Enriched Content: Concept, Architecture, Implementation, and Applications

by

Hung-Hsien Chang

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

New York University

May, 2003

Professor Ken Perlin

Dedication

To my parents, for their extreme patience and supports.

To Ya-Ya, for its “understanding”.

Acknowledgements

It has been a long journey. A page or two of descriptions do not do justice to the people who have assisted me over the years; however, that is why they call such thanks an acknowledgement.

For New York University, I thank my advisor Ken Perlin who have given valuable guidance at critical moments, over the years he showed me his fun computer graphics demo and we shared a few child-like geeky moments, and of course not to mention the influence of his more incurable optimism (I thought I was incurably optimistic.) My reader committee have been supportive: Alan Siegel who has patiently and constantly given me advice on both academics and life, Ernest Davis whose "extreme" encouragement and enthusiasm are crucial while his scrutinizing my work kept me on my toes. My other two thesis committee members are generous and understanding, to say the least. Richard Cole, who I believe, is surprised to see my change, and Dennis Shasha is enduring to my ever changing mind in the last minute.

I also thank Chee Yap, Ricky Pollack, Ravi Boppana, Dan Melamed, David Geiger, Ben Goldberg, Bud Mishra, Joe Spencer, late Robert Paige, Arthur Goldberg, Jack Schwartz, Eero Simoncelli, Zvi Kedam, Dennis Zorin, Victor Schwartz, Vasilis Vassalos, and Deborah Weichenberg. All of them have benefited me.

There is no graduate student who could finish his degree without the support of office administration. At Computer Science: Anina Karmen, Rosemary Amico, and Don Freda are very helpful. Thank Maria Petagna for many interesting conversations. At Mathematics Department: Reeva Goldsmith assists me on Latex. At Center for Advanced Technology: I thank Madrid Tennant, Kevin Feely, Abren

Agai, and Debbi Baum. At Faculty Technology Center, where I worked while finishing my degree, I thank Robyn Berland, Vincent Doogan, Richard Malenitza, Keith Adams, and Frank Leoprosti from ITS. Special thanks to librarian Carol Hutchins for many pleasant conversations. At NYU Writing Center: Sally Stratakis-Allen, Wendy, Nicole Wallack, Susan, Kathy Graber, M.J, Alice Fasano, Eric, and Tom Kitchen are helpful to improve my writing. without them, my dissertation would not be descent enough.

For University of California at Santa Barbara: I thank Alan Konheim who introduced me to the world of secrets: cryptography and was very supportive at my Berkeley intercampus period, and late Ronald Book for his computational complexity seminar.

For MIT: I thank Mike Sipser, Silvio Micali, and Marvin Minsky for inspiring me. Regrettably, I did not follow Marvin's advice to "stick around" in the media lab.

At Berkeley: I thank late Eugene Lawler who treated me like an on-campus student rather than an intercampus student, Richard Karp for his encouragement and lecture, and Manual Blum, who have generously taken me as his student, for advice and the unique "learning" experience at Berkeley.

At National Taiwan University: I thank Principal Chen Sun who saw a "possibility" in me and wrote me a recommendation letter.

Friends are important.

At Berkeley: I thank Vassilis Papavassiliou and Dan Garcia, for their candidate feedback on whatever I say, at UCSB: Douglas Chang for his friendship and his help. At New York University: Antti Pihlaja, late Henning Biermann for his 'high'

kindness and friendliness, Basu Saugata, William Casey, and Toto Paxia for enduring my never ending bugging on them. I thank Xiaodong Fu and Fangzhe Chang for their Window programming advice, Peter Friz for many pleasant lunches and 3 mins mathematics, Mary Elizabeth Liber Benac for her kindness and listening, Teresa Kochis, Didem, and Jeanea for their candor. Outside the academics: I thank Paul O'Reily and Danielle Glennon. At home: Kwai-Kwai, Lily-Lily, and Solina have accompanied me all this time; Chai-Chai and Chun-Chun have forced me consistently staying upstairs to work. I thank Yukihiro Matsumoto (Matz) for the scripting language "Ruby".

This dissertation, for those whose shoulders I stand on, it is my gratitude; for those who will stand on mine, my honor; for those whose toes I have stepped on (none, I believe, but if any), my apology; and for those who may have stepped on mine, my answer.

