

# Design of an RSS Crawler with Adaptive Revisit Manager

Bum-Suk Lee<sup>1</sup>, Jin Woo Im<sup>1</sup>, Byung-Yeon Hwang<sup>1</sup>, and Du Zhang<sup>2</sup>

<sup>1</sup> Department of Computer Science and Engineering, The Catholic University of Korea, 43-1 Yeokgok 2-dong, Wonmi-gu, Bucheon-si, Gyeonggi-do 420-743, Republic of Korea  
{bslee,cukfan,byhwang}@catholic.ac.kr

<sup>2</sup> Department of Computer Science, California State University, Sacramento, U.S.A.  
zhangd@ecs.csus.edu

**Abstract**—RSS (Rich Site Summary, or Really Simple Syndication) is widely used for notifying readers of updated information on blogs and feeding news to readers quickly. RSS is very simple, and so is mostly used as a web service. However there is no satisfactory search engine which works for RSS. The reason is that RSS is continuously modified, and the structure of general search engines is ineffective to collect information from RSS sources.

In this paper, we discuss a web crawling algorithm, and propose a structure for an RSS crawler which is geared toward collecting and updating RSS in the Web2.0 environment. The proposed method (1) uses visited domain name history to predict the location of the RSS of a new seed URL, and (2) updates RSS information adaptively, based on some update-checking heuristics. These approaches can serve as cornerstones for an efficient and effective RSS search engine.

**Keywords**—Web2.0, RSS, Crawler, Adaptive Revisit manager

## 1. Introduction

Web2.0 [1] is a next generation web service paradigm which is different from past technologies. In this paradigm, information providers and users are engaged in interactive communications about their demands and needs [2]. RSS (Rich Site Summary or Really Simple Syndication), a novel XML based technique in the Web2.0 environment, is used for transferring data easily, and notifying readers of updates to blogs via RSS reader applications or meta-blogs almost in real-time.

Historically, RSS has been used to refer to “Rich Site Summary,” “RDF Site Summary” and “Really Simple Syndication.” It is a data format based on XML, used to supply update information on websites with frequent content updates [3]. RSS is very simple, and so is mostly used as a web service especially on blogs. People can subscribe to an RSS “feed” with an RSS reader application. The feed lets people know that new information is available even though they do not visit the website, and people can use the reader application to categorize contents to their taste.

In response to the increasing use of RSS, sites have emerged which offer an RSS reader-like portal service,

such as “allblog.net” and “naaroo.com” in Korea. These sites are called meta-blogs. People can add their RSS address to the meta-blogs by themselves. The meta-blogs check updates of registered blogs and categorize contents like news articles for portal sites. However the meta-blog is inconvenient because they involve passive registration of RSS by personal bloggers. Also people can only search for registered contents, and so the search results of portal contents are limited to narrow topics.

Development of a search engine specifically for RSS is needed to solve these problems. Generally speaking, a web search engine consists of three parts: a crawler, an indexer, and a searcher [4,5]. The crawler collects information from websites, and the indexer manages an index of gathered information to support fast retrieving. The searcher offers searching results based on the index. There are studies on RSS gathering, but the results have been less than satisfactory thus far. In this paper, we propose an RSS crawler that takes advantage of the features of RSS. This proposed RSS crawler makes use of gathered domain information to find the location of RSS from seed URLs. Furthermore it adaptively checks the updates from sites of collected RSS with an RSS manager. To do this, the RSS manager keeps information on the update time of each RSS feed.

The rest of the paper is organized as follows: Section 2 offers a brief overview and related work of the search engine and RSS. Section 3 describes the proposed RSS crawler with emphasis on the RSS crawling algorithm for effectively gathering RSS update information and on the structure of the RSS manager for adaptive updates of RSS feeds. Section 4 gives a comparison between our proposed approach and some existing work. Finally, in Section 5, we conclude the paper with remarks on future works.

## 2. Related Work

### 2.1. Web Crawler and RSS Search Engine

One of the most important reasons for Google to become a tremendously successful company in the web

service field is that Google has developed a search engine that crawls the enormous web very effectively. Google uses a totally automatic crawler and a specialized page-rank algorithm [6]. The days to register and categorize website information manually by humans are long gone. Most of the web search engines nowadays use some sort of automatic crawling algorithms.

There are researches for developing RSS crawler by various researching group [7-10]. Among them, beta version of Feedshow.com was developed by French researchers in 2006. It gathers RSS with crawler and provides a search engine. Furthermore Bloglines.com and Blogpulse.com are developed with their own RSS aggregator. However most of these services cannot adequately satisfy people's demands and needs, because the searching results are too limiting to fine useful information. Also ranges of the results are limited to their locality, such as Europe or the States. It seems that the problem stems from the lack of some effective crawling algorithm for aggregating RSS and from the disconnection between the web service and the features of RSS which update frequently.

