RSS-Crawler Enhancement for Blogosphere-Mapping

Justus Bross, Patrick Hennig, Philipp Berger, Christoph Meinel
Hasso-Plattner Institute, University of Potsdam
Prof.-Dr.-Helmert-Str. 2-3, 14482 Potsdam, Germany
{justus.bross, office-meinel}@hpi.uni-potsdam.de
{philipp.berger, patrick.hennig}@student.hpi.uni-potsdam.de

Abstract— **The massive adoption of social media has provided new ways for individuals to express their opinions online. The blogosphere, an inherent part of this trend, contains a vast array of information about a variety of topics. It is a huge think tank that creates an enormous and ever-changing archive of open source intelligence. Mining and modeling this vast pool of data to extract, exploit and describe meaningful knowledge in order to leverage structures and dynamics of emerging networks within the blogosphere is the higher-level aim of the research presented here. Our proprieteary development of a tailor-made feed-crawler-framework meets exactly this need. While the main concept, as well as the basic techniques and implementation details of the crawler have already been dealt with in earlier publications, this paper focuses on several recent optimization efforts made on the crawler framework that proved to be crucial for the performance of the overall framework.**

Keywords – weblogs, rss-feeds, data mining, knowledge discovery, blogosphere, crawler, information extraction

I. INTRODUCTION

Since the end of the 90s, weblogs have evolved to an inherent part of the worldwide cyber culture [9]. In the year 2008, the worldwide number of weblogs has increased to a total in excess of 133 million [14]. Compared to around 60 million blogs in the year 2006, this constitutes the increasing importance of weblogs in today's internet society on a global scale [13].

One single weblog is embedded into a much bigger picture: a segmented and independent public that dynamically evolves and functions according to its own rules and with ever-changing protagonists, a network also known as the "blogosphere" [16]. A single weblog is embedded into this network through its trackbacks, the usage of hyperlinks as well as its so-called "blogroll" – a blogosphere-internal referencing system.

This interconnected think tank thus creates an enormous and ever-changing archive of open source intelligence [12]. Modeling and mining the vast pool of data generated by the blogosphere to extract, exploit and represent meaningful knowledge in order to leverage (content-related) structures of emerging social networks residing in the blogosphere were the main objective of the projects initial phase [4].

Facing this unique challenge we initiated a project with the objective to map, and ultimately reveal, content-, topic- or network-related structures of the blogosphere by employing an intelligent RSS-feed-crawler. To allow the processing of the enormous amount of content in the blogosphere, it was necessary to make that content available offline for further analysis. The first prototype of our feed-crawler completed this assignment along the milestones specified in the initial project phase [4]. However, it soon became apparent that a considerable amount of optimization would be necessary to fully account for the strong distinction between crawling regular web pages and mining the highly dynamic environment of the blogosphere.

Section II is dedicated to related academic work that describes distinct approaches of how and for what purpose the blogosphere's content and network characteristics can be mapped. While section III focuses on the crawler's original setup, functionality and its corresponding workflows, the following section IV is digging deeper into the optimization efforts and additional features that were realized since then and that ultimately proved to be crucial for the overall performance. Recommendations for further research are dealt with in section V. A conclusion is given in section VI, followed by the list of references.

II. RELATED WORK

Certainly, the idea of crawling the blogosphere is not a novelty. But the ultimate objectives and methods behind the different research projects regarding automated and methodical data collection and mining differ greatly as the following examples suggest:

While Glance et. al. employ a similar data collection method as we do, their subset of data is limited to 100.000 weblogs and their aim is to develop an automated trend discovery method in order to tap into the collective consciousness of the blogosphere [7]. Song et al. in turn try to identify opinion leaders in the blogosphere by employing a special algorithm that ranks blogs according to not only how important they are to other blogs, but also how novel the information is they contribute [15]. Bansal and Koudas are employing a similar but more general approach than Song et al. by extracting useful and actionable insights with their *BlogScope*-Crawler about the 'public opinion' of all

blogs programmed with the blogging software blogspot.com [2]. Bruns tries to map interconnections of individual blogs with his *IssueCrawler* research tool [5]. His approach comes closest to our own project's objective of leveraging (content-related) structures and dynamics of emerging networks within the blogosphere.

