WebRelievo: A System for Browsing and Analyzing the Evolution of Related Web Pages

Masashi Toyoda and Masaru Kitsuregawa
Institute of Industrial Science, University of Tokyo
4-6-1 Komaba Meguro-ku, Tokyo, JAPAN
E-mail: toyoda,kitsure@tkl.iis.u-tokyo.ac.jp

Abstract

WebRelievo is a system for browsing and analyzing the evolution of the web graph structure based on link analysis. This system enables us to answer historical questions, and to detect changes in topics on the Web. WebRelievo extracts web pages related to a focused page using link analysis, and visualizes the evolution of their relationships with a time series of graphs. This visualization enables us to understand when related pages appeared, and how their relationships have evolved over time. The user can interactively browse those related pages by changing the focused page and by changing layouts of graphs. WebRelievo is implemented on six Japanese web archives crawled from 1999 to 2003.

1 Introduction

The Web has been growing and changing its structure by reflecting real social activities. For example, when important events such as war and terrorism occur in the real world, many web pages about these events are created, and become connected by hyperlinks. Since hyperlinks represent attention of page authors to the destination pages, we could detect changes in trends on the Web from the evolution of the hyperlink structure. Now, it becomes an important issue to track structural changes in the web.

We propose the WebRelievo system for browsing and analyzing the evolution of the web structure based on a series of large *web archives*. Currently, we use six web archives of Japanese web pages crawled from 1999 to 2003. This system allows us to answer historical questions, and to detect changes in topics on the Web. Figure 1 shows a screen snapshot of WebRelievo. Six graphs represent the same part of those six web archives. In those graphs, each node represents the URL, and each edge represents the relationship between URLs extracted by link analysis. These graphs are aligned from left to right then top to bottom by their time. Positions of URLs are synchronized over time, so that the user can easily recognize the changes in graphs.

To visualize the structure of the web, we do not use directly *the web graph*, in which nodes are web pages and edges are hyperlinks. It is because the web graph itself is too complicated to understand and to visualize its structure. Since, famous web pages are linked to by thousands of other pages, it is difficult to display even a subgraph of the web in limited screen space.

Therefore, WebRelievo visualizes relationships between web pages calculated by a link analysis technique, and show the evolution of their relationships. To calculate relationships, we use a related page algorithm (RPA) based on HITS [9] that takes a seed page as an input, and outputs related pages to the seed, by extracting a specific pattern of densely connected subgraph in the web. We visualize the relationships in each web archive by a *derivation graph* that is a directed graph representing how each page derives other pages by RPA.