There is also interest in developing the RSS indexer. Brooks and Montanez introduced autotagging and hierarchical clustering into the blogosphere [11]. The automatically generated tags were useful to help users with their choices. However these tags could not be combined with each other in any meaningful relationships, so people had to connect them by hands. It is acknowledged in the paper that the autotagging algorithm needs to be improved.

## 2.2. Features of RSS

RSS is updated when new content is added on a website or blog. The period of the updates is determined by the preferences of the information provider or the character of the service type. RSS can be located in the same URL-subpath as the blog in some cases, such as WordPress (an installable type blog) and Blogger (a joinable type blog), which are the two most popular blogging tools in the world.

The period of update is important in the collection of information from RSS. The internet has been growing consistently over the past two decades, and a large number of web pages are newly created every day. Furthermore the bloom of blogs results in new contents every day, with frequent updates all over the world. These voluminous newly created web pages far exceed the ability of search engines to crawl and process, so the revisit period for a particular website to check its update may be far longer than the update period of the site itself. This causes deterioration in the quality of search results, because the results may not contain the most recently updated contents until the crawler revisits the websites that appear in the search results.

This situation gets worsened especially in blogs. Many people post their thoughts about life and social issues, and

often discuss these issues with other people. However many events happen everyday, so information and issues that people need and are interested in change fast.

Precision and recall are the popular evaluation measures for search engines. Precision is the percentage of related documents with keywords in the results. Recall is the ratio of number of related documents with matching keywords returned by a search to the whole number of related documents. A search engine with a long revisit period often cannot show newly updated information. As a result, the recall quality of the search engine deteriorates. On the other hand, while frequent revisits can improve recall quality, they cause unnecessary network traffic and maintenance costs.

A main feature of RSS in both installable type of blogs and joinable type of blogs is that RSS is located in the same subpath as the blog, and so we can gather RSS easily based on the domain information without visiting all URL links. With the addition of a new module to use domain information, we can improve crawling speed.

## 3. Proposed RSS Crawler

### 3.1. Gathering RSS with Domain Information

The main use of RSS is for blogs and news feeds. Between these two types of web services, blogs use RSS mostly widely. Accordingly, we can design an RSS crawler which gathers RSS by considering the typical features of blogs. The most commonly used blogs belong to either the installable and joinable type, such as WordPress or Blogger. In installable type blogs, people install the blog application on a hosted site in their own domain, while people join and get sub level domain access in the joinable type.

People have their own domain and web server in the case of installable type blogs. For example, WordPress has to be installed on a web server, and so its RSS is located in the same sub-path, assuming the owner did not change when they installed it. This feature is also applied with other kinds of installable type blogs. Therefore when the crawler visits a blog, it can recognize which kind of installable blog application is used while reading any web pages on that domain. With proper storage and indexing of these kinds of information, the crawler can find the location of an RSS without visiting all links.

Blogger is a representative example of the joinable type of blogs. In this case, the blog creates a sub level domain with a user ID, so if a user joins with ID "tom" then, the address of the blog might be generated as "http://tom.blogspot.com/". Joinable blogs which generate sub level domains have a feature that allows RSS to be found in the same sub path location as the blog.

In this paper, we propose a path manager that uses the domain and path information when it crawls. The path manager analyzes and stores information about current

pages, and predicts the location of RSS in the current domain. The structure of the proposed crawler is shown in Figure 1. The crawler in Figure 1 has two features which are different from normal search engine crawlers: (1) a part for comparing domain information, and (2) a part for checking whether the blog is an installable type or a joinable type.

In the first step, the path manager checks that there is a visited upper level domain based on the top-level domain and second-level domain. This helps the crawler find RSS quickly in the case of joinable type blog. Next, the crawler checks whether the blog is an installable type, if there is no matched domain. The crawler identifies an installable type blog based on the pattern of path links and text. These can be analyzed and maintained during the crawling process. If the seed URL is based on either an installable or joinable type of blog, the crawler may find the location of RSS, and it can reduce unnecessary visits to find RSS.

