Chapter 2

Datasets

2.1 Preamble

In this chapter, we briefly describe about our datasets along with the process that we have carried out for creating the dataset and also about the benchmark dataset creation tools. We explain different types of URLs captured. And rest of the chapter includes documents with constrained and unconstrained data.

Any newly designed model has to be experimentally tested to verify the robustness of the developed system. In order to conduct extensive experiments, what is really required is an existing system. Literature survey reveals that existing systems are designed to extract only HTML URLs that too based on only particular Web pages links. The existing system does not extract the entire URLs of given Web domain. A few of the crawlers extract entire URLs of given domain but they fail to check redundancy, extraction of other domain link which are present in the given domain Web pages and also failed to classify particular URLs. In Literature, available URL crawlers are either designed with more number of constraints which are of no use as far as our objective is concerned. Therefore in view of this, we have created a large collection of dataset during the course of this research work. Here, we describe about the databases used in our

_Some parts of the material in this Chapter has appeared in the following research article_

experiments and the corresponding training and testing procedures. In most of our experiments, we have done comparative analysis of the results obtained with developed system (Our own dataset created in view of this research work). All our experiments are carried out on a machine with Intel Core2Duo processor with a speed of 2.7GHz and 4GB RAM memory, JAVA programming language under windows platform.

2.2 XML URL Dataset Creation

W3 (World Wide Web) consortium stated that, HTML has a lot of drawbacks such as limited defined tags, not case sensitive, semi-structured and designed for only to display data with limited options. To overcome these difficulties, a few technologies have been introduced such as XML, Flash (with good design options) and so on [14]. Therefore, Web developers started to migrate to develop Web pages on these kinds of emerging Web Technologies to provide a better description of semantic structure of the Web page contents. Therefore, these days we can see more Web pages on Web which are developed using XML and Flash technologies [16]. This has led to opening of many research avenues on these new technologies. To pursue research initially we need a Data Set (URL/Web page Collections). Getting .xml extension based URLs is very difficult task because if we perform string based search in search engines such as Google, Yahoo etc., (Ex: based on ‘.xml’ query) it will provide us the content based related results. We need semantic structure related results and also existing XML URLs will not be stored in any servers to fetch directly as Benchmark dataset. Therefore, we need a new system/method to perform the task and fulfill the requirements. Based on this requirement(s), we have introduced a system (crawler) to extract all URLs of given Web domain(s) and classify XML URLs out from extracted URLs. The proposed algorithm extracts URLs effectively and classifies XML URLs efficiently. The proposed system fails to extract Flash based Web pages, because Flash Web pages/URLs do not contain or any extensions
Datasets

such as XML, HTML Web pages. Therefore, for Flash Web page dataset, we have performed manual analysis based on semantic orientation. Here we have collected around 800 Flash Web URLs. In Section 2.6, Table 2.3 depicts various types of Flash Web pages which we have collected.

2.2.1 Preprocessing for Dataset Creation

The Figure 2.1 depicts the overall architecture of the proposed system. Here, we planned to extract all URLs of a given Website and classify the XML URLs out of it. In our proposed system, in the very first step, around 120 million Web Domains (.net, .org, .com, .info, .us, .biz and .sk) are downloaded from Web (www.premiumdrops.com) and are dumped into DB for example www.brainbench.com, www.hollywoodauditions.org and so on. Then, they are fetched one by one and given as input to the URL Extractor. URL Extractor first extracts URLs of main page of a given Web Domain (Web site), and then it works recursively to retrieve entire URLs of Website by getting input of extracted URLs.

Finally, by using string based search, XML URLs are classified and stored into a separate array. Algorithm recursively continues for the next elements from DB, in case of no element is found in DB algorithm will terminate. The time complexity of this searching phase is O(mn), where m is number of Web sites and n is number of URLs. We have extracted more than 5000 XML URLs by using the proposed method. Table 2.1 depicts the statistical reports of total Web domains downloaded, size of each file and domains used to extract XML URLs.

