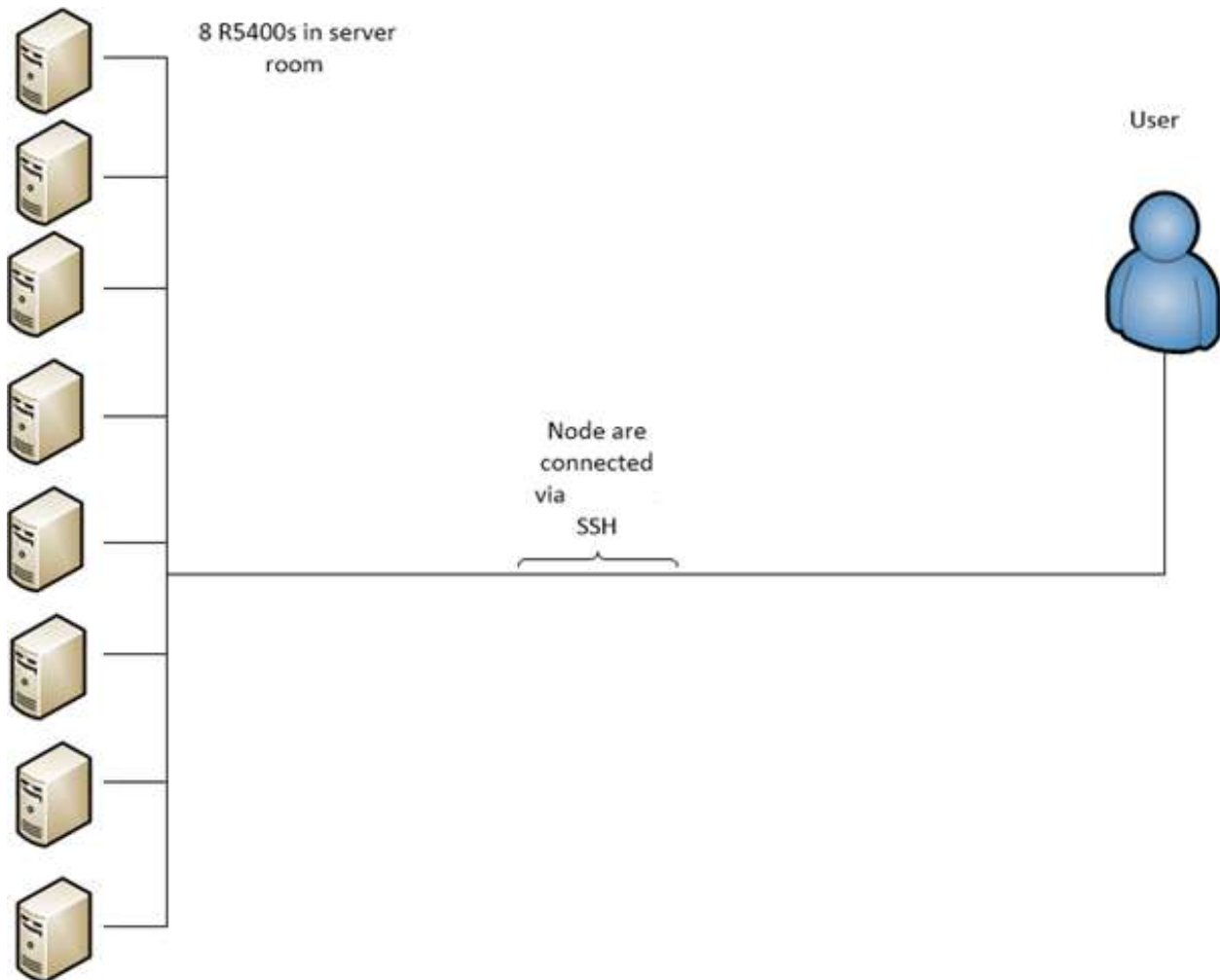


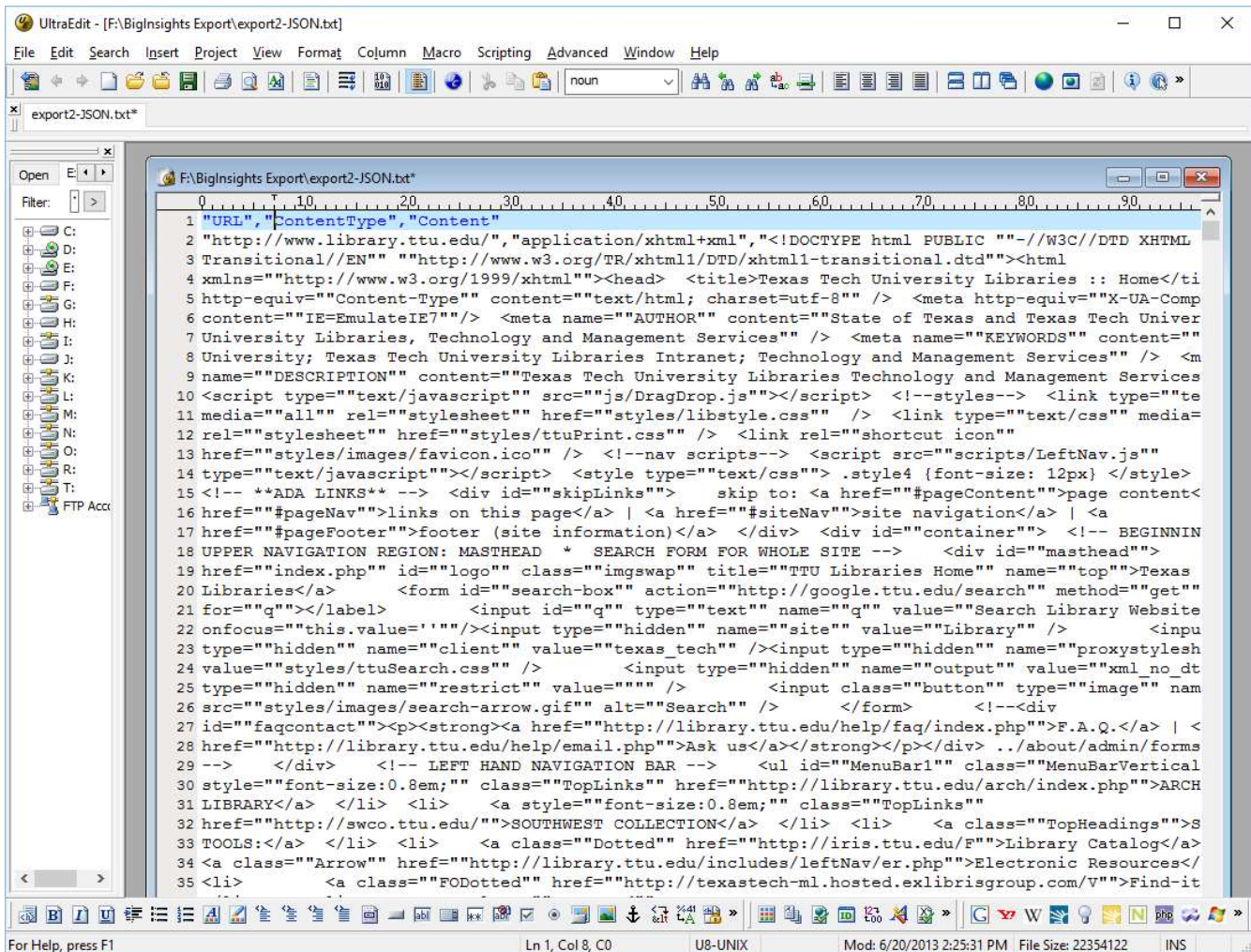
Figure: BigInsights Production Cluster

d. Scanner Code

Web crawl data (Illustration of sample data)

A web crawl was performed on <http://www.library.ttu.edu> using IBM BigInsights. Following is a sample of the web crawl data. Note that the file is larger than what is illustrated here. This is essentially a sample of how the data can be viewed from the web crawler component of BigInsights. Typically, web crawler data is presented as one file per URL. What BigInsights allows is the creation of a single file for the entire result

of what was crawled by the web crawler tool. A number of options are available for data presentation. Here JSON text format was used.



The screenshot shows the UltraEdit text editor with a file named 'export2-JSON.txt' open. The file contains a JSON representation of a web page source code. The JSON structure includes fields for 'URL', 'ContentType', and 'Content'. The 'Content' field contains the raw HTML source code of a web page, which includes a DOCTYPE declaration, meta tags, a title, a description, and various links and scripts. The editor's interface includes a menu bar, a toolbar, and a file explorer on the left side.

```
1 "URL", "ContentType", "Content"
2 "http://www.library.ttu.edu/", "application/xhtml+xml", "<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
3 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"><html
4 xmlns="http://www.w3.org/1999/xhtml"><head> <title>Texas Tech University Libraries :: Home</ti
5 http-equiv="Content-Type" content="text/html; charset=utf-8" /> <meta http-equiv="X-UA-Comp
6 content="IE=EmulateIE7"/> <meta name="AUTHOR" content="State of Texas and Texas Tech Univer
7 University Libraries, Technology and Management Services" /> <meta name="KEYWORDS" content="
8 University; Texas Tech University Libraries Intranet; Technology and Management Services" /> <m
9 name="DESCRIPTION" content="Texas Tech University Libraries Technology and Management Services
10 <script type="text/javascript" src="js/DragDrop.js"></script> <!--styles--> <link type="te
11 media="all" rel="stylesheet" href="styles/libstyle.css" /> <link type="text/css" media=