The journey goes on...

Abstract

Since the debut of the World Wide Web, Web users have been facing the following problems:

Extended Semantics When we read or study a digital documents, be it a web-page or not, we often run into contexts in which we have stop to start a search on these context. It causes interruption and costs time.

Reverse Hyperlink When we visit a web page, we might be curious about what other hyperlinks point to the visited page. These links would most likely be of related interests. Can we get the "real time" information about what other pages are pointing to this page?

Version Control Many of us have been frustrated and even annoyed when the hyperlink that we follow gives us a "404 not found" or that the retrieved web-page whose content is entirely different from the one we have bookmarked. Could we also have access to the past versions even if the hyperlink has been removed or the content has been changed?

Composition Assistant Writing is not an easy task. We labor to structure a body of text, sort out ideas, find materials, and digest information. We wish there is an automated service that can associate the context we have produced with other contents (on the Web) and bring these web contents to us for reference.

In this thesis, we provide an unified framework and architecture, named enriched content, to resolve the above problems. We apply the architecture and

show how the enriched content can be used in each application. We demonstrate that this method can be a new way of writing add-on functions for various document applications without having to write individual plug-in for each application or to re-write each application. We also briefly discuss possible future development.

Contents

List of Figures

List of Tables

Part I

Concepts, Architecture, Implementations

Chapter 1

Enriched Content

1.1 Problem Scenario

Nowadays web browsing has become an integrated part of our perceptual and cognitive processes. We frequently access and acquire new information and knowledge through the web. Digital documents also play an important role in daily life. Almost all documents have digital copies. The following are some scenarios in our daily digital document experiences. They can be classified as either reading scenario or writing scenario. Extended semantics, reverse link, and version control are about problems we encounter while reading, and composition assistant is a problem we encounter while writing.

Extended Semantics When we read or study an article, be it a web page or a downloaded document for other applications, we often run into contexts in which we would like to know more. In the past, we could either go to the library or call an expert. Now, we could launch a search on the Web. However, we still need to start up a browser, enter a search engine url, type

in the query, and wait for the result.

This is cumbersome and sometimes very time consuming. Wouldn't it be nice if the system knew what we wanted and looked it up for us? While this is appealing, the "intelligent" engine that is capable of carrying out this task doesn't exist. So what can we do?

Reverse Hyperlink On other occasions, when we visit a web page, we might be curious about what other hyperlinks point to the visited page or to the page we are about to visit. These links would most likely be of related interest. In fact, that is how Google does its ranking and searching. It would be useful to see, in "real time", how many hyperlinks point to a page and to be able to pick from one of the "reverse links" and visit.

Version Control Many of us have been frustrated and even annoyed when the hyperlink that we follow gives us a "404 not found" or the document whose version is entirely different from the previous one we have read. Shouldn't there be a mechanism that automatically manage websites and make various versions of their pages available? Shouldn't a user have the option to choose from different versions at the point of click?

Composition Assistant Writing is not an easy task. We labor to structure a body of text, sort out ideas, find materials, and digest information. At the same time, we look for new angles and insights. If there is a service that can associate the context we have produced with other contents (on the Web) and bring them to us, then they can be sources for our inspiration. This automation can be instrumental to our creative experience.

The aforementioned scenarios happen to different users at various times and we can't help but ask the question: can these problems be solved? Most importantly, can we have the same services on all applications, so no matter what authoring or browsing software we are using, we could have the above functionalities at the point of click? In other words, could these capabilities be provided within a single framework?

Enriched content architecture is set to provide an answer for this quest.

1.2 Reformulating the Quest

Dipping deeper into what we have encountered, we reformulate the activities behind the scene and show how enriched content could give new insights and solutions. Our first example, "extended semantics", is an application of enriched content.

1.2.1 Extended Semantics

The current content of website is designed by the authors. Usually they have specific purposes: for an education institution, it will generally describes its faculty, current research, past projects, courses homepages and more. Most of the websites ignore the user's needs (intentionally or unintentionally) for other correlated information.