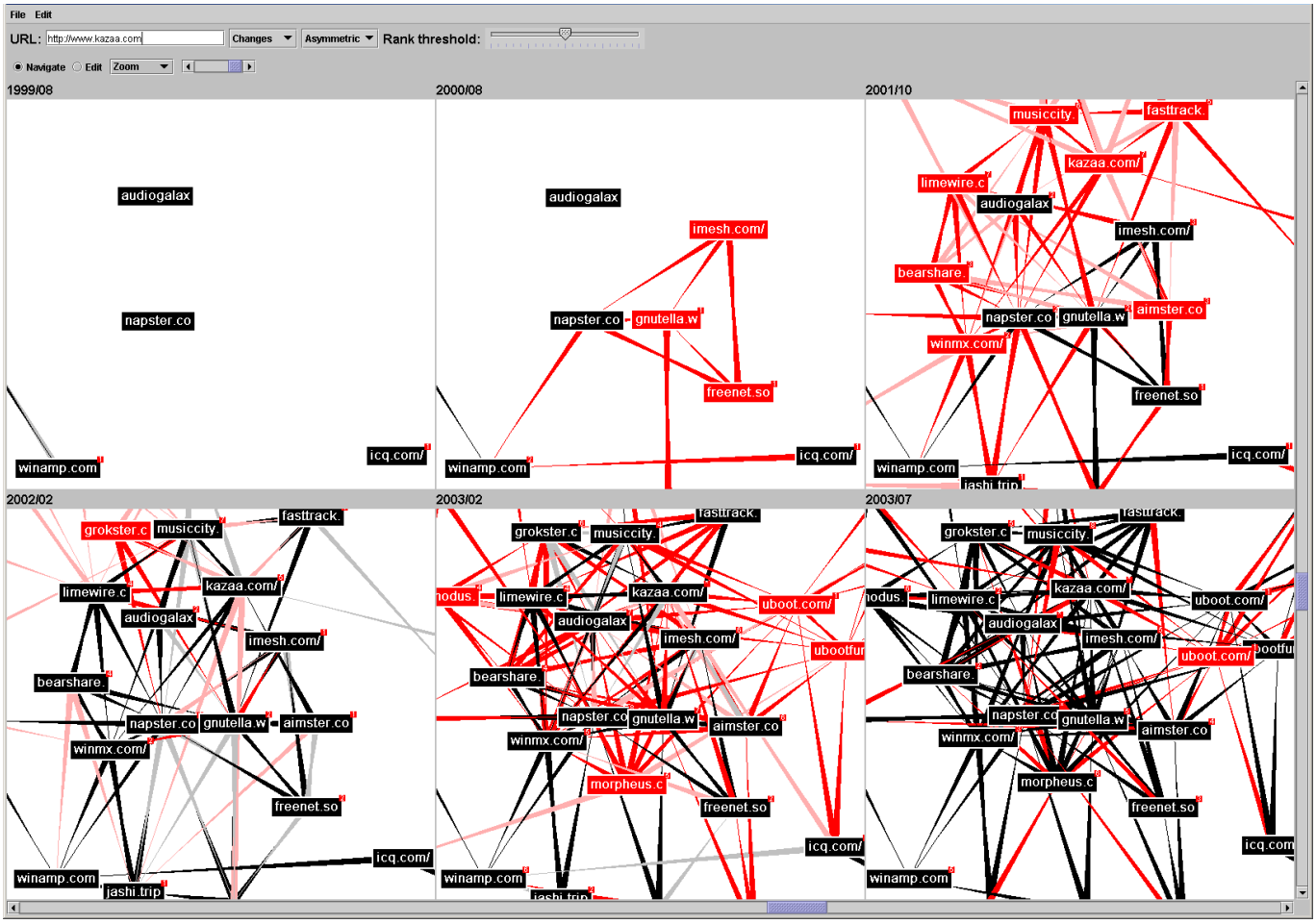

Figure 1: A screen snapshot of WebRelievo

For all web archives, WebRelievo makes derivation graphs around a focused page given by the user, then align these graphs as shown in Figure 1. The direction of each edge is shown by its thickness at each ends. That is, each edge starts at the thick end, and goes to the thin end.

This visualization enables the user to understand when related pages appeared and disappeared, and how their relationships have changed over time. In addition, the user can interactively browse those related pages by changing the focused page and by dragging nodes.

The rest of this paper is organized as follows. Section 2 shows some related work. Section 3 explains details of the derivation graph. Section 4 describes how to visualize the evolution and how the user can interact with WebRelievo. Section 5 shows some evolution examples with WebRelievo. The system architecture and implementation are described in Section 6. Finally, we conclude in Section 7.

2 Related Work

2.1 Visualizations of Evolution

There have been some work on visualizing evolution of information structure [3, 4, 5, 13]. Chen examined animated visualization of the evolution of co-citation networks of scientific publications [3]. Chen compared

two link reduction techniques to show which one is suitable for animation of the evolution [4]. Chi proposed the time tube technique [5] to visualize the evolution of a single web site structure, and accesses patterns on that site. It visualizes the hierarchical structure of the web site as a disk tree, in which the root page is put on the center, and child pages fan out from the root. Multiple disks are created for each time period, and aligned left to right by their time, so that the user can observe changes over time. The time tube visualization is basically similar to WebRelievo, while we mainly visualize relationships between web sites, and we do not restrict the structure to the tree.

Our previous work [12, 13] visualized evolution of web communities. A web community is a set of web pages with a common interest on a topic. In [12], we proposed a method for extracting all web communities from a single web archive using the derivation graph. In [13], we extracted all web communities from periodically crawled web archives, and visualized changes of these communities, such as growth, shrinkage, merge, and split. Rather, WebRelievo visualizes page level changes of relationships by the derivation graph. It can be used for detailed examination of web communities.