12 rel="stylesheet" href="styles/ttuPrint.css" /> <link rel="shortcut icon"
13 href="styles/images/favicon.ico" /> <!--nav scripts--> <script src="scripts/LeftNav.js"
14 type="text/javascript"></script> <style type="text/css"> .style4 {font-size: 12px} </style>
15 <!-- **ADA LINKS** --> <div id="skipLinks"> skip to: <a href="#pageContent">page content<
16 href="#pageNav">links on this page</a> | <a href="#siteNav">site navigation</a> | <a
17 href="#pageFooter">footer (site information)</a> </div> <div id="container"> <!-- BEGINNIN
18 UPPER NAVIGATION REGION: MASTHEAD * SEARCH FORM FOR WHOLE SITE --> <div id="masthead">
19 href="index.php" id="logo" class="imgswap" title="TTU Libraries Home" name="top">Texas
20 Libraries</a> <form id="search-box" action="http://google.ttu.edu/search" method="get"
21 for="q"><label> <input id="q" type="text" name="q" value="Search Library Website
22 onfocus="this.value=''" /><input type="hidden" name="site" value="Library" /> <input
23 type="hidden" name="client" value="texas_tech" /><input type="hidden" name="proxystylesh
24 value="styles/ttuSearch.css" /> <input type="hidden" name="output" value="xml_no_dt
25 type="hidden" name="restrict" value="" /> <input class="button" type="image" nam
26 src="styles/images/search-arrow.gif" alt="Search" /> </form> <!--<div
27 id="faqcontact"><p><strong><a href="http://library.ttu.edu/help/faq/index.php">F.A.Q.</a> | <
28 href="http://library.ttu.edu/help/email.php">Ask us</a></strong></p></div> ../about/admin/forms
29 --> </div> <!-- LEFT HAND NAVIGATION BAR --> <ul id="MenuBar1" class="MenuBarVertical
30 style="font-size:0.8em;" class="TopLinks" href="http://library.ttu.edu/arch/index.php">ARCH
31 LIBRARY</a> </li> <li> <a style="font-size:0.8em;" class="TopLinks"
32 href="http://swco.ttu.edu/">SOUTHWEST COLLECTION</a> </li> <li> <a class="TopHeadings">S
33 TOOLS</a> </li> <li> <a class="Dotted" href="http://iris.ttu.edu/F">Library Catalog</a>
34 <a class="Arrow" href="http://library.ttu.edu/includes/leftNav/er.php">Electronic Resources</
35 <li> <a class="FODotted" href="http://texastech-ml.hosted.exlibrisgroup.com/V">Find-it
```

Sample web crawl data from the TTU Library website. Shown here in text format.
See below for the same data shown in HEX format.