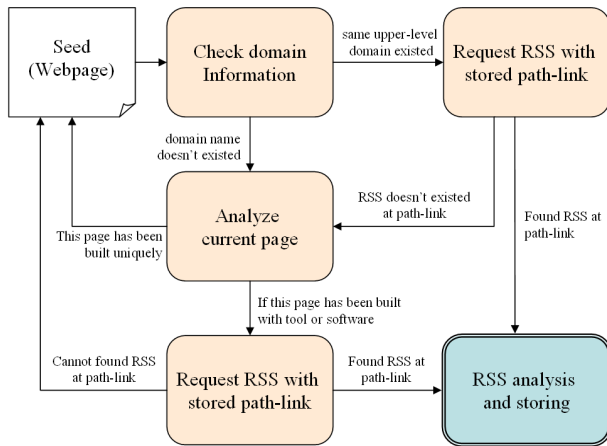


Figure 1. A structure of crawler for gathering RSS.

### 3.2. Adaptive Revisit Manager

In this subsection we describe an adaptive revisit method to check updates of RSS based on the update period of gathered RSS. Generally speaking, web crawlers revisit gathered websites occasionally to check updates and find errors in URL connections. RSS is more efficient on websites which are frequently updated than less visited websites. Furthermore information consumers are greatly satisfied when the RSS always includes recent information. Accordingly, the meta-blog and RSS reader applications have to provide recent information to satisfy people. The proposed crawler analyzes the <pubDate> element in gathered RSS, and uses it to predict the next update time. The <pubDate> element includes a date and time in the form: “<pubDate>Wed, 16 Jan 2008 01:51:44 +0900</pubDate>”. With this information we can determine when the website is updated: daily or on specific day such as on Saturday. The crawler predicts next update time based on analyzed data, and revisits the RSS at that

time. This adaptive revisit method is based on the posting pattern of the information provider.

The proposed crawler considers three pieces of information in the <pubDate> element: (1) The update time interval. The crawler finds the time interval of update, e.g., daily or weekly. This can be calculated from date information. (2) The day of the week. Posting a new article is the result of human work. People have their own life patterns, so the analysis of the day of the week can be an important clue. (3) The update time. A frequently updated time is also based on human patterns, and so we can expect that it is an effective indicator. Hence the crawler can revisit during frequently updated time to reduce the gap between posting time and revisit time. Based on the aforementioned heuristics we can expect an improvement in the recall quality of our search results.

Next we introduce an analysis method to be utilized by the adaptive revisit manager. This method analyzes gathered RSS files and applies a statistical method on the day of week and an adaptive method on time. Let  $P_u$  denote the probability of future day's update. Let  $U$  and  $\bar{U}$  be the value of recent updated data ( $U$  is constantly 1) and the mean of past updated data respectively. The algorithm estimates the  $\bar{U}$  using  $\bar{U} = \delta_u / \delta_w$  where  $\delta_w$  is a whole period for all data and  $\delta_u$  is a number of days where there exist updated data. Table 1-(a) shows the presence or absence of updated data in a sample RSS. In this example,  $\delta_w$  is 45 and  $\delta_u$  is 25, so  $\bar{U}$  is almost 0.556.

Table 1. Presence or absence of update in a sample RSS.

Sat.	Sun.	Mon.	Tue.	Wed.	Thu.	Fri.
1	0	0	0	1	1	0
1	0	0	0	1	0	1
1	1	0	1	0	1	1
1	1	1	1	1	0	0
1	1	1	0	1	0	0
0	1	0	0	1	0	0
1	1	1				

(a)

Sat.	Sun.	Mon.	Tue.	Wed.	Thu.	Fri.
6/7	5/7	3/7	2/6	5/6	2/6	2/6

(b)

The term  $\alpha$  is a weight factor that controls the rate of convergence of the algorithm (always at 0.9), and  $\beta$  is a statistical weight factor that is calculated from the mean of update on the day of week. The  $\beta$  can be calculated according to Table 1-(b). The statistical weight factor of Tuesday which is a next day of the latest update is 0.333(2/6) in this example. Finally the probability is estimated using  $P_u = \alpha\beta U + (1-\alpha)\bar{U}$ . We determine 0.5 as the threshold. Here, the probability value is 0.355. It is smaller than the threshold, so we can predict that there may not exist an update on the next day. After that the algorithm recalculates an update probability of the day after tomorrow (Wednesday). In this case  $\delta_w$  increases to 46. With the same method,  $\bar{U} = 25/46 \approx 0.543$  and  $\beta$  is 0.833(5/6), and so  $P_u$  is

0.804. We know that most likely there will be an update on Wednesday.

#### 4. Comparison

In this section we compare our approach with some of the similar web services. Table 2 summarizes the comparative results. The similar web service providers include: Google, Feedshow, Bloglines, Blogpulse, and Allblog.

Table 2. A comparison with similar web services.