The needs from users vary by their semantic intention (information seeking intention or media consumption intention). Each individual has a different semantic network associated with his unique experience. One thing enriched content tries to do is to add a menu of links for each word or phrase to extend its semantics. Given

any word, such as "Graphics", there are several associations for this word. In the realm of computer science, we could come up with terminologies like: computer graphics, computer graphics researchers, computer graphics research centers, computer graphics design, computer graphics designers, computer arts and so forth. We can have these options as links to other resources so the user could have one click-drag access. We do this for each word or phrase on the page. Enriched content will partially automate this process.

Ideally, the choices of menu should rely on the context but current computational linguistics programs are not intelligent enough to provide a satisfactory answer yet. Of course, we can increase the closeness of the sought information by a simple contextual analysis on the sentence where the word and phrase resides. Then a query is formed and submitted to resources. The resources could be a manually made database or information extracted from the existing database or search engine. They also can be constructed and tailored for a specific purpose.

1.2.2 Bridging the Gaps between Ideas

When we are working on a long article, we sometimes have the experience of staring at the computer screen (the ceiling, or the wall) when nothing is coming out of our heads. "Writer's block" it is called. But we also have the experience of not being able to quickly dictate ideas flying from our minds. The phenomena is as if there is a "concept dam" that holds our ideas, and then, all of a sudden, it lets the concepts pour out. For instance, in some situations, we are struck by an idea that links different elements of concepts together into one single entity.

Presumably, our working mind is constantly seeking connections between what

we want and what we know. If more plausible cues can be brought to our attention, the greater the likelihood that our concept dams will open or that "lightening ideas" will come forth. These cues act as stepping stones in our creative process. Cues can come from the content produced thus far. Or it can be an indirect reference such as a reverse link. Visiting a webpage by following the reverse direction of a hyperlink may tell us the "cause" of the association; we may discover other correlated webpages in the neighborhood of the link on the visited page.

Thinking backward is just as important as thinking forward. In terms of viewing webpages, traversing backwards on links can be useful to our information gathering.

1.2.3 Evolutionary Thought Pathway

Our thinking is constantly changing without our conscious awareness. Articles and documents are the snapshots of thoughts at that instant of creation. As thoughts progressively evolve, we create different articles which could vary in theme, even if they are about the same subject matter.

As a reader, we do not have the the privilege of peeking into the fascinating (or hairy) authoring process such as creating, adding, rephrasing, pruning and so on. We wish to be able to access at least the stabilized versions at each stage. This would provide us a better understanding of temporal progress along the line of thought.

Selecting and viewing different versions of the same document helps these activities. Enriched content architecture could facilitate this functionality.

1.3 Context Unit Type

If we step back and analyze the problem scenarios carefully, then we will find that we are dealing with content that comes in two types. One is text type; the other is link type. [1]

These are within the analysis of the aforementioned problem scenarios. For multi-media documents, we may find static pictures, sound clips, and videos. While images with HTML tags can be annotated, other types of media type in general do not fall into our consideration. Off line documents with embedded pictures will have "mark location" problems as we will see in a later chapter.

1.3.1 Text Type

Text type content is text based. For example, a word, such as 'excellent', is a text type. A sentence, sentences, a paragraph, paragraphs, or a full document, are all text type. Text type comes in plain text format. We do not distinguish the variation of font, font color, and its size. We treat all text type the same: they are just text.

1.3.2 Link Type

Link type has two forms. One appears in the text and is usually prefixed by "http://", for example, "`http://www.cs.nyu.edu`". The other form is what we encounter on a webpage. It is usually underlined and highlighted with different color, for example, **Computer Science Department of NYU**. If we look at

[1] There are other media type. For instance, video, images, and audio. But these are not in the scope of our discussion.

the source of the HTML, then we can find 'http://www.cs.nyu.edu'.

1.4 Enriched Content Process

We use extended semantics as an example to illustrate how the enriched content process is carried out. The reasons of this architecture are explained in Section 3.3. The description reflects the steps taken in the following picture:

(a): Client side:

User is using an application which could be a browser or an editing tool; she sees an unfamiliar word or phrase that she would like to learn more about. Moving the mouse over the word, she then clicks over the text. The client side module receives a click message and gets the target context from the focused application.