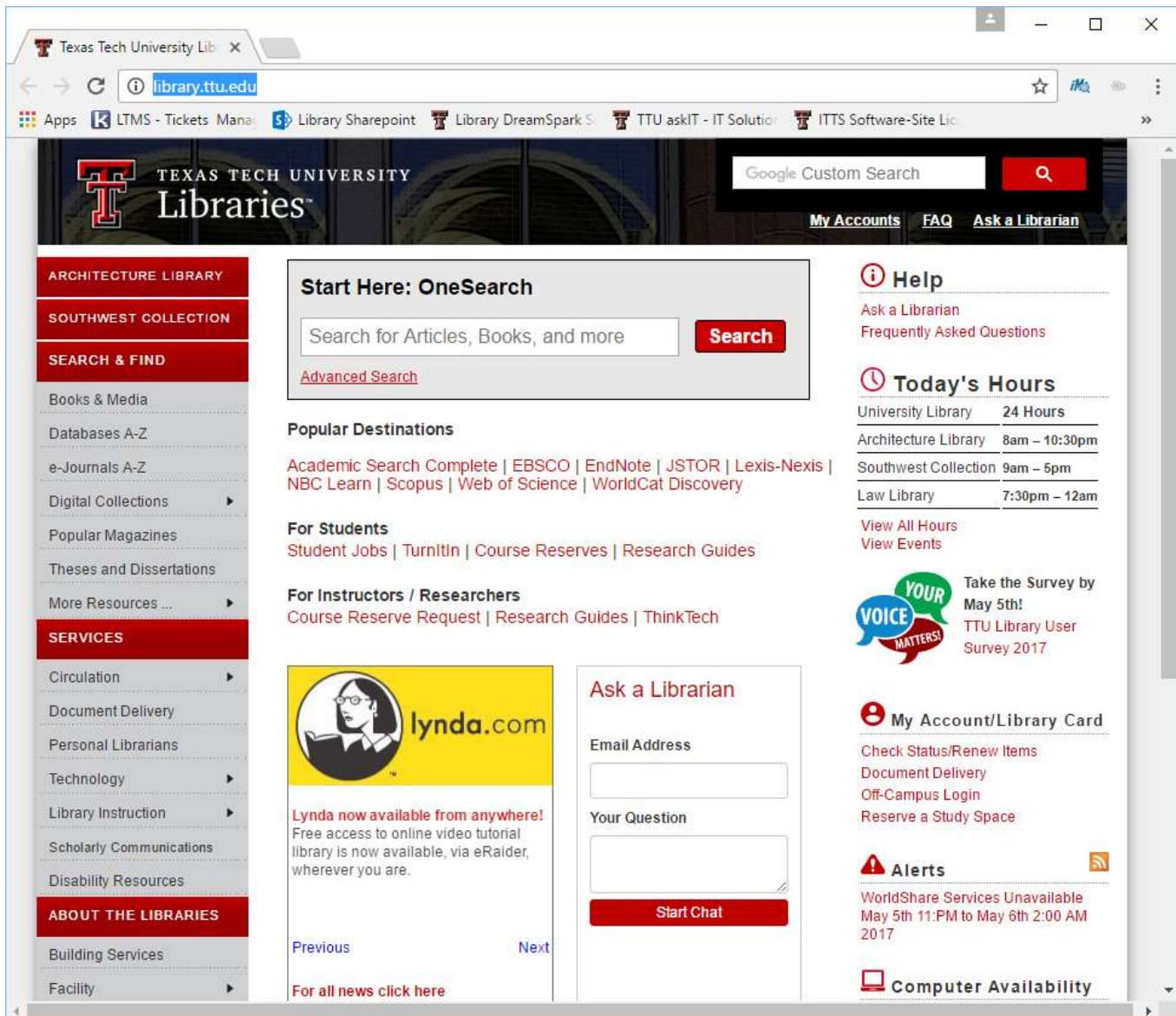
The next page shown below has the same data shown in HEX format so that the actual output can be viewed. This is the data which my Scanner application was to read and process into TOKENS for database insertion and further processing.

Same page as above (<http://www.library.ttu.edu>) shown in HEX format using the output from the Ultraedit software tool.

The data contains numerous tags and formatting. The website <http://www.library.ttu.edu> contains sidebars, header and footers. These are all formatted using html tags and web crawl data contains all such information. The above crawl data starts with the main website 'library.ttu.edu' and then crawls to the other websites contained in main website. Similarly, it keeps on crawling gathering more and more

data. A full web crawl on library.ttu.edu has over 7500 websites. So, there is a lot of data and this data has to be scanned and filtered in order to make use of it.

A typical page from the TTU Library website appears as follows:



<http://www.library.ttu.edu> typical TTU Library home page

The scanner I wrote in Microsoft Visual Studio removes the tags and excess space from the web crawl data.

Code Description

The program consists many modules, which are described in this section. I will describe main function of each module along with input and output of each module.

Main Module

In the main module, the program calls the functions. There are four functions, each performing a specific task to filter the web crawl data.

```
Sub Main()  
    'Main module that calls the functions  
  
    'Function to remove div tags  
    removediv()  
    'Function to separate websites and number them  
    separateWebsites()  
    'Function to remove HTML tags  
    removeHTML()  
    'Function to remove excess space and lines.  
    remove_Space()  
  
End Sub
```

removediv() function

This function removes the div tags from the main crawl data. 'div' tags contain information about the formatting of the webpage. We want to store the man text/content of each website. We discard 'div' tags data because it does not contain the content that we want. The input for this function is the web crawl file. This function generates two output files: withDiv.txt which contains the removed div tags and withoutDiv.txt where div tags are replaced with empty string.

```
Function removediv()  
    'Function to remove div tags from the source file. Div tag contains  
    irrelevant informations so it is removed.
```

'The source file is the main webcrawl file. After removing div tags, the content is stored in withoutDiv.txt file.

'The removed div tags are stored in withDiv.txt to check what contents have been discarded.

```
Dim source_file As New
StreamReader("C:\Users\supoudya\Documents\WebsiteSummary files\source.txt")
Using withDiv As New
StreamWriter("C:\Users\supoudya\Documents\WebsiteSummary files\withDiv.txt")
Using withoutDiv As New
StreamWriter("C:\Users\supoudya\Documents\WebsiteSummary
files\withoutDiv.txt")
Dim str As String
str = source_file.ReadLine
Dim count As Integer = 0
While source_file.Peek <> -1

'Regex expression is used to remove div tags and are
stored in withDiv file.
Dim n As Match = Regex.Match(str, "<div[^>]*>(.*?)</div>")
', RegexOptions.Multiline)

withDiv.WriteLine(n.Value)

'The div tags are replaced with empty string and stored in
withoutDiv.txt file.
str = Regex.Replace(str, "<div[^>]*>(.*?)</div>", "")
withoutDiv.WriteLine(str)

str = source_file.ReadLine()
End While
End Using
End Using
source_file.Close()
Console.WriteLine("Done with removediv function")
Console.ReadLine()
Return 0
End Function
```

Input: source.txt, which is same as the web crawl data mentioned above.