2.2 Related Page Algorithms

WebRelievo uses a related page algorithm (RPA) that takes a seed page as an input, and outputs related pages to the seed. We adopt a RPA based on the notion of *authorities* and *hubs* proposed by Kleinberg [9]. An authority is a page with good contents on a topic, and is pointed to by many good hub pages. A hub is a page with a list of hyperlinks to valuable pages on the topic, that is, points to many good authorities. HITS [9] is an algorithm that extracts authorities and hubs from a given subgraph of the Web with efficient iterative calculation.

RPA first builds a subgraph of the Web near the seed, and extracts densely connected authorities and hubs in the graph using HITS. Then authorities are returned as related pages. There are some variants of RPA based on HITS, such as Companion [6]. Since existing RPAs provide insufficient precision, we use an improved algorithm Companion– [12] in our previous work. The algorithm of Companion– is given in Appendix.

There are also some RPAs that are not based on HITS. Lempel and Moran [10] proposed another approach based on a random walk model for calculating authorities. Flake et al. [7] redefined a community including given seed pages as a subgraph that is separated from the Web using a maximum flow/minimum cut framework. Although we use a HITS based RPA, what we need is ranked lists of pages related to seeds. Therefore, it is easy to replace RPA to other algorithms such as SALSA.

3 Derivation Graph

In this section, we first introduce the notion of derivation graph proposed in [12]. Then we describe how to select and build a time series of derivation graphs that is newly required by WebRelievo.

Since the structure of the web graph is too complicated to visualize and to understand, we simplify the web graph using a related page algorithm (RPA) that returns related pages to a given page. That is, we visualize how each page derives other pages by RPA. To represent such relationships, we use a notion of derivation graph $DG = (V, E)$. Each node $v \in V$ represents a page. Each directed edge $e \in E$ from a node p to another node q, represents that p derives q as one of the top N related pages.

An appropriate value of N would differ according to the focused topic. By changing the parameter N, we can change the density of the derivation graph. When N becomes smaller, DG only connects more densely connected authorities, and becomes sparser. When N becomes larger, DG connects less densely connected authorities, and DG becomes denser. Note that the notion of DG does not depend on a specific RPA. We can use any RPAs, such as HITS, Companion, and SALSA, for building DG.

When a DG is still complicated to understand, we can make the DG more simple by extracting mutually connected nodes by derivations in DG. That is, p derives q and vice versa. Using a HITS based RPA, we found that such nodes are closely related in our previous work[12]. The extracted graph is called the symmetric derivation graph (SDG). In SDG, we only use symmetric derivations between nodes in DG. That is, a edge between p and q exist in SDG when p derives q and vice versa. Note that the density of SDG can also be changed by N. In WebRelievo, the user can choose from DG and SDG for visualizing relationships between pages.

WebRelievo shows the evolution, using a time series of derivation graphs built from a time series of web archives: $\{DG_t = (V_t, E_t) | 1 \leq t \leq T\}$. The subscript t denotes the time when each archive crawled (1 is the first time and T is the last time). Since each DG_t is too large to browse in a single screen, WebRelievo extracts and shows subgraphs of DGs around a given focused node p: $\{DG_t(p) = (V_t(p), E_t(p)) | 1 \leq t \leq T\}$.

To show appearance and disappearance of all nodes related to p over time, we extract pages related to p for each time, and track all of these pages. The following is the process to build a time series of those subgraphs:

1. For each time t, extract a set $R_t(p)$ of nodes related to p. It means that each node in $R_t(p)$ is pointed to by p in DG_t.
2. For each $DG_t(p)$, $V_t(p) = V_t \cap (\cup_t R_t(p))$.
3. For each $DG_t(p)$, $E_t(p) = \{(u, v) \in E_t | u, v \in V_t(p)\}$.

4 Visualization and User Interaction

WebRelievo visualizes a time series of DG using an automatic and dynamic graph layout algorithm based on a kind of force-directed model [8]. When visualizing a time series of DGs, the system should clearly show the difference between DGs. The following is the requirements for layouting DGs.

- The same node should be located at the same position over multiple DGs.
- Changes in nodes and edges of neighboring DGs should be easily recognized.