![Figure 2.1: Architecture of the proposed system](image-url)
Table 2.1: Statistics of Web domains

<table>
<thead>
<tr>
<th>Sl. No</th>
<th>Web Domains</th>
<th>Count</th>
<th>File Size</th>
<th>Used Domains</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>.com</td>
<td>1,13,359,058</td>
<td>1.2GB</td>
<td>12,000</td>
</tr>
<tr>
<td>2</td>
<td>.net</td>
<td>15,252,879</td>
<td>125MB</td>
<td>12,000</td>
</tr>
<tr>
<td>3</td>
<td>.org</td>
<td>10,461,938</td>
<td>92MB</td>
<td>12,000</td>
</tr>
<tr>
<td>4</td>
<td>.info</td>
<td>5,735,255</td>
<td>98MB</td>
<td>10,000</td>
</tr>
<tr>
<td>5</td>
<td>.us</td>
<td>1,834,025</td>
<td>25MB</td>
<td>10,000</td>
</tr>
<tr>
<td>6</td>
<td>.biz</td>
<td>109,127</td>
<td>16MB</td>
<td>5,000</td>
</tr>
<tr>
<td>7</td>
<td>.sk</td>
<td>243,592</td>
<td>13MB</td>
<td>5,000</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>149,545,831</td>
<td></td>
<td>66,000</td>
</tr>
</tbody>
</table>

2.2.2 Flow of the Proposed Model for Dataset Creation

Figure 2.3 clearly depicts the flow of proposed architecture. First, we retrieve the Web Domain name from DB and then input it into Link Extractor (URL Crawler). The Link Extractor gives existing URLs of home page of Web Domain and stores it into a stack called ‘M’.

In second step, sub URL Crawler read URL one by one from ‘M’ meanwhile it check the condition whether the end of an stack (‘M’) or not, if not it proceeds. Otherwise for true case, system will give read URL as input into sub URL Crawler, it extracts all URLs from current Web page and store them into a stack called ‘m’. Now by using sequential search method check for existence in ‘M’ for each element of ‘m’. Append the URL to ‘M’ in case of not existence, otherwise move to next element (URL) until the end of stack ‘m’.

In general extracted i\textsuperscript{th} URL from ‘m’ is compared with all URLs of ‘M’. If any redundant link is noted, then search will move to ‘i+1’\textsuperscript{th} location element, otherwise URL will be appended to the stack M. This process will be continued until ‘i’ reaches to end of a stack ‘m’. The above process continues for all element of the array M. The Figure 2.2 shows the entire flow of the proposed system.
Datasets

Figure 2.2: Flow of the proposed system
After that by using string based search method XML URLs are identified by searching ‘.xml’ string on each element (URL) of ‘M’ and the URLs found are stored into another stack called ‘X’.

Suppose string has not been found search will move to next element. Once this process ends, algorithm will move back to read next element from stack ‘M’ and do the same process till the end of stack ‘M’. From beginning, algorithm will work for each Web Domain from DB. Therefore, finally we have got list of existing XML URLs of retrieved Web Domain from DB. Algorithm will end when retrieving Web Domain reaches to an end of DB.

Figure 2.3: Architecture of proposed Crawler
2.2.3 Algorithm of the Proposed Method

Input: Web Domain $D_i$

Output: Existing XML URLs

Other Variables: $M$, $m$, $X$

<table>
<thead>
<tr>
<th>Extract the input Web Domain ($D$) from DB</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M = $ Extract URLs of first page of Website</td>
</tr>
<tr>
<td>for $i=1$ to end of $M$ begin</td>
</tr>
<tr>
<td>$m = $ Extract URLs of $M[i]$;</td>
</tr>
<tr>
<td>for $j=1$ to end of $m$ begin</td>
</tr>
<tr>
<td>if (not existence of $m[j]$ in $M$) upend $m[j]$ to $M$; end</td>
</tr>
<tr>
<td>end</td>
</tr>
<tr>
<td>$j=0$;</td>
</tr>
<tr>
<td>for $i=1$ to end of $M$ begin</td>
</tr>
<tr>
<td>if (existence of String(.xml) in $M[i]$) begin</td>
</tr>
<tr>
<td>$X[j] = M[i]$;</td>
</tr>
<tr>
<td>$j++$; end</td>
</tr>
<tr>
<td>end</td>
</tr>
</tbody>
</table>

Nomenclature:

$D_i$ → Web Domain

DB → Data Base – collection of Web Domains

$M$ → URLs of input Website

$m$ → URLs of input Webpage

$X$ → XML URLs

2.3 Observations

Experimental results ensure that the system analysis, search time and data set creation time gets reduced by using the proposed system of XML URL dataset creation. This method is very simple to implement. The proposed method will be more useful for future research avenues.
2.3.1 Problems Encountered while Retrieving URLs

While extracting XML URLs system got following problems.