(b): Network:

The context data is sent through the network to a pre-assigned server where it is further analyzed and interpreted.

(c): Server side:

The server module studies the request and accompanying data, extracts context, and formulates queries for web resources. The server launches the queries to various web resources, requesting relevant information.

(d): Server side:

Numerous webpages and documents are collected as a result of the queries. The server filters and summarizes qualified feedback, and composes the summary in

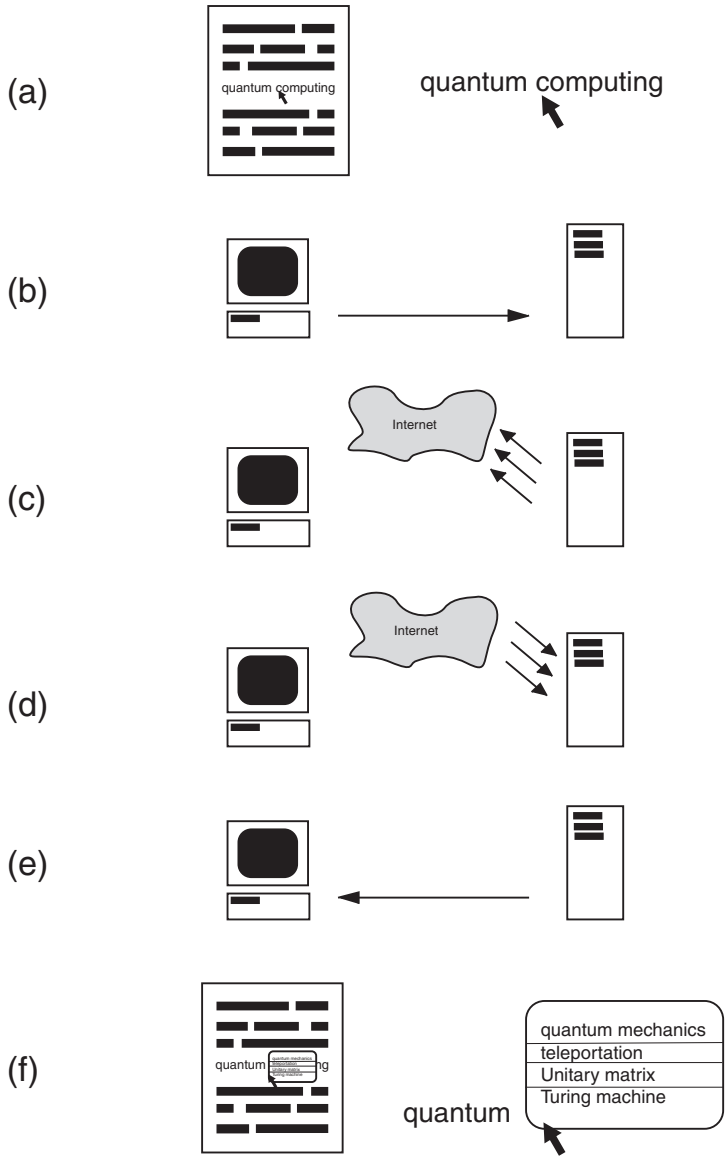


Figure 1.1: Enriched Content Process

menu format.

(e): Network:

The server passes the menu format containing list of resources to the client side.

(f): Client side:

The client module displays a selection menu on the point of click. The user can choose items from the menu or simply discard them by clicking the 'close' button on the menu. If she selects an item on the menu, the client module will launch a browser opening the url address associated with the menu item. Only keywords are displayed on the menu item; the url address associated with the keywords are not shown.

This completes the cycle of an enriched content service process.

1.5 Motivation and Related Work

Here is a brief illustrations of related work. A more detailed comparisons with tables and descriptions are at the end of this section. We can classify the problems by context unit type. Extended semantics and composition assistant are text type, reverse link is link type, and version control can be either text type or link type. Enriched content is related to Web augmentation and open hypermedia. [Bouvin 00]. One can view enriched content as a way to facilitate and ease the third-party application integration which is a difficult problem in open hypermedia [Davis 94, Whitehead 97].