4.1 Synchronized Layout

To satisfy the first requirement, we modify the force-directed model by adding the feature to synchronize the position of the same node in multiple DGs. The force-directed model considers a graph as a physical system, in which attractive forces F_a are exerted on all pairs of connected nodes, and repulsive forces F_r are exerted on all pairs of nodes. In WebRelievo, F_a and F_r is defined as a function of the distance d between two nodes as follows:

$$F_a(d) = d^2/k^2, F_r(d) = -k/d$$

Where k is the ideal length between nodes.

The synchronization of node positions is performed between neighboring DGs. That is, each node in each DG is attracted to the position of the same node in the previous and next DGs. We add the following two forces for each node.

$$F_{prev}(d_p) = d_p/2, F_{next}(d_n) = d_n/2$$

In these equations, d_p and d_n are the distances from the node to the same node in the previous and next DG, respectively. These distances are zero, when the same node does not exist in the neighboring DGs.

Initially, nodes are randomly located in each panel, then each nodes are iteratively moved by those forces. The layout is fixed, when the total movement of nodes become less than a threshold. This iterative layout is shown by animation in WebRelievo.

4.2 Showing Differences between DGs

For the second requirement, nodes and edges are colored by their types of changes. For example, if a node or an edge appeared at time t, we use a red color for it. In this way, the user can see what kinds of changes have occurred and will occur on each node and edge.

Each $DG_t(p)$ is compared with the previous one $DG_{t-1}(p)$, and the next one $DG_{t+1}(p)$, then changes in each node (or edge) are classified into four types and colored as follows:

Stay–Stay When the node (or edge) exists in both DG_{t-1} and DG_{t+1}, it is colored black to show its stability.

Stay–Disappear When the node (or edge) exists in DG_{t-1}, but does not exist in DG_{t+1}, it is colored light gray to show its stay from $t-1$ and disappearance at $t+1$.

Appear–Stay When the node (or edge) does not exist in DG_{t-1}, and exists in DG_{t+1}, it is colored red to show its appearance and stay to $t+1$.

Appear–Disappear When the node (or edge) exists neither in DG_{t-1} and in DG_{t+1}, it is colored light red to show its volatility.

4.3 User Interaction

WebRelievo supports the following user interaction.

- The user can designate the focused URL by typing the URL in the input box, which is in the top-left of Figure 1. Then all derivation graphs around the URL are built and displayed. When the user designate another URL in the input box, graphs are built for the another URL, and merged to existing graphs. The same operation can be done on existing URL using a pop-up menu.
- The user can scroll and zoom into graphs. These kinds of changes in a graph are immediately propagated to all graphs for keeping layouts synchronized. Nodes in all graphs can be moved by dragging. When the user dragged a node in a graph, the node with the same URL is moved to the same position in each graphs. Layouts of all graphs are re-calculated and animated by the force-directed model, simultaneously. The user can easily keep track of the evolution after those operations.
- The user can examine the effect of parameter N in Section 3, by dynamically changing N with the slider that is in the top-center of Figure 1. When N is changed by the user, all graphs are immediately changed using N.
- The user can choose visualization of graphs from DG and SDG (See Section 3). When SDG is chosen, the user can reveal asymmetric edges around a selected URL using the pop-up menu of the URL. This change is also propagated to all graphs.

5 Evolution Examples

In this section, we show two evolution examples with WebRelievo. The first example in Figure 2 shows the evolution of P2P file sharing systems. In this case, the user wants to know the history of P2P systems, and designates the URL of a famous P2P system Kazaa (`www.kazaa.com`). Figure 2 shows the evolution by SDG (Figure 1 uses DG). From this result, the user can see that the first P2P system is Napster, and then various systems appeared such as Gnutella (2000), WinMX (2001), and Kazaa (2001). Famous systems became densely connected, and formed an almost clique.