- Special symbols in between Web domains such as '(),' and 'space'.
- Web domains which are not renewed.

Special symbols such as comma (,), dollar ($) and space are included between the Web domains and Web domains which are not being active (not renewed) are distracts the algorithm from extracting URLs.

Here, in pre-processing method by using word stemming process, we have removed the special symbols such as comma and space. Inactive Web domains are removed by source code extractor. Once completing the pre-processing, active Web domains are given as input to the crawler and extracted the URLs as per domain requirement.

2.4 Dataset Discussion

After pre-processing, Web domains are given as input to the crawler and XML URLs are classified from extracted URLs. Obtained XML URLs are analyzed and categorized based on their semantic structure.

2.4.1 Different types of URLs retrieved

In the data set analysis part, we clearly identify the different kinds of XML Web pages based on its semantic structure, correspondence between the Web pages, tag information and basic XML characteristics of Web pages etc. Based on these features, XML Web pages can be manually classified into RSS pages, XML pages, HTML based XML pages and XML code (site map XML) pages. RSS pages are further divided based on its tag information such as plain (normal), image, media and atom.

The dataset analysis classifications are shown in the Figure 2.4 clearly.
Classification of XML URL:

![XML URL classification diagram]

Figure 2.4: XML Web page categories

Web page analysis:
Once the dataset analysis has been done the semantic structure is clearly studied in order to segment the Web pages. Web page analysis is equivalent to semantic structure analysis. Semantic structure analysis is a methodology for systematic description of the intended meaning of language, conceptual structure, and its lexical and syntactic description. Semantic structure of pure XML, code base (sitemap) XML, HTML based XML and RSS XML’s are analyzed clearly.

2.4.1.1 Semantic structure of code based XML Web pages
Code based or sitemap Web pages usually display the Web page as coded information instead of graphical representational view. Its semantic structure describes the opening tags along with the related information within the closing tags. Most of the sitemap Web pages are repeated in their tag names and some sitemap Web pages refer the link information also. General code information will simplify the difficulties of creating the Web pages. Web author likes to create...
Datasets

simple Web page without including any image or media as external files. Example of Code based or sitemap Web pages is as shown in Figure 2.5.

Example of code based XML Web page is http://qzz.in/sitemap.xml

```xml
<urlset>
  <url>
    <loc>http://qzz.in/</loc>
    <lastmod>2013-05-24T16:30:16+00:00</lastmod>
    <changefreq>always</changefreq>
    <priority>1.0</priority>
  </url>
</urlset>
```

Figure 2.5: Code based Web page

2.4.1.2 Semantic structure of HTML based XML Web pages

In the dataset analysis it has been clearly observed that obtained Web pages from the dataset creation process are XML Web pages although some of the XML
Datasets

Web pages are semantically coherent with the basic HTML predefined tags. Content of these types of Web pages are identified by one of the existing system called boiler pipe. Boiler pipe extracts the HTML content and it displays the Web page content. Figure 2.6 depicts the example of HTML embedded XML Web page. Semantic structure of HTML based XML Web pages are similar to the code based or sitemap XML Web pages, because XML Web pages are Extensible Markup Language it consists of SGML (Standard Generalized Markup Language) and HTML languages inbuilt with it. Hence the semantic structure will appear similarly.

![HTML embedded XML Web page](image)

**Figure 2.6: HTML embedded XML Web page**

An example of semantic structure of HTML embedded XML Web page is:


Tag 1: html
Description: Hyper Text Markup Language.

Tag 2: div
Datasets

Description: The `<div>` tag defines a division or a section in an HTML document. The `<div>` tag is used to group block-elements to format them with CSS.

Tag 3: title
Description: The title of the document.

