Prototyping Open Hypermedia Extention on LivingOM

Tomohiro Oda

Apr 29, 2002

Abstract

1 Statement of the Problem

In a community of interests, people with different specialities of practices collaborate to persuit a common interest. It is important to develop a shared and well defined vocabulary during collaborations because different practices often come with different terminologies.

LivingOM is a hypermedia system[1][2] which supports development of common vocabulary in a community. Using LivingOM, users of a community can start with a set of nebulous definitions of terms and then elaborate it into well-understood common language. The "Concepts" of LivingOM employs automatic linking function. Artifacts in LivingOM's repository can be augmented by automatic linking to concepts' definition pages. A user of LivingOM can find a term in a document and check the term whether or not its definition is valid for the user. If the user is not comfortable with the definition, the user can modify the definition. As such modifications happens, users in a community can virtually negotiate definition of a term to finally have a definition of the term which can be satisfiable by all members.

On the other hand, creativities of a group is not carried out only by internal activities, but also involves external sourses and activities, for example, browsing web pages provided by other groups of similar interests and introducing them to other members. In the current implementation of LivingOM, a user has to up-load an external artifact into a repository in order to get automatic links to concept definitions. Because many activities of users are sometimes carried out information spaces out of LivingOM, extending the LivingOM's autolink function to directly handle external artifacts can widen its applications.

2 Rationale

A number of hypermedia systems are studied and developed to support collaborative activities. Hypermedia systems can dynamically generate appearances of a document using hyperlinks among documents and user models including contexts[1][2][3]. LivingOM[4][5] is a hypermedia system which support users to develop shared understandings such as terminologies, documents, personal informations, discussions and links.

In this project, two prototypes of open hypermedia extention to LivingOM have been developed. One is a proxy server named lomScope and another is entity on web space named lomMirror. While the current implementation of LivingOM requires a user to up-load an external artifact into a repository in order to get automatic links to concept definitions. This limitation can bring inconvinience when a user is reading documents of other groups. The two prototypes enable a user to use artifacts in LivingOM even when the user is reading a document out of LivingOM's repository. In this report, the two systems will be described and discussed.

3 lomScope

3.1 Technical Approach

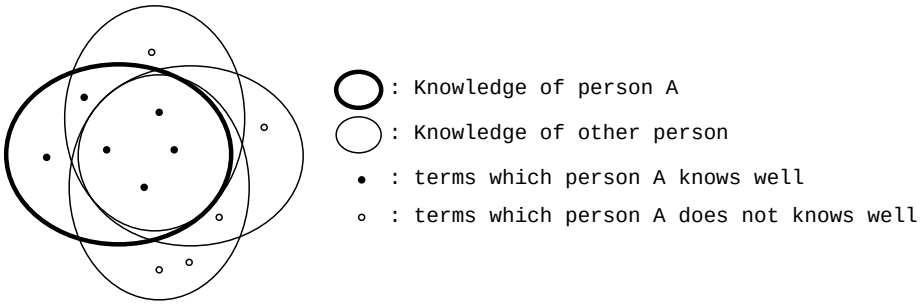

Figure 1: Symmetry of Ignorance in terminologies

The figure 1 shows that each member of a community has own vocabulary. LomScope takes advantage of difference of vocabulary coverage by individuals.

LomScope is a proxy based hypermedia system which augment a HTML document using artifacts in LivingOM's "Concepts" repository. All HTTP requests and the HTML documents from webservers are passed via lomScope.

LomScope assumes that it serves for single user and develops a user model according to HTML documents which the user has ever read. The user model is a histogram of words. LomScope determines whether or not the user already knows a word well and provide different looks of links. The current prototype use the histogram to change colors of autolinked words according to the histogram, because the user may expect different role of autolinking according to familiarity to words.

For example, assuming that a reader is already familiar with Java programming and is reading a paper on Java systems. The reader may first want to know which his interested words appear in which part and it is reasonable behavior that the reader looks for his favorite Java terms by keyword search. The first item can support such activities by highlighting keywords which the user is already familiar with.

On the other hand, when a reader is reading a paper related to his particular work but is not in his speciality. For example, when a Java programmer who is not familiar with 3D graphics is building a 3D graphic application on Java, and the Java programmer is reading a webpage titled "Introduction to 3D graphics with OpenGL." In this case, autolinking terms of computational geometry and 3D graphics, which the reader is not familiar with, can be helpful to the reader's learning activity.

The lomScope can thus be an adaptive system to single user and further possible adaptive features based on user model will be discussed later as a future work.

3.2 Description of the System

LomScope is a proxy based hypermedia system for single user implemented in Python[6]. It accepts any HTTP request from a web browser, relays the request to the webserver designated by the request, and insert links according to "Concepts" repository and "User Model".

The "Concepts" repository reads all terms and URLs of term definition pages from LivingOM and caches them into local memory. This repository can be considered a community model because "Concepts" represents which terms people of the community uses to mean what.

On the other hand, the user model in lomScope is a histogram of words which the user has ever read. It represents which terms the user is familiar with and which the user is not. As the user read webpages though lomScope, lomScope develops the user model and reflect it into HTML documents returned to web browser.