Overall, it is striking that many respectable research projects regarding knowledge discovery in the blogosphere [1] [10] hardly make an attempt in explaining where the data - necessary for their ongoing research - comes from and how it is ultimately obtained. We perceive it as nearsighted to base research like the ones mentioned before on data of external services like Technorati, BlogPulse or Spinn3r [6]. We also have ambitious plans of how to ultimately use blog data [3] - we at least make the effort of setting up our own crawling framework to ensure and prove that the data employed in our research has the quantity, structure, format and quality required and necessary [4].

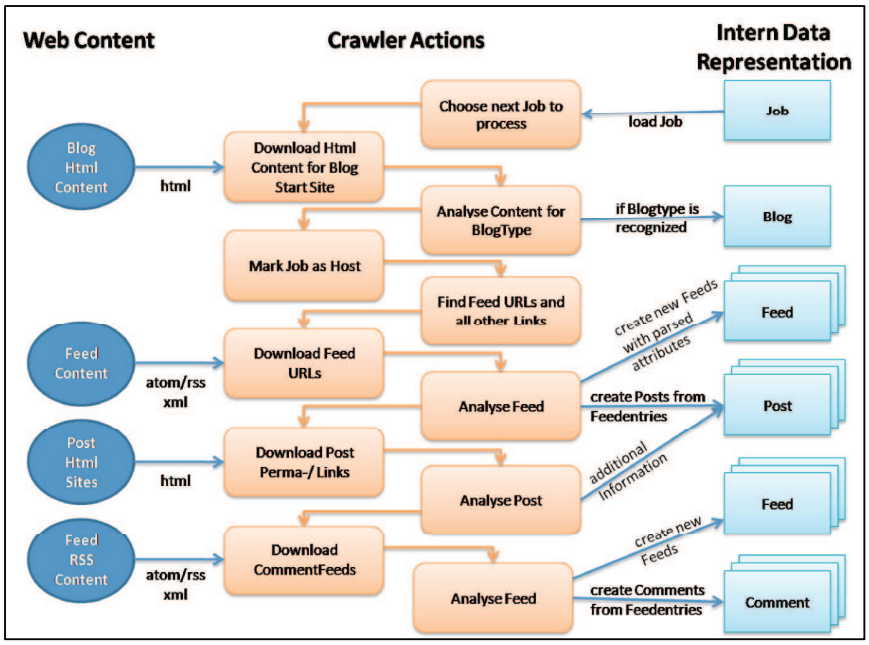

Figure 1. Action Sequence of RSS-Feed Crawler

III. Original Crawler Setup

The feed crawler is implemented in Groovy[1], a dynamic programming language for the Java Virtual Machine (JVM) [8]. Built on top of the Java programming language, Groovy provides excellent support for accessing resources over HTTP, parsing XML documents, and storing information to relational databases. Features like inheritance of the object-oriented programming language are used to model the specifics of different weblog systems. Both the specific implementation of the feed crawler on top of the JVM, as well its general architecture separating the crawling process into retrieval, scheduling and retrieval, allow for a distributed operation of the crawler in the future. Such distribution will become inevitable once the crawler is operated in long-term production mode. These fundamental programming characteristics were taken over for the ongoing development of the crawler framework.

The crawler starts his assignment with a predefined and arbitrary list of blog-URLs (see figure 1). It downloads all available post- and comment-feeds of that blog and stores them in a database. It than scans the feed's content for links to other resources in the web, which are then also crawled and equally downloaded in case these links point to another blog. Again, the crawler starts scanning the content of the additional blog feed for links to additional weblogs.

Whenever a link is analyzed, we first of all need to assess whether it is a link that points to a weblog, and also with which software the blog is created. Usually this information can be obtained via attributes in the metadata of a weblogs header. It can however not be guaranteed that every blog provides this vital information for us as described before. There is a multitude of archetypes across the whole HTML page of a blog that can ultimately be used to identify a certain class of weblog software. By classifying different blog-archetypes beforehand on the basis of predefined patterns, the crawler is than able to identify at which locations of a webpage the required identification patterns can be obtained and how this information needs to be processed in the following. Originally the crawler knew how to process the identification patterns of three of the most prevalent weblog systems around [11]. In the course of the project, identifications patterns of other blog systems followed. In a nutshell, the crawler is able to identify any blog software, whose identification patterns were provided beforehand.