Tag 4: timestamp
Description: Describes the published date and time of the Webpage

Tag 5: p
Description: The `<p>` tag defines the paragraph

2.4.1.3 Semantic structure of pure XML Web pages

XML is known to specify the user defined tags in the Web pages. The semantic structure includes the tag information which is normally identical and its source code depends on the Web author. Initially, Web page information is defined with the opening tags and closing tags. Some of the link information are specified with the `<link>` tag or through the reference tags i.e. `<href>`. Figure 2.7 depicts example of Pure XML Web page

![Pure XML Web page](www.w3schools.com/xml/simplexml.xml)

**Figure 2.7:** Pure XML Web page
An example of semantic structure of Pure XML Web page is:

http://www.w3schools.com/xml/simplexsl.xml

Tag 1: breakfast_menu
Description: contains the sub divisions of breakfast menu

Tag 2: name
Description: name of the breakfast

Tag 3: price
Description: price of the menu

Tag 4: description
Description: description about the menu

2.4.1.4 RSS Web pages and their semantic structure

RSS Rich Site Summary (originally RDF Site Summary, often dubbed Really Simple Syndication) is a family of Web feed formats used to publish frequently updated works—such as blog entries, news headlines, audio, and video—in a standardized format. An RSS document (which is called a "feed", "Web feed", or "channel") includes full or summarized text, plus metadata such as publishing dates and authorship.

RSS feeds benefit publishers by letting them indicate content automatically. A standardized XML file format allows the information to be published once and viewed by many different programs. They benefit readers who want to subscribe to timely updates from favorite Websites or to aggregate feeds from many sites into one place.

RSS feeds can be read using software called an "RSS reader", "feed reader", or "aggregator", which can be Web-based, desktop-based, or mobile-device-based. The user subscribes to a feed by entering into the reader the feed's URI or by clicking a feed icon in a Web browser that initiates the subscription process. The RSS reader checks the user's subscribed feeds regularly for new work, downloads any updates that it finds, and provides a user interface to monitor
and read the feeds. RSS allows users in avoiding manual inspection of all the Websites they are interested in, and instead subscribe to Websites such that all new content is pushed onto their browsers when it becomes available.

**Semantic Structure of RSS page:**

RSS Web page is semantically different from the XML Web pages. XML Web pages are developed with the user defined tags but, RSS structure information's are predefined. RSS Web page initially starts with the `<Channel>` element and `</channel>` element describes the end of RSS Web page. `<Channel>` elements contain the child element like `<title>`, `<link>`, `<description>`, `<item>`, `<image>`, `<category>` etc.. Each child elements tags describe the Web page content information. `<item>` elements also contain its child tag elements like `<author>`, `<category>`, `<comments>`, `<description>`, `<link>`, `<title>` elements etc.. Some of the tag elements are optional and some are mandatory elements based on the Web author who develops the Web page. Figure 2.8 depicts the example of RSS XML Web page.

![RSS XML Web page](image)

**Figure 2.8:** RSS XML Web page
An example of semantic structure of RSS XML Web page is:

http://www.aurmed.sk/rss.xml

Tag 1: Title
Description: The title of the document.

Tag 2: link
Description: Link allows users to click their way from page to page.

Tag 3: pubDate
Description: It describes the date and time of the publication of the page.

Tag 4: description
Description: It describes the overall content of the page.

2.4.2 Classes of RSS pages:

From the dataset analysis RSS Web pages are again classified as 4 different kinds of Web pages based on graphical representation and semantic structure.

- Plain RSS
- Image RSS
- Media RSS
- ATOM

2.4.2.1 Plain RSS

Normal RSS Web pages are contains only text information. It’s semantic structure are same as RSS Web page but it does not contain any external file information like image, media audio etc. Figure 2.9 shows the semantic tree structure of PLAIN RSS Web page.