Output 1: withdiv.txt

```
<div id=""skipLinks""> skip to: <a href=""#pageContent"">page content</a> | <a
href=""#pageNav"">links on this page</a> | <a href=""#siteNav"">site navigation</a> |
<a href=""#pageFooter"">footer (site information)</a> </div>
```

```

<div id=""skipLinks"">                skip to: <a href=""#pageContent"">page content</a>
| <a href=""#pageNav"">links on this page</a> | <a href=""#siteNav"">site
navigation</a> | <a href=""#pageFooter"">footer (site information)</a>                </div>
<div id=""skipLinks"">                skip to: <a href=""#pageContent"">page content</a>
| <a href=""#pageNav"">links on this page</a> | <a href=""#siteNav"">site
navigation</a> | <a href=""#pageFooter"">footer (site information)</a>                </div>
<div id=""skipLinks"">                skip to: <a href=""#pageContent"">page content</a>
| <a href=""#pageNav"">links on this page</a> | <a href=""#siteNav"">site
navigation</a> | <a href=""#pageFooter"">footer (site information)</a>                </div>

```

Output 2: withoutdiv.txt

```

"http://www.library.ttu.edu/", "application/xhtml+xml", "<!DOCTYPE html PUBLIC ""-
//W3C//DTD XHTML 1.0 Transitional//EN"" ""http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd""><html xmlns=""http://www.w3.org/1999/xhtml""><head> <title>Texas
Tech University Libraries :: Home</title> <meta http-equiv=""Content-Type""
content=""text/html; charset=utf-8"" /> <meta http-equiv=""X-UA-Compatible""
content=""IE=EmulateIE7""/> <meta name=""AUTHOR"" content=""State of Texas and Texas
Tech University, University Libraries, Technology and Management Services"" /> <meta
name=""KEYWORDS"" content=""Texas Tech University; Texas Tech University Libraries
Intranet; Technology and Management Services"" /> <meta name=""DESCRIPTION""
content=""Texas Tech University Libraries Technology and Management Services Intranet""
/> <script type=""text/javascript"" src=""js/DragDrop.js""></script> <!--styles-->
<link type=""text/css"" media=""all"" rel=""stylesheet"" href=""styles/libstyle.css""
/> <link type=""text/css"" media=""print"" rel=""stylesheet""
href=""styles/ttuPrint.css"" /> <link rel=""shortcut icon""
href=""styles/images/favicon.ico"" />

```

separateWebsites() function

This function separates the websites in the crawl data. It uses line feed character to separate the websites. The function reads first character from the input file and stores it in the array named input. Then it compares the character with the line feed character for all the characters in the input file. When there is a match, a newline is added along with a number which is incremented for each new website. The number gives information about total number of websites in the data. The input for this function is the withoutDiv.txt file. The output file SeparateWebsites.txt contains the separated websites.

```

Function separateWebsites()
    'Function to separate websites
    'The websites are separated based on line feed character and are written on a
    'new
    'file SeparateWebsites.
    'The websites are numbered to know the total number of websites separated.
    Dim websites As New
        StreamReader
        ("C:\Users\supoudya\Documents\WebsiteSummaryfiles\withoutDiv.txt")
        Using outputFile As New StreamWriter
            ("C:\Users\supoudya\Documents\WebsiteSummary files\SeparateWebsites.txt")
            Dim count As Integer = 0

            'Initializing the character array input
            Dim input(290000000) As Char

            input(count) = Convert.ToChar(websites.Read())

            Dim i As Integer = 0, j As Integer = 1, k As Integer = 0, l As Integer = 0

            'Using StreamReader's method 'Peek' to loop till the end of the file
            While websites.Peek <> -1
                'Reading each character from 'websites' and storing it in input
                input(count) = Convert.ToChar(websites.Read())
                'input(count) = Regex.Replace(Convert.ToString(input(count)), "")
                'Comparing to the line feed character and separating each website
                If String.Compare(input(count), Chr(10)) = 0 Then
                    outputFile.WriteLine(vbCrLf & "NUMBER{0}", j)
                    j = j + 1
                    'Writing the contents of each website in SeparateWebsites.txt
                    While k < count
                        outputFile.Write(input(k))
                        k = k + 1
                    End While

                    End If

                    'Increasing the index of the array input
                    count += 1

                End While

            End Using

            Console.WriteLine("Done with separateWebsites function")
            Console.ReadLine()

```

Return 0
End Function

Output : **SeparateWebsites.txt**

NUMBER1

I added the text 'NUMBER' with a counter to keep track of the number of websites in the crawl data.

```
"http://library.ttu.edu/about/admin/disability_resources.php","application/xhtml+xml", "<
!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
""http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd""><html
xmlns=""http://www.w3.org/1999/xhtml""><head>      <title>Texas Tech University
Libraries :: TTU Libraries' Disability Resources</title>  <meta http-equiv=""Content-
Type"" content=""text/html; charset=utf-8"" /><meta http-equiv=""X-UA-Compatible""
content=""IE=EmulateIE7""/>  <meta name=""AUTHOR"" content=""State of Texas and Texas
Tech University, University Libraries, Technology and Management Services"" />  <meta
name=""KEYWORDS"" content=""Texas Tech University; Texas Tech University Libraries
Intranet; Technology and Management Services"" />  <meta name=""DESCRIPTION""
content=""Texas Tech University Libraries"" />
```

NUMBER2

```
"http://library.ttu.edu/about/admin/forms.php","application/xhtml+xml", "<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
""http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd""><html
xmlns=""http://www.w3.org/1999/xhtml""><head>      <title>Texas Tech University
Libraries :: Forms</title>  <meta http-equiv=""Content-Type"" content=""text/html;
charset=utf-8"" /><meta http-equiv=""X-UA-Compatible"" content=""IE=EmulateIE7""/>
      <meta name=""AUTHOR"" content=""State of Texas and Texas Tech University,
University Libraries, Technology and Management Services"" />  <meta
name=""KEYWORDS"" content=""Texas Tech University; Texas Tech University Libraries
Intranet; Technology and Management Services"" />  <meta name=""DESCRIPTION""
content=""Texas Tech University Libraries"" /><!--styles-->      <link
type=""text/css"" media=""all"" rel=""stylesheet"" href="".././styles/libstyle.css""
/>
      <link type=""text/css"" media=""print"" rel=""stylesheet""
href="".././styles/ttuPrint.css"" />      <link rel=""shortcut icon""
href="".././styles/images/favicon.ico"" />      <!--[if lte IE 6]>      <link
rel=""stylesheet"" type=""text/css"" href="".././styles/ie.css"" />
      <![endif]><!--nav scripts-->      <script
src="".././scripts/LeftNav.js"" type=""text/javascript""></script>      <script
src="".././scripts/TopNav.js"" type=""text/javascript""></script>      </head>
      <!-- **ADA LINKS** -->
```