The recognition of feeds can similarly to any other recognition-mechanism be configured individually for any blog-software there is. Usually, a web service provider that likes to offer his content information in form of feeds, provides an alternative view in the header of its HTML pages, defined with a link tag. This link tag carries an attribute (rel) specifying the role of the link (usually "alternate", i.e. an alternate view of the page). Additionally, the link tag contains attributes specifying the location of the alternate view and its content type. The feed crawler checks the whole HTML page for exactly that type of information. In doing so, the diversity of feed-formats employed in the

[1] http://groovy.codehaus.org/

web is a particular challenge for our crawler, since on top of the current RSS 2.0 version, RSS 0.9, RSS 1.0 and the ATOM format among others are also still used by some web service providers. Some weblogs above all code lots of additional information into the standard feed. The original version of the crawler only supported standard and well-formed RSS 2.0 formats, of which all the information of our currently employed object-model is readout.

Whenever the crawler identifies an adequate (valid) RSS-feed, it downloads the entire corresponding data set. The content of a feed incorporates all the information necessary, to give a meaningful summary of a post or comment – thus a whole weblog and ultimately the entire blogosphere. General information like title, description, categories as well as the timestamp indicating when the crawler accessed a certain resource, is downloaded first. Single items inside the feed represent diverse posts of a weblog. These items are also downloaded and stored in our database using object-relational mapping[2] (refer to figure 2). The corresponding attributes are unambiguously defined by the standardized feed formats and by the patterns that define a certain blog-software. On top of the general information of a post, a link to the corresponding HTML representation is downloaded and stored as well. In case this information is not provided in the feed information of a blog provider, we are thus still able to use this link information at a later point for extended analyses that would otherwise not be possible. Comments are the most important form of content in blogs next to posts, and they are usually provided in form of feeds as well.

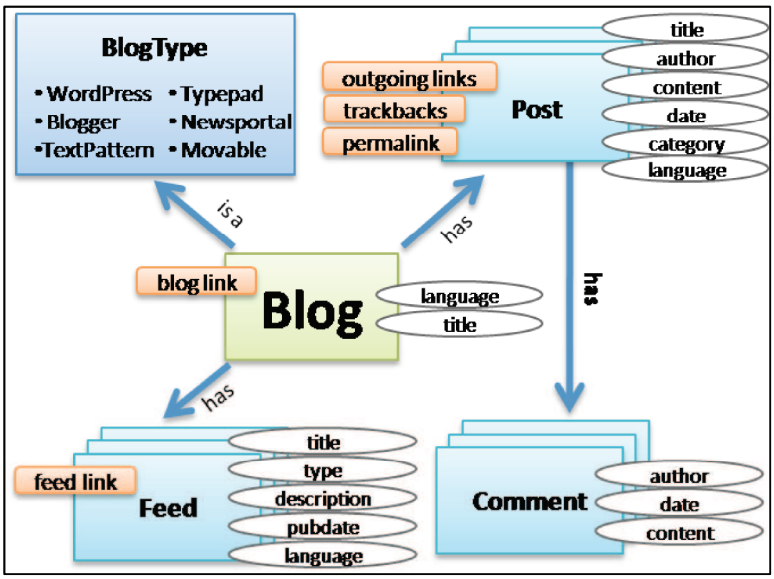

Figure 2. Intern Data Representation

However, we do need to take into account that a comment's feed-information is not always provided in the same form by all blog software systems. This again explains why we pre-defined distinct blog-software classes in order to provide the crawler with the necessary identification patterns of a blog system. Comments can either be found in the HTML header representation or in an additional XML attribute within a post feed. Comment feeds are also not provided by every blogging system. With the predefined identification patterns, our crawler is however able to download the essential information of the comment and store it in our database. Another important issue is the handling of links that are usually provided within posts and comments of weblogs. In order to identify network characteristics and interrelations of blogs within the whole of the blogosphere, it is not only essential to store this information in the database, but to save the information in which post or comment this link was embedded.