Example of Plain RSS is http://www.120shiyou.info/rss.xml

Figure 2.10 depicts the example of Plain RSS Web page and its semantic structure.
Figure 2.9: Semantic Tree Structure of PLAIN RSS Web page
Datasets

2.4.2.2 Image RSS

Exam of Image RSS is http://www.aventuria.sk/rss.xml. Figure 2.11 shows the example of Image RSS and its semantic structure.
Figure 2.12: Semantic Tree Structure of IMAGE RSS Web pages

Figure 2.12 represents the semantic tree structure of Image RSS Web page. Image RSS Web pages are semantically same as RSS Web pages but different from the normal RSS. As name itself describes that these Web pages are contained with the image and text information.
2.4.2.3 Media RSS

Media RSS Web pages are different from the normal and image RSS Web pages which are semantically coherent with the RSS Web pages. Generally these Web pages are contained media files like audio and video as external file information. Example of Media RSS is www.logwin-logistics.com/rss.xml. Figure 2.13 shows the semantic tree structure of Media RSS Web pages and Figure 2.14 depicts the example Web page of Media RSS and its semantic structure.

![Diagram of Semantic Tree Structure of MEDIA RSS Web pages]

**Figure 2.13:** Semantic Tree Structure of MEDIA RSS Web pages
2.4.2.4 ATOM

The Atom Syndication Format is the next generation of XML-based file formats, designed to allow information—the contents of web pages. ATOM feeds are same as the RSS Web pages and it sometimes it referenced as alternative of the RSS Web pages. These are small text files giving the information about the updated content on the Website. Figure 2.15 depicts the pictorial example of ATOM Web page and its semantic structure.

Example: www.parketmilo.sk/zdroj/atom.xml

Some of the difference between the RSS and ATOM are:

- ATOM is an IETF (Internet Engineering Task Force) standard while RSS is not.
- ATOM feeds explicitly indicates the content while the browser is left to figure out whether the RSS feed contains plain text or escaped HTML.
- ATOM code is modular and reusable while RSS code is not.
RSS still holds dominance in the syndication format due to its head start and popularity.

Table 2.2 gives clear view of difference between the RSS and ATOM tag elements

<table>
<thead>
<tr>
<th>NAME</th>
<th>ATOM channel /ATOM item</th>
<th>RSS 2.0 &lt;CHANNEL&gt;</th>
<th>RSS 2.0 &lt;ITEM&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Container</td>
<td>Feed/ entry</td>
<td>Channel</td>
<td>Item</td>
</tr>
<tr>
<td>Title</td>
<td>Title/title</td>
<td>Title</td>
<td>Title</td>
</tr>
<tr>
<td>URL</td>
<td>Link/link</td>
<td>Link</td>
<td>Link</td>
</tr>
<tr>
<td>Summary</td>
<td>Subtitle</td>
<td>Description</td>
<td>Description</td>
</tr>
<tr>
<td>Date</td>
<td>Updated</td>
<td>Last build date</td>
<td>Pub date</td>
</tr>
<tr>
<td>Logo</td>
<td>Icon</td>
<td>Image</td>
<td></td>
</tr>
<tr>
<td>Author</td>
<td>Author</td>
<td>Managing editor</td>
<td></td>
</tr>
</tbody>
</table>

Figure 2.15: ATOM Web page and its Semantic Structure

Table 2.3 describes the number of different samples for each type of URLs extracted.
Table 2.3: Description of XML Dataset

<table>
<thead>
<tr>
<th>Sl. No.</th>
<th>Database Type</th>
<th>No. of Samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Pure XML</td>
<td>800</td>
</tr>
<tr>
<td>2</td>
<td>HTML based XML</td>
<td>1200</td>
</tr>
<tr>
<td>3</td>
<td>RSS XML</td>
<td>2100</td>
</tr>
<tr>
<td>4</td>
<td>Code based XML</td>
<td>600</td>
</tr>
</tbody>
</table>

2.5 Performance Analysis of proposed model with existing systems

To check the robustness of the proposed algorithm, here we have experimentally tested it by comparing the performance of existing systems.

2.5.1 Win Web Crawler (WWC)

Win Web Crawler is the one of top crawler to extract all matching URLs from search engines. WWC extracts URLs based on keywords, it means WWC extracts URLs of particular specified domain URLs such as Travel, Location, Photography, Offshore, Banking, Tax, Accounting, Health, Medicine etc.,

2.5.2 Web Crawler (Web Spider, Ant, Automatic Indexer)

A Web crawler starts with a list of URLs to visit, called the seeds. As the crawler visits these URLs, it identifies all the hyperlinks in the page and adds them to the list of URLs to visit, called the crawl frontier. URLs from the frontier are recursively visited according to a set of policies. If the crawler is performing archiving of Websites it copies and saves the information as it goes. Such archives are usually stored such that they can be viewed, read and navigated as they were on the live Web, but are preserved as 'snapshots'.