removeHTML() function

This function removes the html tags from the data and replaces it with an empty string. The input file used in this function is the output file generated by separateWebsites() function, SeparateWebsites.txt. It uses regular expressions to replace html tags. Along with the tags, this function also replaces the special symbols and parentheses with empty string. The output file generated by this function is WithoutTags.txt.

```
Function removeHTML()
    'This function reads from SeparateWebsites.txt file and removes tags and
    special symbols that are not relevant to the content of the file.
    'It uses regex to replace the tags and symbols with empty string and stores the
    content without HTML tags in WithoutTags.txt file.

    Dim RemovedTags As New StreamReader("C:\Users\supoudya\Documents\WebsiteSummary
    files\SeparateWebsites.txt")
    Using withoutTags As New
    StreamWriter("C:\Users\supoudya\Documents\WebsiteSummary files\WithoutTags.txt")
        Dim str As String

        'Using regex to remove the html tags and special symbols and replacing it
        with empty string
        str = RemovedTags.ReadLine()
        While RemovedTags.Peek <> -1
            str = Regex.Replace(str, "<[^\>]*\>", "")
            'str = Regex.Replace(str, "/*/", "")
            str = Regex.Replace(str, "-.*?>", "")
            str = Regex.Replace(str, " [ ] ", "")
            str = Regex.Replace(str, "{.*?}", "")
            str = Regex.Replace(str, "\"\"\"", "")
            str = Regex.Replace(str, "\|", "")
            str = Regex.Replace(str, "\\", "")
            str = Regex.Replace(str, ";", "")
            'str = Regex.Replace(str, "\.", "")
            'str = Regex.Replace(str, "\:", "")
            str = Regex.Replace(str, "\(", "")
            str = Regex.Replace(str, "\?", "")
            str = Regex.Replace(str, "\+", "")
            str = Regex.Replace(str, "\=", "")
            str = Regex.Replace(str, "\%", "")
            str = Regex.Replace(str, "\&", "")
            str = Regex.Replace(str, "\}", "")
            str = Regex.Replace(str, "\(..*?\)", "")
            withoutTags.WriteLine(str)
            str = RemovedTags.ReadLine()
        End While

        RemovedTags.Close()
    End Using
    Console.WriteLine("Done with removeHTML function")
End Function
```



```

        Console.ReadLine()
        Return 0

End Function

```

Output: WithoutTags.txt

NUMBER1

```

http://www.library.ttu.edu/,application/xhtml+xml, Texas Tech University Libraries ::
Home .style4 nbspnbspFor all news click here
Connect: nbsp var gaJsHost https: document.location.protocol https://ssl. :
http://www.document.writeunescape3Cscript src' gaJsHost google-analytics.com/ga.js'
type'text/javascript'3E3C/script3Etry catcherr Last updated: June 11, 2013

```

NUMBER2

```

http://library.ttu.edu/about/admin/disability_resources.php,application/xhtml+xml,Texas
Tech University Libraries :: TTU Libraries' Disability Resources

```

```

var gaJsHost https: document.location.protocol
https://ssl. : http://www.document.writeunescape3Cscript src' gaJsHost google-
analytics.com/ga.js' type'text/javascript'3E3C/script3Etry catcherr

```

remove Space() function

This function removes the excess space and empty lines from the input file. The input file for this function is the file generated by removeHTML function, WithoutTags.txt. We have removed the div tags, html tags and special symbols from the main file and replaced them with empty string. As a result, there are lots of empty space in the file which needs to be managed. This is the main job of this function. The output generated by this function is written on the output file WithoutExcessSpace.txt.

```

Function remove_Space()
    'The crawl data contains a lot of excess space and blank lines. This function
    removes the space and empty lines from the file
    'and stores it in WithoutExcessSpace.txt file
    Dim RemoveSpace As New StreamReader("C:\Users\supoudya\Documents\WebsiteSummary
    files\WithoutTags.txt")

```

```

Using space As New StreamWriter("C:\Users\supoudya\Documents\WebsiteSummary
files\WithoutExcessSpace.txt")
    Dim str4 As String, str9 As String = ""

    str4 = Convert.ToString(RemoveSpace.ReadLine)

    Dim a As Integer = 0, b As Integer = 0

    While RemoveSpace.Peek <> -1

        If (str4.Trim <> "") Then

            'Replace the new line and empty space with empty string.

            str9 = Regex.Replace(str4, "([\r\n])\s", " ")
            str9 = Regex.Replace(str9, "[\t]+", " ")

            space.WriteLine(str9.Trim)
            a = a + 1
        End If

        str4 = Convert.ToString(RemoveSpace.ReadLine)
    End While
End Using

RemoveSpace.Close()

Console.WriteLine("Done with remove_Space function")
Console.ReadLine()
Return 0
End Function

```

Output: WithoutExcessSpace.txt

NUMBER1

http://www.library.ttu.edu/,application/xhtml+xml, Texas Tech University Libraries ::
Home .style4 nbspnbspFor all news click here Connect: nbsp var gaJsHost https:
document.location.protocol https://ssl. : http://www.document.writeunescape3Cscript
src' gaJsHost google-analytics.com/ga.js' type'text/javascript'3E3C/script3Etry
catcherr Last updated: June 11, 2013

NUMBER2

http://library.ttu.edu/about/admin/disablity_resources.php,application/xhtml+xml, Texas
Tech University Libraries :: TTU Libraries' Disability Resources var gaJsHost https:
document.location.protocol https://ssl. : http://www.document.writeunescape3Cscript
src' gaJsHost google-analytics.com/ga.js' type'text/javascript'3E3C/script3Etry
catcherr

separateComma() function

This function separates the text based on comma. The text after comma is written in the next line. This is the first process to format our text. We want one word on each line to make it easier to compare the words with the stop words and with the synonym database. The input for the file is WithoutExcessSpace.txt. It is the file generated by remove_Space function. The output generated by this function is stored in the file CommaSeparated.txt.

```
Function separateComma()  
    'The crawl data is not structured. This function separated the data based on  
    comma.  
    Dim input As New StreamReader("C:\Users\supoudya\Documents\WebsiteSummary  
files\WithoutExcessSpace.txt")  
    Using commaSep As New StreamWriter("C:\Users\supoudya\Documents\WebsiteSummary  
files\CommaSeparated.txt")  
        Dim comma As String, comma1() As String  
        Dim context As String = ""  
        Dim i As Integer = 0  
        comma = Convert.ToString(input.ReadLine)  
  
        While input.Peek <> -1  
            comma1 = comma.Split(",")  
            For Each n In comma1  
                commaSep.WriteLine(n)  
            Next  
            comma = Convert.ToString(input.ReadLine)  
        End While  
    End Using  
    input.Close()  
    Console.WriteLine("Done with separateComma() function")  
    Console.ReadLine()  
    Return 0  
End Function
```