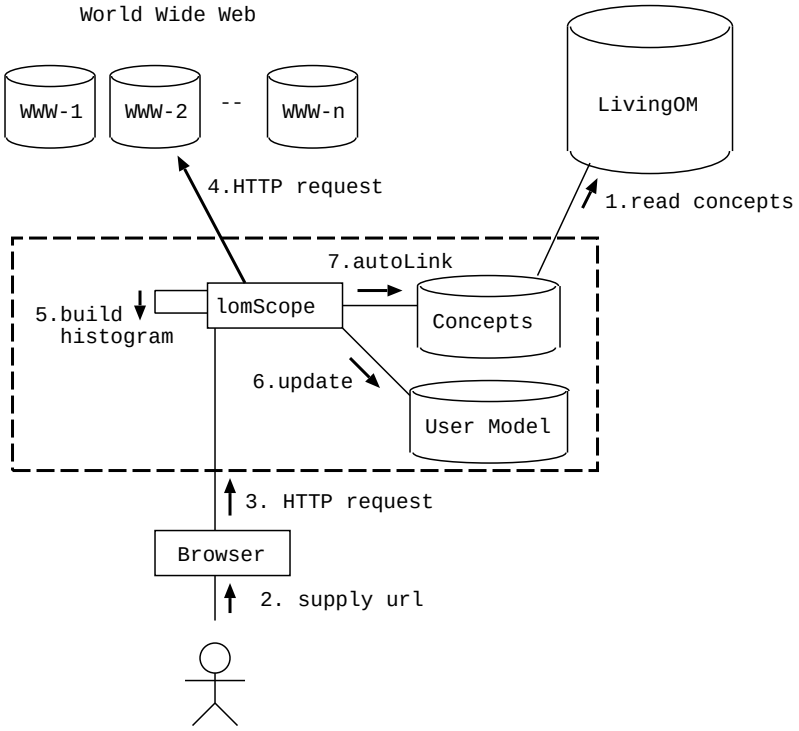

Figure 2: Basic Design of lomScope

The followings are basic functions of lomScope.

1. read terms and URLs of their definition pages from a LivingOM's "Concepts" repository.
2. accept a HTTP request from a HTTP client(web browsers)
3. send the accepted HTTP request to other webservers and receive HTML documents
4. traverse the received HTML document to build a histogram of words
5. add the histogram to another histogram which represent
6. insert links into received HTML documents using terms and definitions' URLs
7. return HTML documents to the HTTP client

3.3 Description of the System Behavior

A user can specify two modes of lomScope. One selection is enable/disable autolinking so that the user can read the original HTML document if necessary. Another selection is enable/disable word separation. A keyword can be a substring in a larger word, and some languages including Japanese and Chinese do not have word separators. Therefore, word separation should be on user's choice.

Figure 2 shows a sequence of functions to be invoked.

- LomScope first updates the "Concepts" repository and the user model, and wait for HTTP request from web browser.
- When lomScope accepts a HTTP request, it relays the request to the webserver specified in the HTTP request, and receives a HTML document.
- LomScope updates a histogram of words in the received document and save it into a local disk.

- If lomScope's autolinking is enabled, lomScope have the "Concepts" repository insert links into the received HTML document.
- LomScope adds settings control buttons into the resulting HTML document, and then returns the document to the web browser.

By settings control buttons, users can always change enable/disables and also update the "Concepts" repository from LivingOM.

3.4 Evaluation of System

The following is a brief evaluation of lomScope.

- Performance: lomScope can process HTTP request and HTML documents in reasonable response time.
- Reliability: lomScope sometimes stuck in middle of process. Supposed causes will be listed later.
- Accuracy: With word separators, lomScope can perform acceptably accurate keyword matching in HTML documents. Without separators, lomScope often make wrong links which results in developing wrong user model.

During the prototyping, I found difficulties to implement lomScope. These difficulties often make lomScope stuck in middle of process.

- HTTP is becomming complex and behaviors of webservers and web browsers are not well unified.
- HTML documents are not well uniformed. Authors often use wrong tags to make visual arrangement.

3.5 Potential Furthor Development

- Use of similarity matching techniques to supply more information into HTML document.
- More kinds of links to LivingOM's pages to encourage users to reflect the result of reading into LivingOM.
- HTTP handling and HTML processing should be more robust.
- Shared histogram of words by a community may be useful to embed more information into HTML documents.

4 lomMirror

4.1 Technical Approach

The lomMirror is an entity on Zope webserver[7] which redirects user's HTTP request to another website and add links to concept pages in LivingOM. The lomMirror is supposed to be shared and used by members and visitors of a community.

4.2 Description of the System

A lomMirror is an entity on Zope webserverk which has three attributes: (1) a file name of lomMirror(the last portion of URL of lomMirror), (2) a URL of a target website to mirror, and (3) URL of LivingOM.