How often a single blog is scanned by our crawler depends on its cross-linking and networking with other blogs. Blogs that are referenced by other blogs via trackbacks, links, pingbacks or referrers are thus visited with a higher priority than others by the crawler. Well-known blogs that are referenced often within the blogosphere are also revisited and consequently updated more often with our original algorithm. It can be considered possible that with this algorithm blogs of minor importance are visited rarely – a side-effect that we do not consider to be limiting at this time. Implementing a different algorithm could at all times be realized by substituting the so-called "scheduler" of our crawler. As we will see in the following (section IV.f), this proved to be fundamentally important.

IV. Ongoing Optimization Efforts

A. *Identification of blogrolls*

A 'blogroll' is a list of links in a blog that a blogger defines irrespectively of the usual posting activities in his weblog. Usually, weblogs of friends or colleagues, with similar interest and topics or personal favorites are listed in such a blogroll. It is usually positioned in the sidebar of the starting page and represents one of the solely static parts of a weblog.

Since the original implementation of the crawler was determined to only analyze the RSS-feeds of weblogs, the information incorporated within blogrolls was entirely ignored. We did however come to the conclusion that blogrolls do represent an exclusive set of links to other weblogs due to the fact that their graphical positioning in the blog is visible to nearly every visitor of a weblog. To adhere to the distinct importance of interlinkages between weblogs, blogroll-linkages therefore had to be considered as well.

Identifying links within the listing of a blogroll is however not as easy as automatically identifying links within

[2] https://www.hibernate.org/

the standardized format of RSS-feeds. An analysis of the entire HTML-page of a weblog is necessary to identify a blogroll due to the fact that HTML-structure of blog-pages can fundamentally differ between bloggers and the blog software they employ. This is why blogrolls cannot be found at a particular position within the HTML-tree – also because they are often not embedded with an explicit tag.

Via a random analysis of the 100 biggest weblogs, we managed to identify patterns within their respective HTML-Content with which we are now able to identify blogrolls for the majority of blogging software systems around. These patterns are based upon the following two characteristics of blogrolls:

First, additional features such as a blogroll is often embedded into blog software systems via so-called 'plug-ins' – small software packages that can be modularly added to the main blogging system. Hereby, blogrolls can ultimately be identified via particular CSS-classes or Tag-IDS.

Secondly, the boxes on web pages that incorporate blogrolls are usually labeled with a common title (‚blogroll', 'bloglisting', or ‚blogrolle' in German language), after which the listing of referenced external weblogs follows within the HTML-code. This identification pattern was also used to track referencing links to other weblogs within the HTML code.

On the basis of the first identification pattern, blogrolls are identified and saved as links in the database. If this criterion will be sufficient to identify blogrolls within the majority of weblogs crawled remains to be seen and should be subject to further analyses. First results however indicate that this routine might be sufficient. The appliance of the second identification method requires the identification of specific titles and corresponding content within an HTML-document. This second method was not implemented yet, since we believe that the first method is sufficient for the crawler framework.

B. *Identification von trackbacks*

The identification of 'trackbacks' underlies similar particularities as the identification of blogrolls. 'Trackbacks' are referencing links to single posts, through which a blogger can explicitly state that a post of another weblog is of interest for himself and his own weblog. Regrettably, these special interlinkages are not represented in a common and standardized way throughout the blogosphere. This is why trackbacks generated within a weblog made of one blog software system might not be recognized as such in a weblog of another system. The rationale of identifying trackbacks is similar to the one of blogrolls: trackbacks are exclusive links that represent interdependencies of special interest between weblogs. They should therefore be tracked as well. Trackbacks can usually be identified as follows:

Trackbacks can – similar to blogrolls – be represented in an extra box within the HTML representation of a post. These areas can therefore also be identified via particular IDs or CSS-classes, but also through the recognition of the plain title 'trackback'.

Trackbacks are usually depicted as a citation of the referencing post in the commenting-section of a post (extract of the post's content) which is furthermore referenced via a hyperlink. This appearance of trackbacks can solely be identified via its unique citation form (usually like follows: "*[...]extract[...]*").

Manual trackbacks can also be found in the commenting section of a post. Manual trackbacks are referencing links that bloggers can enter in blog systems that do not support automated trackbacks - meaning that a blogger cannot add a trackback via an automated pingback or via a manual entry. Comparable with cross references to special news channels in services like Twitter, bloggers can than as alternativly enter comments in a format comparable to *„@Trackback myblog.de/p=12"*.