Year	Period	Crawled pages	Total URLs	Links
1999	Jul. to Aug.	17M	34M	120M
2000	Jun. to Aug.	17M	32M	112M
2001	Oct.	40M	76M	331M
2002	Feb.	45M	84M	375M
2003	Feb.	66M	384M	1058M
2003	Jul.	98M	601M	1587M

Table 1: Details of web archives

The second example shows the evolution of search engines for mobile phone internet services (mainly for the i-mode service of NTT DoCoMo). In this case, the user want to know how the trend of those search engines has changed, and designate a famous search engine (`mobile.yahoo.co.jp`). In Figure 3, we can clearly see the changes in the trend of i-mode search engines. In 1999, search engines for i-mode were mainly provided by small companies, and they formed a clique. In 2000, major companies such as Yahoo! and Lycos began to provide i-mode search engines, and gradually the clique moved to these major companies. In 2003, the clique in 1999 was disappeared.

6 Architecture and Implementation

The architecture overview of WebRelievo is shown in Figure 4. WebRelievo is based on three databases built from web archives. We use six web archives of Japanese web pages crawled from 1999 to 2003 (See Table 1). Our crawler collected pages in the breadth-first order. From 2001, the number of pages became more than twice of the 2000 archive, since we improved the crawling rate. Until 2002, we collected pages in only .jp domain. From 2003, we began to collect pages in other domains, such as .com, if they are written in Japanese. Those collected documents are stored in the document archive on the right-bottom of Figure 4. Each document can be retrieved by its URL and the time of crawling.

From each archive, we built a link database (on the right-center of Figure 4) with URLs and links by extracting anchors from all pages in the archive. Our link database included not only URLs inside the archive, but also URLs outside pointed to by inside URLs. As a result, the graph included URLs outside .jp domain, such as .com and .edu. Table 1 also shows the number of links and the total URLs. For efficient link analysis, each link database is implemented as a main-memory database that provided out-links and in-links of a given URL. Its implementation was similar to the connectivity server [1].

The related page database on the right-top of Figure 4 is used to speed up the retrieval of derivation graphs. It stores pre-calculated results of RPA for popular URLs. The popularity of a URL is determined by the number of in-links. In our implementation, we pre-calculate related pages for URLs that have three or more in-links from other web servers.

The related page manager, in the center of Figure 4, provides information of derivation graphs to the evolution viewer on the top-left of Figure 4. This manager builds subgraphs of DGs for the focused URL using the related page database. When required URLs are not stored in the related page database, it calculates related pages by RPA using the link database. The RPA used by the manager can be changed to any RPA based on link analysis. In that case, the related page database should be also replaced.

The evolution viewer is the user interface of WebRelievo. The viewer displays DGs provided by the related page manager. We use the TouchGraph [11] to layout DGs, and modified the layout algorithm to synchronize node positions. The web browser (e.g. Mozilla) is used to browse contents of URLs designated by the user. The browser accesses the document database to show past documents.