	Search Engine with Crawler					Meta-blog
	Our approach	Google	Feedshow	Bloglines	Blogpulse	Allblog
Method for collecting RSS	<b>RSS crawler</b>	web crawler	RSS crawler	RSS crawler	RSS crawler	register RSS by hands
Method for checking update	<b>adaptive</b>	static	static	static	static	adaptive
Times until Update reflection	<b>short</b>	long	long	short	normal	normal

As we mentioned in this paper, our approach consists of two major components. (1) The crawler that crawls RSS efficiently with gathered domain information. It reduces useless out-links so as to find RSS fast. (2) Revisit manager that is based on an adaptive and statistical method for checking updates. This helps to reduce the cost for checking the updates, and to promptly reflect the updates as soon as they occur. Google has absolutely nice performance but its method for checking updates takes too much time. Performance of Feedshow is very poor while Bloglines and blogpulse are in good performance range. However Bloglines checks updates once an hour and Blogpulse checks it once a day, thus creating unnecessary checking cost. Allblog is a different kind of service. It is a meta-blog, and people register their RSS to share with others. Its checking update strategy is adaptive according to frequency of RSS update. On balance, our approach can provide efficient performance with low update-checking cost.

#### 5. Conclusion

RSS is the most widely used information distribution technique in the Web2.0 environment. It is an open standard for transferring data efficiently based on XML. RSS is widely used in many fields such as news and blog feeds, and data transferring on mash-up services. However, the state-of-the-practice in the field is that few existing RSS search engines is efficient and effective. As a result, development of better RSS search engines is critically needed.

In this paper, we described the design of an RSS crawler which gathers and updates RSS efficiently. The crawler, the main part of a search engine, is a tool to visit the vast World Wide Web and collect data. Commonly the crawler recursively analyzes out-links of web pages starting from a seed URL, but this method wastes too much time and

network traffic to work efficiently for RSS. Furthermore we wanted to design a crawler just for gathering RSS effectively. The proposed RSS crawler in this paper has two main features: (1) It gathers RSS easily based on path and domain analysis of installable and joinable type blogs, which represent the majority of RSS use. (2) It revisits gathered RSS adaptively based on update pattern analysis of the period, day of the week, and time of updates on the target site.

The proposed RSS crawler appears to gather RSS more efficiently, and the adaptive revisit method can improve recall quality of the search results. The proposed method offers the possibility of reducing wasted time on crawling, and improving recall quality. For future work, we intend to implement the proposed RSS crawler and evaluate its performance.

#### References

- [1] T. O'Reilly, "What Is Web2.0: Design Patterns and Business Models for the Next Generation of Software," self published on [www.oreilly.com](http://www.oreilly.com), 09/30/2005.
- [2] D. E. Millard and M. Ross, "Web2.0: Hypertext by Any Other Name?," In Proc. of ACM Conf. on Hypertext and Hypermedia 2006, pp. 22-25, 2006.
- [3] RSS version 2.0 Specifications, <http://blogs.law.harvard.edu/tech/rss>, 2003.
- [4] S. Brin and L. Page, "The Anatomy of a Large-Scale Hypertextual Web Search Engine," Computer Networks, Vol. 30, No. 1-7, pp. 107-117, 1998.
- [5] N. Blaž, "A Survey of Focused Web Crawling Algorithms," In Proc. of the Conf. on Data Mining and Warehouses, 2004.
- [6] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank citation ranking: Bringing order to the web," Unpublished Manuscript, 1998.
- [7] I. Rose, R. Murty, P. Pietzuch, J. Ledlie, M. Roussopoulos, and M. Welsh, "Cobra: Content based Filtering and Aggregation of Blogs and RSS Feeds," In Proc. of the 4th Symposium on Networked Systems Design and Implementation, pp. 29-42, 2007.
- [8] X. Li, J. Yan, Z. Deng, L. Ji, W. Fan, B. Zhang and Z. Chen, "A novel clustering-based RSS aggregator," In Proc. of 16th Int'l Conf. on WWW, pp. 1309-1310, 2007.
- [9] S. Buraga and T. Rusu, "Search Semi-Structured Data on Web," In Proc. of the 7th Int'l Symposium on Automatic Control and Computer Science, 2001.
- [10] D. Chmielewski and G. Hu, "A Distributed Platform for Archiving and Retrieving RSS Feeds," In Proc. of the 4th Annual Int'l Conf. on Computer and Information Science, pp. 215-220, 2005.
- [11] C.H. Brooks and N. Montanez, "Improved Annotation of the Blogosphere via Autotagging and Hierarchical Clustering," In Proc. of the WWW2006, 2006.