The large volume implies that the crawler can only download a limited number of the Web pages within a given time, so it needs to prioritize to download. The
Datasets

A high rate of change implies that the pages might have already been updated or even deleted.

Crawling policy

The behavior of a Web crawler is the outcome of a combination of policies:

- a selection policy that states which pages to download,
- a re-visit policy that states when to check for changes to the pages,
- a politeness policy that states how to avoid overloading Web sites, and
- a parallelization policy that states how to coordinate distributed Web crawlers.

2.5.3 Spadix URL Extractor (SUE)

SUE extracts URL with title, description, keywords meta data from entire Websites, list of URLs or search engine results. It presents results in url, base, domain, title, description, keyword, date modified, page size, etc. and user can

Screen Shots of SUE
save extracted data in text, excel, html file or CSV text format to import the output in any complex database tool as desire. Program has numerous filters to restrict extraction like - URL filters, date modified, file size, etc. It allows user-selectable recursion levels, retrieval threads, timeout, proxy support and accesses password-protected sites.

Since all the above Crawlers extract URLs based on certain conditions which fail to fulfill our following requirements.

- All existing URLs of given Web site (Home page) www.w3futures.com.
- Filter other Web site URLs while crawling particular.
- Classify XML URLs from extracted URLs.
- Automatically proceed for each Web site from DB.
- Extract unlimited number of URLs without any constraints.

Our proposed algorithm covers all the above requirements.
2.6 Flash URL Dataset Creation

The above proposed system fails to extract and create Flash URL Dataset, as Flash URLs do not contain any extension such as .xml, .html. Also from the literature survey it has been clearly observed that there is no Flash Dataset created to handle the segmentation process. Here are some example Flash URLs

- http://www.comunicacion.com/
- http://www.marvismint.com/
- http://www.sensisoft.com/
- http://www.asual.com/swfaddress/samples/flash/
- http://www.isim.ac.in/mlw
- http://www.noleath.com
- http://waterlife.nfb.ca/
- http://www.whittakers.co.nz/
- http://clicktoaid.sg/
- http://www.3spin.de/

Figure 2.16: Example of Flash Web page
Datasets

**Figure 2.17:** Example of HTML Web page

**Figure 2.18:** Example of XML Web page
Datasets

Therefore, here Flash URLs are collected based on analyzing its semantic structure. Flash Web pages does not allow users to view source code as it will be in the movie format (.swf - Shock Wave Flash). The Figure 2.16 and Figure 2.17 are shows the difference between Flash and HTML Web pages when user(s) right clicks on Web pages to view source code.

XML Web pages also allow users to view page source. The Figure 2.18 depicts the clear view of accessibility of source.

As in Figure 2.16, specified method we have created the sufficient amount (220 URLs) of Static Flash URL Dataset, 160 dynamic Flash Web pages and 400 HTML with Flash contents Web pages. Dataset has been divided as training and testing sets for experimental purpose.

Following Table 2.4 describe the number of different samples for each type of URLs extracted.

<table>
<thead>
<tr>
<th>Sl. No.</th>
<th>Database Type</th>
<th>No. of Samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Static Flash Web pages</td>
<td>220</td>
</tr>
<tr>
<td>2</td>
<td>Dynamic Flash Web pages</td>
<td>160</td>
</tr>
<tr>
<td>3</td>
<td>HTML with Flash Contents</td>
<td>400</td>
</tr>
</tbody>
</table>

2.7 Summary

In this chapter, we have presented the approach followed to create a large collection of database which contains XML URLs. We have also mentioned the complexity/problems encountered while acquiring these URLs. We have described the different types of XML URLs extracted based on their semantic orientation by analyzing them manually.

Also we have described the performance analysis with existing systems along with their pros and cons. Further by this, described about Flash dataset creation.
method. In our research, the data set created is utilized for experimentation purpose of our proposed preprocessing and recognition algorithms which is a very challenging task.