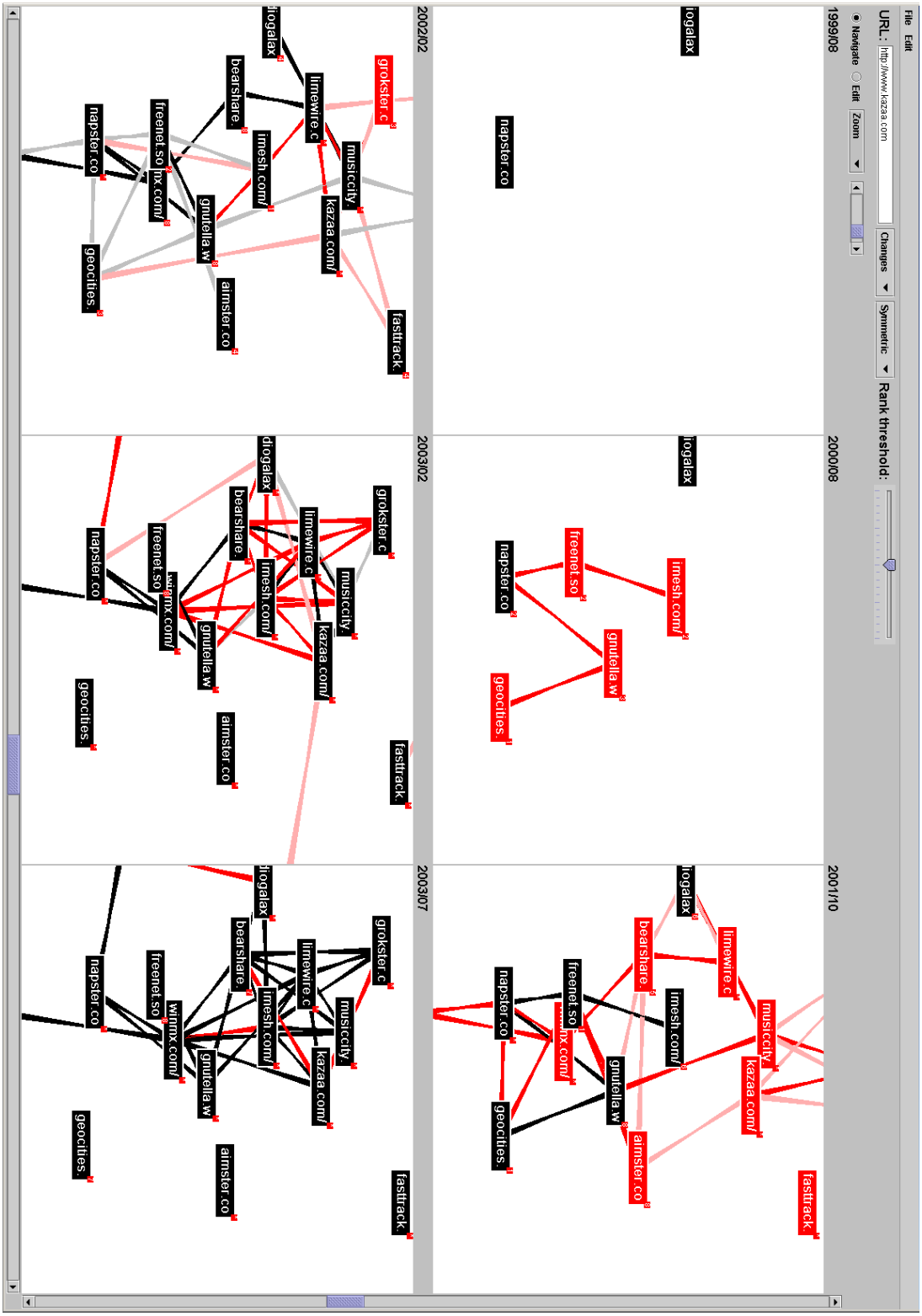

Figure 2: Evolution of P2P file sharing systems

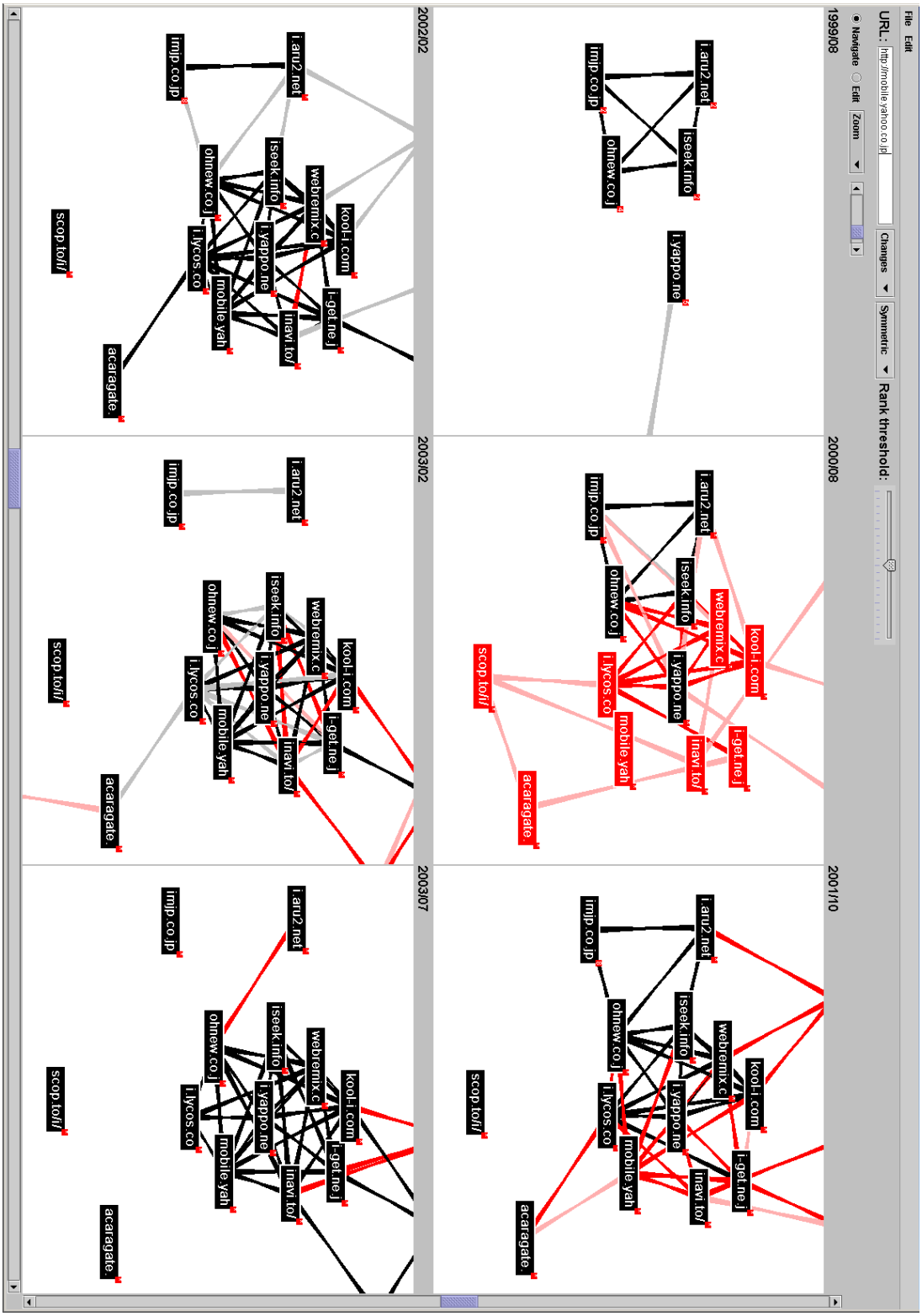

Figure 3: Evolution of search engines for mobile phone internet services

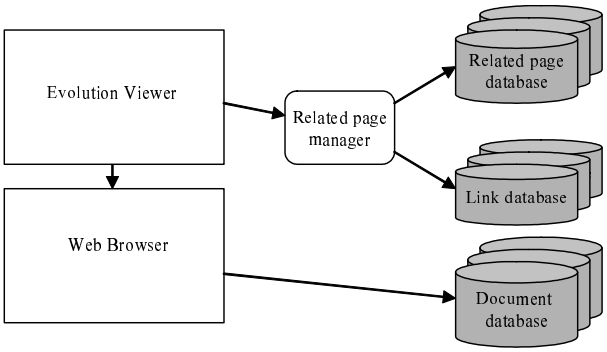

Figure 4: Architecture overview

7 Conclusions

We have proposed the WebRelievo system for browsing and analyzing the evolution of the web structure based on a HITS based link analysis technique. This system allows us to answer historical questions, and to detect changes in topics on the Web, from appearance and disappearance of pages on a focused topic, and evolution of their relationships. WebRelievo helps us understand the evolution by showing a time series of derivation graphs, and their synchronized layouts.

WebRelievo can detect detailed changes in relationships of pages. However, it is difficult to see the global changes in the Web. Now we are implementing clustering of web pages for showing larger areas of derivation graphs. We also plan to crawl the web more frequently, and visualize more continuous evolution of the web.

References

[1] Krishna Bharat, Andrei Broder, Monika Henzinger, Puneet Kumar, and Suresh Venkatasubramanian. The Connectivity Server: fast access to linkage information on the Web. In *Proceedings of the 7th International World Wide Web Conference*, pages 14–18, 1998.

[2] Krishna Bharat and Monika Henzinger. Improved Algorithms for Topic Distillation in a Hyperlinked Environment. In *Proceedings of ACM SIGIR '98*, pages 104–111, 1998.