Using lomMirror, members of a community can virtually share external sources and develop their common language in LivingOM. Although LivingOM can copy and incorporate documents from external websites into its repository, lomMirror does not make copy of the documents. Because external sources are outcomes of other groups,

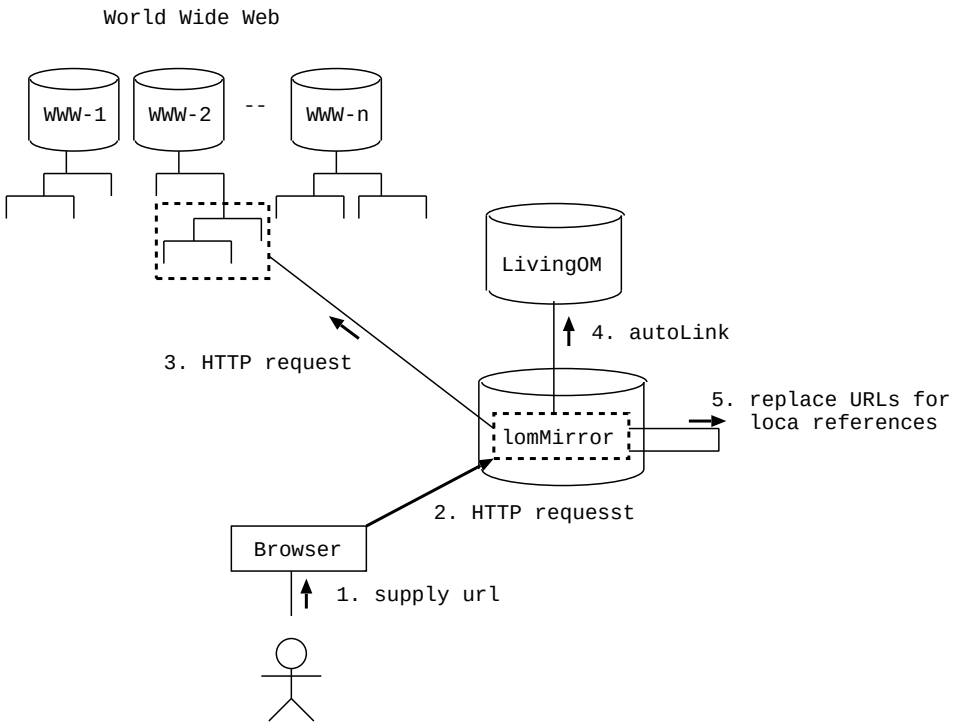

Figure 3: Basic Design of lomMirror

it may be modified and improved as time goes by. Members of the community can see latest versions of documents and incorporate ideas in those documents to develop and use their own shared language. This advantage can be significant when mirroring dynamic websites like swikis.

A community can also use lomMirror to publish their documents to external people. They can use web authoring tools to create a fancy looking website and create a lomMirror targeting their website to make their language understandable by other people.

4.3 Description of the System Behavior

In the current prototype, a user specifies the three attributes to create a lomMirror on Zope. Assuming that a lomMirror is created by the following parameters,

- URL: http://hostname.my.domain/mirror
- target URL: http://webserver.somewhere.else/site/
- LivingOM URL: http://livingOM.my.domain/livingOM/

a user can access to lomMirror in the pattern "http://hostname/mirror?target=¡url¿", where a relative path in the target URL. Every access to the lomMirror will be automatically relayed to the target website "http://webserver.somewhere.else/site/¡url¿. The lomMirror receives the original HTML document, throws it to LivingOM to insert links to the LivingOM's concepts pages. The resulting HTML document contains links to other HTML documents in the target webserver. LomMirror replaces "hrefs" attributes of a-tags and other link elements to direct the destination to the lomMirror instead of the target website. Then the lomMirror returns the augmented HTML document.

4.4 Evaluation of the System

The current implementation has the following problems:

- lomMirror sometimes fail to relay HTTP request depending on target servers.
- lomMirror someimes fail to insert links into HTML documents.
- lomMirror does not replace "redirect" response in HTTP.

All problems above are implementation issue and can be improved in further development.

4.5 Potential Further Developments

In addition to the problems described in the previous section, there are possible future works for lomMirror.

- Incorporate community model (shared histogram of words) described in the Potential Future Developments of lomScope.
- Use of similarity matching techniques to supply links to similar documents in LivingOM's repository.

5 Remarks

This project has been carried out in collaboration with Taro Adachi and Jonathan Ostwald.

References

[1] Grønbæk, K. and Trigg, R.H., "From Web to Workplace: Designing Open Hypermedia Systems", The MIT Press, 1999

[2] Anderson, K. and et al., "Chimera: Hypertext for Heterogeneous Software Environments", the European Conference on Hypermedia Technology in Edinburgh, Scotland, on September 22, 1994

[3] Bailey, C., El-Beltagy, S. R. and Hall, W. (2001) Link Augmentation: A Context-Based Approach to Support Adaptive Hypermedia. De Bra, P. (Ed.): Third Workshop on Adaptive Hypertext and Hypermedia, Arhus, Denmark 2001

[4] "livingOM", http://webguide.cs.colorado.edu:9080/livingOM

[5] "LivingOMDocumentation", http://webguide.cs.colorado.edu:9080/livingOMDoc

[6] "Python Language Website", http://www.python.org/

[7] "Zope.org", http://www.zope.org/