To extract manual trackbacks with an automated crawling system, you should thus be able to identify the word "trackback" and an immediately following link within a comment. Due to this rather general pattern we ultimately abdicated it from our analysis.

C. *Reliability of the Feedparsing*

The original crawler implementation processed feeds by making use of an XMLSluper API of Groovy that incorporates a LazyXMLDOM Parser. An HTML document was hereby converted in a valid DOM object by the XMLSluper API that was then analyzed for ATOM or RSS2.0 feed tags. This mechanism was however not perfectly applicable to map the entire blogosphere, since there are still many predecessor versions of RSS around (RSS0.9 or RSS1.0). We originally tried to embrace all feed formats by making use of the ROME framework of Apache in our original implementation [4]. Since the ROME API works on top of the Java SAXParser that in turn collapses every time it comes across invalid XML structures like unclosed tags (e.g. content in posts), special characters and other XML non -conform constructs, we had to come up with another solution. It was thus necessary to clean all feeds before the parsing process. Due to an enormous amount of characteristics that needed be adhered to in this regard, a manual implementation was not feasible. We therefore make use of the HTMLCLeaner, a library developed with the objective to clean XHTML pages. This cleaner successfully corrects any impurities in the feed format: it not only automatically adds valid namespaces, it also correctly closes HTML tags that were left open and therefore invalid and sources out all XML-reserved constructs in a corresponding CDATA tag. The subsequent result can then successfully be processed by the ROME framework.

D. *Language Detection*

In the course of the crawler project we came to the conclusion that language detection of those blogs crawled might be an interesting value-add when it comes to the analyses of the data crawled. However, the attribute

‚language' is only used in very rare occasions in the structure of a feed.
This is why the language detection module „JLangDetect"[3] was recently implemented on top of the original implementation. This library is written entirely in C-code and published under the Apache2 license. It requires so called 'gam-trees'. By making use of the *European Parliament Parallel Orpus*, we are now able to identify the following languages: danish (da), german (de), greek (el), english (en), spanish (es), finnish (fi), french (fr), italian (it), dutch (nl), portuguese (pt), and swedish (sv). The crawler analyzes the content's language of every post. This information is furthermore saved in our database (see figure 2). Since posts with different languages may have been published within one single weblog, language parameter of an entire blog is set according to the majority of posts with common language.

E. Postlinks

We soon had to discover that downloaded feeds in our original crawler implementation often incorporated only a short extract of the corresponding post content. This is due to a configurable setting in the backend of blog software systems. Since we are especially interested in content analyses of single weblogs at a later project phase, the entire HTML page of the posts therefore needed to be downloaded via the permalink address. Since there is a permalink address within every feed, this could be realized fairly easy.
Overall network analysis of the blogosphere is of major interest for us as well. It is therefore imperative that link information extracted from feed-crawling or parsing activities can be perfectly allocated to the corresponding posts, comments or weblogs. We therefore adapted the original crawling algorithm to ensure that not just feed-content is analyzed for link information, but HTML-pages of posts as well.

F. Priorization

How often a single blog is scanned by our crawler should depend on its cross- linkages with other blogs. Blogs that are referenced by other blogs via trackbacks, links, pingbacks or referrers are thus visited with a higher priority than others by the crawler. Well-known blogs that are referenced often within the blogosphere are also revisited and consequently updated more often with our original algorithm. It can be considered possible that with this algorithm blogs of minor importance are visited rarely - a side-effect that we do not consider to be limiting at this time. Since the blogosphere is constantly changing with new blogs being setup and other blogs disappearing, it is of crucial importance that the crawler preferable also finds new blogs and not only refreshes existing ones. We realized this requirement during our ongoing enhancement efforts on the basis of "priorities" - hereafter referred to as "Prio". A Prio is the number of hops necessary to get from the initial URL starting page to a particular blog, whereas all blogs within the starting list do have a Prio-value of 0. All those links that are collected on the front pages of one of the starting list blogs thus have a Prio-value of 1.