[3] Chaomei Chen and Leslie Carr. Visualizing the evolution of a subject domain: A case study. In David Ebert, Markus Gross, and Bernd Hamann, editors, *IEEE Visualization '99*, pages 449–452, San Francisco, 1999.

[4] Chaomei Chen and Steven Morris. Visualizing Evolving Networks: Minimum Spanning Trees versus Pathfinder Networks. In *IEEE Visualization 2003*, pages 67–74, 2003.

[5] Ed H. Chi, James Pitkow, Jock D. Mackinlay, Peter Pirolli, Rich Gossweiler, and Stuart K. Card. Visualizing the Evolution of Web Ecologies. In *Proceedings of ACM SIGCHI '98*, pages 400–407, 1998.

[6] Jeffrey Dean and Monika R. Henzinger. Finding related pages in the World Wide Web. In *Proceedings of the 8th World-Wide Web Conference*, pages 389–401, 1999.

[7] Gary W. Flake, Steve Lawrence, and C. Lee Giles. Efficient Identification of Web Communities. In *Proceedings of KDD 2000*, pages 150–160, 2000.

[8] Thomas M. J. Fruchterman and Edward M. Reingold. Graph drawing by force-directed placement. *Software - Practice and Experience*, 21(11):1129–1164, 1991.

[9] Jon M. Kleinberg. Authoritative Sources in a Hyperlinked Environment. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, pages 668–677, 1998.

[10] R. Lempel and S. Moran. The Stochastic Approach for Link-Structure Analysis (SALSA) and the TKC Effect. In *Proceedings of the 9th World-Wide Web Conference*, pages 387–401, 2000.

[11] Alexander Shapiro. Touchgraph. http://www.touchgraph.com/.

[12] Masashi Toyoda and Masaru Kitsuregawa. Creating a Web Community Chart for Navigating Related Communities. In *Conference Proceedings of Hypertext 2001*, pages 103–112, 2001.

[13] Masashi Toyoda and Masaru Kitsuregawa. Extracting evolution of web communities from a series of web archives. In *Proceedings of the Fourteenth Conference on Hypertext and Hypermedia (Hypertext 03)*, pages 28–37, August 2003.

Appendix: Companion–

Companion– takes a seed page as an input, then outputs related pages to the seed. It first builds a subgraph of the Web around the seed, and extracts authorities from the subgraph as related pages.

First, it builds a vicinity graph of a given seed, which is a subgraph of the web around the seed. A vicinity graph is a directed graph, (V, E), where nodes in V represent web pages, and edges in E represent links between these pages. V consists of the seed, a set of nodes pointing to the seed (B), and an another set of nodes pointed to by nodes in B (BF). When following outgoing links from each node in B, the order of links in the node is considered. Not all the links are followed but only R links immediately preceding the link pointing to the seed, and R links immediately succeeding the link. This is based on an observation that links to related pages are gathered in a small portion of a page.

To each edge, it assigns two kinds of weights, an *authority weight* and a *hub weight* for decreasing the influence of a single server. The authority weight is used for calculating an authority score of each node, and the hub weight is used for calculating a hub score of each node. Companion– uses the following weighting method proposed by Bharat and Henzinger [2]: (1) If two nodes of an edge are in the same server, the edge has the value 0 for both weights; (2) If a node has n incoming edges from the same server, the authority weight of each edge is $1/n$; and (3) If a node has m outgoing edges to the same server, the hub weight of each edge is $1/m$.

Then it calculates a hub score, $h(n)$ and an authority score, $a(n)$ for each node n in V. The following is the process of the calculation, where $aw(n, m)$ and $hw(n, m)$ represent the authority weight and the hub weight of the edge from n to m, respectively.

Step 1. Initialize $h(n)$ and $a(n)$ of each node n to 1.

Step 2. Repeat the following calculation until $h(n)$ and $a(n)$ have converged for each node n.
For all node n in V, $h(n) \leftarrow \sum_{(n,m)\in E} a(m) \times hw(n, m)$
For all node n in V, $a(n) \leftarrow \sum_{(m,n)\in E} h(m) \times aw(m, n)$
Normalize $h(n)$, so that the sum of squares to be 1.
Normalize $a(n)$, so that the sum of squares to be 1.

Step 3. Return nodes with the N highest authority scores.