Output: CommaSeparated.txt

NUMBER1

<http://www.library.ttu.edu/>

```

application/xhtml+xml
Texas Tech University Libraries :: Home .style4 nbspnbspFor all news click here
Connect: nbsp var gaJsHost https: document.location.protocol https://ssl. :
http://www.document.writeunescape3Cscript src' gaJsHost google-analytics.com/ga.js'
type'text/javascript'3E3C/script3Etry catcherr Last updated: June 11
2013

```

NUMBER2

```

http://library.ttu.edu/about/admin/disablity_resources.php
application/xhtml+xml
Texas Tech University Libraries :: TTU Libraries' Disability Resources var gaJsHost
https: document.location.protocol https://ssl. :
http://www.document.writeunescape3Cscript src' gaJsHost google-analytics.com/ga.js'
type'text/javascript'3E3C/script3Etry catcher

```

removeStopWords() function

This function removes the repetitive words from the file. Some of the stop words are:

a's
able
about
above
according
accordingly
across
actually
after
Afterwards
became
because
become
becomes
becoming
been
before
Beforehand
c
c'mon
c's
came
can
can't
cannot
cant
cause
causes etc.

These words are stored in a file called 'stopwords_en'. Each line contains one words. The function reads each word from our crawl data and compares it to the words in the file. If there is a match, the words in the crawl data are replaced with empty string. Words and symbols which is not required in crawl data can be removed by adding it in the 'stopwords_en'. This function also separates the crawl data according to space. So, each word after space is written in new line. This again is done to make it easier to compare the words with synonym database. The input for this function is the file CommaSeparated.txt. This function generates three output file: SpaceSeparated.txt where the text is separated according to space, RemovedWords.txt which contains the removed stop words and WithoutStopWords.txt which contains the crawl data without the stop words.

```
Function removeStopWords()
    'The crawl data contains words like the, on, in, is etc. which are not
    significant. This function removes such words.
    'It reads a text file containing stopwords on each line and compares it each
    word in crawl data.
    'If it is a match then it replaces it with empty string and stores the replaced
    words in RemovedWords.txt
    Dim count As Integer = 0
    Dim str() As String, str1 As String, str2 As String
    Dim input As New StreamReader("C:\Users\supoudya\Documents\WebsiteSummary
    files\CommaSeparated.txt")
    Using spaceSep As New StreamWriter("C:\Users\supoudya\Documents\WebsiteSummary
    files\SpaceSeparated.txt")
        Using discard As New
        StreamWriter("C:\Users\supoudya\Documents\WebsiteSummary files\RemovedWords.txt")
            Using withoutStopWords As New
            StreamWriter("C:\Users\supoudya\Documents\WebsiteSummary files\WithoutStopWords.txt")
                'Storing the contents of the text file in str1 as a string
                str1 = Convert.ToString(input.ReadLine())

                Dim inLoop As Integer = 0, newline As Integer = 1

                While input.Peek <> -1
                    str = str1.Split(" ")
                    For Each substring In str
                        'Declaring a new stream reader inside the loop to match
                        stopwords to each string in str1
                        Dim stopWord As New
                        StreamReader("C:\Users\supoudya\Documents\WebsiteSummary files\stopwords_en.txt")
                        str2 = Convert.ToString(stopWord.ReadLine())
                        'Reading from names2 until the end of file
                        While stopWord.Peek <> -1
                            'Comparing the substring to the stopwords in the str2
```

```

        If String.Compare(substring.ToLower, str2) = 0 Then
            inLoop += 1
            discard.WriteLine(substring)
            substring = substring.Replace(substring, "")
        End If

        'Reading the next stopword
        str2 = Convert.ToString(stopWord.ReadLine())
        newline += 1
    End While

    count += 1
    'Removing the formatting of .pdf files
    substring = Regex.Replace(substring, "^JVBER.*$", "")

    spaceSep.WriteLine(substring.Trim)
    withoutStopWords.Write(substring & " ")
    If Regex.IsMatch(substring, "^NUMB.*") Then
        withoutStopWords.WriteLine()
    End If

    'Closing the streamreader stopWord
    stopWord.Close()

Next

    str1 = Convert.ToString(input.ReadLine())
End While
End Using
End Using
End Using
input.Close()
Console.WriteLine("Done with removeStopWords() function")
Console.ReadLine()
Return 0
End Function

```

Output: SpaceSepatrated.txt

NUMBER1

<http://www.library.ttu.edu/>

Texas
Tech
University
Libraries

When a space is encountered, the words are added to new line. We are formatting the data in this way to make it easier to compare it with the synonym database.

Resources

gaJsHost

document.location.protocol
https://ssl.

http://www.document.writeunescape3Cscript

gaJsHost

type'text/javascript'3E3C/script3Etry

Output: WithoutStopWords:

NUMBER1

http://www.library.ttu.edu/ Texas Tech University Libraries Home nbspnbspFor
news click Connect: gaJsHost document.location.protocol https://ssl.

http://www.document.writeunescape3Cscript gaJsHost

type'text/javascript'3E3C/script3Etry updated: June 11 2013

NUMBER2

http://library.ttu.edu/about/admin/disability_resources.php Texas Tech University
Libraries TTU Libraries' Disability Resources gaJsHost

document.location.protocol https://ssl.

http://www.document.writeunescape3Cscript gaJsHost

type'text/javascript'3E3C/script3Etry

Output: RemovedWords.txt

application/xhtml+xml

::

.style4

all

here

nbsp

var

https:

:

src'

google-analytics.com/ga.js'

catcherr

Last


```
application/xhtml+xml
::
```

formatText() function

This function formats the crawl data so that it can be used with a database software. The data is formatted such that each line contains a URL, a word and the date and time. Regex 'https?:\/\/(www\.)?[-a-zA-Z0-9@:%_\+~#={2,256}\. [a-z]{2,6}\b([-a-zA-Z0-9@:%_\+~#?&//=]*)' is used to match the URL. This function takes SpaceSeparated.txt as input and generates output in the file FormattedText.txt.

```
Function formatText()
    'This function formats the text in the format: URL Word Date

    Dim input As New StreamReader("C:\Users\supoudya\Documents\WebsiteSummary
files\SpaceSeparated.txt")
    Dim date1 As Date = Now
    Using format As New StreamWriter("C:\Users\supoudya\Documents\WebsiteSummary
files\FormattedText.txt")
        Dim str As String

        str = Convert.ToString(input.ReadLine)
        'Regex to match URL
        Dim url As String = "https?:\/\/(www\.)?[-a-zA-Z0-9@:%_\+~#={2,256}\. [a-
z]{2,6}\b([-a-zA-Z0-9@:%_\+~#?&//=]*)"
        Dim savedURL As String = ""
        Dim i As Integer = 0
        While input.Peek <> -1

            For Each n As Match In Regex.Matches(str, url)
                savedURL = n.Value

            Next
            If Regex.IsMatch(str, "NUMBER") Then
                savedURL = ""

            End If
            If (str.Trim <> "") Then
                If Regex.IsMatch(str, savedURL) = 0 Then
                    format.WriteLine("{0} {1}{1,10} {2} {3}{3,10} {4} ", savedURL,
vbTab, str, vbTab, date1)
```

```

        End If
    End If
    str = Convert.ToString(input.ReadLine)
End While
End Using
input.Close()
Console.WriteLine("Done with formatText() function")
Console.ReadLine()
Return 0
End Function

```

Output: FormattedText.txt:

Format of the text: The URL, text that is contained in the URL and, time and date of the crawl.

http://www.library.ttu.edu/	Texas	3/16/2017 4:05:51 PM
http://www.library.ttu.edu/	Tech	3/16/2017 4:05:51 PM
http://www.library.ttu.edu/	University	3/16/2017 4:05:51 PM
http://www.library.ttu.edu/	Libraries	3/16/2017 4:05:51 PM
http://www.library.ttu.edu/	Home	3/16/2017 4:05:51 PM
http://www.library.ttu.edu/	nbspnbspnbspnbspnbspFor	3/16/2017 4:05:51 PM
http://www.library.ttu.edu/	news	3/16/2017 4:05:51 PM
http://www.library.ttu.edu/	click	3/16/2017 4:05:51 PM
http://www.library.ttu.edu/	Connect:	3/16/2017 4:05:51 PM
http://www.library.ttu.edu/	gaJsHost	3/16/2017 4:05:51 PM
http://www.library.ttu.edu/	document.location.protocol	3/16/2017
4:05:51 PM		
http://www.library.ttu.edu/	https://ssl.	3/16/2017 4:05:51 PM
http://www.library.ttu.edu/	http://www.document.writeunescape3Cscript	
3/16/2017 4:05:51 PM		
http://www.library.ttu.edu/	gaJsHost	3/16/2017 4:05:51 PM
http://www.library.ttu.edu/	type'text/javascript'3E3C/script3Etry	
3/16/2017 4:05:51 PM		
http://www.library.ttu.edu/	updated:	3/16/2017 4:05:51 PM
http://www.library.ttu.edu/	June	3/16/2017 4:05:51 PM
http://www.library.ttu.edu/	11	3/16/2017 4:05:51 PM

searchText() function

This function searches for a word in the crawl data. It returns the context of the word as well as the URL of the website where the word occurs. The input for this function is the file WithoutStopWords.txt and the output is generated on the file Searched.txt.

```
Function searchText()
    'This function searches for a word in the FormattedText.txt file and writes the
    output in Searched.txt in the format:
    'URL 'Context words' 'Searched Words' 'Context words'

    Dim searchInStopword As New
    StreamReader("C:\Users\supoudya\Documents\WebsiteSummary files\WithoutStopWords.txt")

    Dim str As String, savedUrl1 As String = "", contextWord As String = ""
    Dim index As Integer
    str = Convert.ToString(searchInStopword.ReadLine())
    Dim pattern As String = "catalog"
    Dim url1 As String = "https?:\|\/(www\.)?[-a-zA-Z0-9@:%_\+~#={2,256}\. [a-
z]{2,6}\b([-a-zA-Z0-9@:%_\+~#?&/=]*)"
    Dim context2 As String = "(?:[a-zA-Z'-]+[^\a-zA-Z'-]+){0,3}" + pattern +
    "(?:[^\a-zA-Z'-]+[a-zA-Z'-]+){0,3}"
    Dim count2 As Integer = 0, count3 As Integer = 0
    Using search As New StreamWriter("C:\Users\supoudya\Documents\WebsiteSummary
files\Searched.txt")

        While searchInStopword.Peek <> -1
            For Each n As Match In Regex.Matches(str, url1)
                savedUrl1 = n.Value
            Next

            For Each m As Match In Regex.Matches(str.ToLower, pattern)
                count3 += 1
                index = m.Index
            Next

            For Each l As Match In Regex.Matches(str.ToLower, context2)
                contextWord = l.Value
                search.WriteLine(" {0}{1}{1,10} {3}", savedUrl1, vbTab, index,
contextWord)

                Next
                count2 += 1
                str = Convert.ToString(searchInStopword.ReadLine())

            End While
            search.WriteLine(vbLf + "The word " + pattern + " is used {0} times",
count3)
        End Using
        searchInStopword.Close()

        Console.WriteLine("Press any key to exit.")
        Console.ReadKey()
```

Return 0
End Function

Output: Searched.txt where the searched word is 'catalog'.

Format: The URL that contains the searched term, the searched term(underlined) with its context. The context is set as three words before and after the searched term.

http://library.ttu.edu/services/technology/soft.php	svg viewer	arcgis 10 <u>catalog</u>	arcgis 10 globe
arcgis			
http://library.ttu.edu/turnitin/turnitin.php	collection search tools:	<u>catalog</u>	electronic resourcesfind study
http://www.library.ttu.edu/articles.php	collection search tools:	<u>catalog</u>	electronic resourcesfind study
http://www.library.ttu.edu/changes.php	collection search tools:	<u>catalog</u>	electronic resourcesfind study
http://www.library.ttu.edu/librarians.php	collection search tools:	<u>catalog</u>	electronic resourcesfind study
http://www.library.ttu.edu/reserves.php	collection search tools:	<u>catalog</u>	electronic resourcesfind study
http://library.ttu.edu/computeravailability/availability.php	collection search tools:	<u>catalog</u>	electronic
resourcesfind study			
http://library.ttu.edu/ithenticate/ithenticate.php	collection search tools:	<u>catalog</u>	electronic
resourcesfind study			
http://library.ttu.edu/news/stories/index.php	collection search tools:	<u>catalog</u>	electronic resourcesfind study
http://www.librarything.com/	date.gettime]librarything	<u>catalog</u>	books onlinevar isbndb
http://ttulawlibrary.worldcat.org/advancedsearch	law institute	cassidy <u>catalog</u>	
http://ttulawlibrary.worldcat.org/advancedsearch	agency documents	cassidy <u>catalog</u>	

Challenges faced while writing the scanner

The main challenge that I faced while writing this program was the use of regular expressions. I use regular expressions in my program to remove tags and symbols. I also use it to find the URL in the data and the context of data. Having never used regular expressions before, I had difficulty constructing the regular expressions for my program. It took quite a bit of research and, trials and errors to figure out the regular expressions required for the program.