To guarantee that the crawler neither only updates those blogs it already found, nor merely tries to find new blogs without updating the information of the existing ones, new jobs to be crawled are scheduled as follows: There are several parallel working analyzers and a scheduler that determines which job will be processed next by the analyzers. The scheduler processes all jobs with Prio=0 on a daily basis. After that, all those links with Prio=1 that point to other blogs are also processed on a daily basis. At the time the analysis of blogs with Prio=1 is completed, the scheduler assigns two thirds of those analyz- ers available to analyze blogs with Prio = 2 that were not analyzed for more than a week. The remaining third of analyzers is assigned with new jobs. At the time these jobs are completed as well, one third of those analyzers available are assigned with jobs that point to blogs of Prio > 3 that were not processed for more than a week. The remaining analyzers are than equally filled up with new jobs. When all blogs in the database are updated, the scheduler assigns all analyzers with new jobs that were not visited so far.

Since the amount of collected jobs grew continuously since the start of the crawler project, it soon became necessary to optimize the queries on the database. Since blogs with Prio ">1" are revisited, it was so far necessary to know which job was pointing on a blog. As a consequence, both entities (jobs and blogs) needed to be logically connected – a highly time-consuming task for the analyzer. The job entity in the database was therefore extended by another field, indicating if a particular job is a blog or not. This considerably increased performance of the crawler.

G. News-portals

News-portals are of particular interest when it comes to the analysis of the blogosphere, since these portals often represent the virtual subsidiary of traditional news corporations. These players were traditional the ones to decide upon the daily headlines worldwide. With the advent of weblogs the rules of this game fundamentally changed. Without any central supervision or editorial standards, weblogs could write about whom and what they wanted – and they could do so a lot faster than traditional news corporations – even though this sometimes proved to be at the expense of journalistic quality. We consider it highly interesting to understand the interdependence of the blogosphere and traditional news corporations. The crawling algorithm therefore needed to be adapted in order to crawl news portals as well. For a start we only included the biggest German news portals in our analysis. Since RSS-feeds in highly respected news portals are greatly standardized and

[3] http://www.jroller.com/melix/entry/nlp_in_java_a_language

well-formed compared to the feed-quality in the blogosphere, we include news portals as another "blogtype" in our framework. In this case, the recognition patterns are linked to an internal URL-list of the biggest 100 German news portals.

Similar to the crawling activity in the blogosphere, news portal pages can then be scanned for links that point to weblogs. Due to the special annotation of news portals, it is then possible to analyze jumps between the blogosphere and "traditional" websites, and to find out what type of medium covered a particular story first.

It would even be possible to generate a kind of weblog-ranking, in which those weblogs that have a traceable influence on traditional media get a higher score than those without.

V. Suggestions for further Research

The feed crawler scans, recognizes and downloads blogs through their URI. It therefore needed to be ensured that a blog can only be found once. In the original implementation it could happen that two different job URLs were pointing on the same blog, which then was saved twice in the database. In the enhanced crawling framework, these blog duplicates can now be easily identified due to their identical host address. It is in contrast to that still not possible when the crawler encounters so-called redirects or short-URIs.

Currently, the feed-crawler recognizes feeds through the official feed link format. It is <link>-tag that has set *type= „application/rss+xml"* or *„application/atom+xml"*. This identifies the referenced resource without doubt as a feed. A tag can additionally be annotated with *rel="alternate"*. This identifies the feed explicitly as an alternative view of the currently opened resource (blog). It is therefore ensured that the feed incorporates the latest posts of a weblog. This method is quite effective to scan blogs for well-formed feeds. However, feeds that were identified this way often only enumerate the most recent posts and misses' information about older posts or categorization.

In order to find as much valid links as possible, our crawler currently scans all links without paying attention to the rel-attribute with the result that single posts could be crawled twice. Due to our internal identification of posts via their links, this is not a disadvantage eat the moment.

Blog-Analysis-Engines such as Spin3r use the so-called 'Brutal Feed recognition" or "Aggressive RSS discovery" method to adhere to this particularity. In doing so, they analyze every single link they come across on a blog page regarding its feed-characteristics. They hereby do not run the risk of skipping blog-feeds due to a missing annotation. The additional analysis however poses substantial additional expenses since the annotation of feeds is widespread in the Web – not only for blogs, but also for news-portals and other websites.

VI. Conclusion

Generally, we try to investigate in what patterns, and to which extent blogs are interconnected. We also have great interest in analyzing the content of single weblogs. In doing so we want to face the challenge of mapping the blogosphere on a global scale, maybe limited to the national boundaries of the German blogosphere. The visualization of link patterns, a thorough social network analysis, and a quantitative as well as qualitative analysis of reciprocally-linked blogs will to a large extend form the following project phase of our overall project, which will build upon the enhanced data collection method and technique described in this paper. Even though the original implementation performed well along the milestones defined within the initial project phase, it soon became apparent that those enhancements discussed in section IV of this paper were crucial for the overall performance of the crawling framework. We conclude that the feed-crawler is now running on a performance level that satisfies all requirements for long-term and large-scale data-mining in the blogosphere. Due to the enormous amount of blogs currently around, as well as those thousands of blogs and posts that add up to this amount of data every day, a final performance analysis of the crawler will follow in a couple of month.

References

[1] Agarwal, N., Liu, H., Tang, L., and Yu, P.S. Identifying the influential bloggers in a community. *Proceedings of the international conference on Web search and web data mining - WSDM '08*, ACM Press (2008), 207.

[2] Bansal, N. and Koudas, N. Searching the blogosphere. *Proceedings of the 10 th International Workshop on Web and Databases (WebDB*, WebDB (2007).

[3] Bross, J., Richly, K., Schilf, P., and Meinel, C. Social Physics of the Blogosphere: Capturing, Analyzing and Presenting Interdependencies of Partial Blogospheres. In M. Nasrullah and A. Reda, *Social Networks Analysis and Mining: Foundations and Applications (forthcoming)*. Springer Verlag, New York / Wien, 2010, 179-198.

[4] Bross, J.F., Quasthoff, M., Berger, P., Hennig, P., and Meinel, C. Mapping the blogosphere with rss-feeds. *The IEEE 24th International Conference on Advanced Information Networking and Application*, IEEE (2010).

[5] Bruns, A. Methodologies for Mapping the Political Blogosphere: An Exploration Using the IssueCrawler Research Tool. *First Monday 12*, 5 (2007).

[6] Chau, M., Xu, J., Cao, J., Lam, P., and Shiu, B. A Blog Mining Framework. *IT Professional 11*, 1 (2009), 36-41.

[7] Glance, N.S., Hurst, M., and Tomokiyo, T. BlogPulse: Automated Trend Discovery for Weblogs. *WWW 2004 Workshop on the Weblogging Ecosystem, ACM, New York 3rded*, (2004).

[8] Gosling, J., Joy, B., Steele, G., and Bracha, G. *The Java language specification*. Addison-Wesley, 2000.

[9] Herring, S., Scheidt, L., Bonus, S., and Wright, E. Bridging the gap: A genre analysis of weblogs. *Proceedings of the 37th Hawaii International Conference on System Sciences (HICSS'04)*, (2004).

[10] Lakshmanan, G.T. and Oberhofer, M.A. Knowledge Discovery in the Blogosphere. *IEEE Internet Computing 14*, 2 (2010), 24-32.

[11] Mintert, S. and Leisegang, C. Liebes Tagebuch ... Sieben frei verfügbare Weblog-Systeme. *iX-Archiv 7*, (2008), 42-53.

[12] Schmidt, J. *Weblogs: eine kommunikationssoziologische Studie*. Uvk, 2006.

[13] Sifry, D. State of the blogosphere, October, 2006. *Sifry.com - Sifry's Alerts*, 2006. http://www.sifry.com/alerts/archives/000443.html.

[14] Smith, T. Power to the People: Social Media Tracker Wave 3. *Retrieved on September 2*, 2008. http://www.goviral.com/articles/wave_3_20080403093750.pdf.

[15] Song, X., Chi, Y., Hino, K., and Tseng, B.L. Identifying Opinion Leaders in the Blogosphere. *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management (CIKM ' 07*, , pp971-974.

[16] Whelan, D. In A Fog About Blogs. *American Demographics July 1*, 2003, 2. http://findarticles.com/p/articles/mi_m4021/is_6_25/ai_105777528/ .