Another problem that I faced while writing the scanner program was the computation of large data. Web crawl returns a large amount of data and initially I had written code which would read the whole data and then process it. This turned out to be inefficient and I ran out memory while processing the complete web crawl data. I modified my program so that it would read single line instead of whole data.

3. Things Learned

I was a junior when I started to work on this project. I had some experience programming but I did not have any experience of working on big project like website summary. I had to learn and research about the tools required for a task at hand before working on it. The first thing I learned was Microsoft Visual Basic. I followed the tutorial from https://mva.microsoft.com/en-us/training-courses/vb-fundamentals-for-absolute-beginners-8297?l=3THjWMYy_4904984382. It consisted 25 episodes and was informative. I also learned Linux commands to set up the cluster. All the servers in the cluster are set up with Red Hat 6.

I also learned about MS Access. I had no familiarity with any database systems and it was a good opportunity to learn. Our main goal is to generate a report writer which involves manipulating the data in the database. I learned ADODB programming so that I could write code in Visual Studio to manipulate data in MS Access. ADODB stands for ActiveX Data Objects which allows a developer to write programs that access data without knowing how the database is implemented. Learning ADODB programming was difficult because there is no proper documentation on it. I looked at few books but had a very little information about it. With the help of my supervisor, faculty member of R&D group, I wrote ADO code to connect to MS Access through Visual Studio.

4. Connection to Academic Study

I had basic programming knowledge when I started this internship. I was familiar with Java, Microsoft Visual Basic and C. The scanner program that I wrote was in Microsoft Visual Basic. It helped me get a head start because I was familiar with it. I still had to learn more about it but it was quite easy as I didn't have to start from ground

zero. The initial program that I wrote consisted three different files. In order to completely run the scanner program and get the output, three files had to be compiled one after another. It did the job but was difficult to read and maintain. The program was not commented as well. I had learned in my data structures class and in my previous programming classes that a program must be easy to read and maintain. Initially, I was only concerned with completing the code but after realizing that my program would be difficult to maintain, I split the program into functions with each function doing a specific task. Then, I combined all the function in a single file.

Another issue I had was that my initial program ran out of memory when the crawl data was large. I was reading the whole crawl data, then was filtering the whole data at once using regular expressions. As crawl data could get very large, I had to change my approach. I knew about the storage of data in memory from my assembly course. I could visualize how memory could be all used up when the whole crawl data is read. I then changed my approach so that I would read the crawl file one line at a time. After reading a line, I would read each words then I would use regular expression on each word to filter them. This reduced the memory usage as well as ensured that it would work with large crawl data.

I also got the opportunity to learn about Raspberry Pi during my internship. I created a security camera using a Raspberry Pi 3 and a camera module. I installed Motion OS on the Raspberry Pi 3 which would take a picture whenever it detected a motion. It got me interested in the Internet of Things and computer networks. There are many applications of Raspberry Pi and has been used to create many smart devices. I decided to take course a computer networks as an elective to learn more about the connectivity between devices and it was mainly because I got a chance to be familiar with it during my internship. Besides Raspberry Pi, I researched about Natural Language Processing Toolkit for Python during the internship. I read about how natural language is processed and how is understood by a machine. The automated call service, spell check feature etc. were all applications of natural language processing. It was interesting and it introduced me to the field of machine learning. I ended up doing an independent study on Machine Learning on spring 2017 semester because I was exposed to it during my internship.

While learning about how web crawl data is generated and learning to use IBM BigInsights, I came across Big Data. I had heard about it a little before but didn't know much about it. My supervisor suggested me to look into Big Data as it was growing very popular and was big on the market. I did look into it and found it to be an exciting field. I particularly got interested in the field of data science where interesting insights could be gained from data. I did more research about it and found that Data Scientist are in demand and the field is growing rapidly. I had plans to join graduate school but was unsure about the topic to choose. Learning about data science through this internship has provided me with a topic which could be my main area of focus in graduate school.

5. Conclusion

I participated in the development and testing of a system which will eventually become a real product. I experienced working with problems for which there was no clearly documented solution and we had to experiment by trial and error, sometimes taking guesses to find the right solution. We experienced numerous undocumented errors for which there were no posts on forums or in help tools. The experiences we had will be posted in a forum so that other users can benefit from our efforts. I got to see practical problems and research writings about the same issues, and this opened up some areas that I have not had the opportunity to see previously. I learned the importance of hardware knowledge and saw how hardware configuration greatly affects software development. I got to work with a complete **solution** and saw more detail than is generally possible in my previous projects.

Attachment: Research & Development Summer Project Internship 2016 (position description)



Research & Development Summer Project Internship 2016

19 hours per week maximum

\$10/hour

Nonbenefits eligible

Exclusive position – student cannot have another on-campus job

Expectation:

A creative student who works effectively in a Research Environment, shows a strong interest in text processing and analysis, and has a career interest in application programming and data analysis with a desire to obtain professional experience.

Job Responsibilities:

- Use web tools to create data sets for analysis.
- Extract content data from textual documents, create metadata.
- Design database for creation of text analyses, manage loading and testing of database
- Build reporting tools that support semantic text analysis
- Manage file system documents.
- Create user manual
- Research Natural Language Processing systems for use in textual analysis

Preferred Skills:

Text processing experience using Microsoft Visual Studio and programming experience in Microsoft Visual Basic and Visual Basic for Applications.

Demonstrated database design skills.

Strong English language skills (grammar, syntax, punctuation) and knowledge of language tools such as thesauri, semantic dictionaries, dictionaries of acronyms, punctuation styles.

Interest in Natural Language Processing and Data Mining; programming knowledge in Python and C++.

Database report writer experience.

To Apply:

Send cover letter and resume to Jim Brewer (jim.brewer@ttu.edu) by May 13, 2016.