1.5.1 Extended Semantics

Our idea for extended semantics is the result of contemplation about what can be done for users after the visit to the Internet World Exhibit in late 1996. At that time, our research focus was not on the World Wide Web. This research started in early 1999, and in the same year, a series of related technologies started to emerge. In March 1999, 3rdvoice.com, an online annotation service; then gurunet.com(now named atomica.com) which is a form of extended semantics, followed by flyswat.com, zapper.com, and finally Microsoft Office suite's smart tag. From past research, we can also find related work on hyperlinks for digital library [Adar 95]. There are projects like Haystack [Adar 99] whose goal is to learn from personal computing history and storage information in order to give better suggestions on urls and related documents. Xerox Parc also has fluid links [Zellweger 98] which acts like Xlink and it is an inline hypermedia annotation. Fluidlink is also applied to open hypermdia [Bouvin 02].

Hypertext has long been envisioned by Vannevar Bush (1945) [Bush 45], Douglas Engelbart [Engelbart 84] and Ted Nelson [Nelson 92]. Many ideas have been thought about but has never been realized. Hypertext implementations exists before the World Wide Web. Projects such as Hyperties [Shneiderman 88] and Apple's hypercard are two examples. There are other hypermedia systems such as NoteCard, Neptune, KMS, Intermedia, Augment and so on. Consult Halasz and Schwartz [Halasz 94] or Carr [Carr 94] for lists of reference. However, linking through a global network is what makes the World Wide Web powerful, as was realized by Tim Berners-Lee [Berners-Lee 00]. Multiple destination linkings were proposed long ago but there is no implementations on the browser yet.

1.5.2 Reverse Link

Reverse link has been done partially and "locally" as Google started its reverse link service in late 2000. Even though they have archived the searched sites, it is a timed archive. In other words, the data is not persistent nor "real time". An archived page gets updated every 2+ months but updating also depends on how "important" (page rank) a page is. Reverse link received new interest at the annual WWW conference in 1996 and 1999; implementations and suggestions were made [Pitkow 96, Chakrabarti 99.1].

1.5.3 Version Control

In WWW Conference and Digital library Conference, version control is covered and exploited [Pettengill 95, Vitali 95, Simmonson 98] . There are version control management commercial products [Hummingbird, Merant] .

1.5.4 Composition Assistant

Watson system [Budzik 99] and Kenjin [Kenjin] focus on assisting writers. Watson supplements the writers with related documents. Watson works on Microsoft WORD and displays a list of recommendation in a mini-browser. The recommendation is collected from a list of search engines and news source.

1.5.5 Related Work Comparisons

The comparison tables are constructed in chronological order column by column with the earliest work first. First column is the label column. Here is an enumeration of what each label means:

[**Year**] The year when the project is shown to the public, or the year the paper is published.

[**Name**] The name of the project, or the name of the author(s).

[**Applicable software**] What software can the service be applied to? For example, if the table cell indicates "Emacs" then the service applies only to the application "Emacs". "Almost All" means the services are applicable for almost all document applications. "IE" stands for "Internet Explorer".

[**Activating method**] How a user activates the service? "ALT + Left click" means to press the ALT key and the left mouse button at the same time. "Change of URL" means changing the URL of a browser either by following a hyperlink or by the user entering the new URL manually. "Start of program" means the service starts running when you activate the software (in the applicable software category).

[**Contextual unit**] This means the "size" of the context unit. For text, it can be a word, a sentence, sentences, a paragraph, paragraphs or a full document.

[**Context retrieval**] How can the context unit (for example, a word) be retrieved after users activate the service? "API" is Application Programming Interface.

[**Contextual analysis**] The computation and process that are used to "understand" what exact context a user is pursuing.

[**Content delivery**] Once the service gathers the extra content (enriched content), how is it delivered to the user? "Client Sever" will involve network

communication while “Local” will not.

[**Result presentation**] The way the enriched content is displayed. Is it shown through a menu or a browser?

Extended Semantics: Text Type

We list only works where the client side is more involved. Server based applications, such as WebWatcher [Armstrong 95] and Balabanovic [Balabanović 95], are not listed here. There are two tables for extended semantics.

Table 1.1 shows services that work only on specific document applications. Other hypermedia systems involve no direct Web integration are not considered here. To see a list of these systems, consult Bouvin [Bouvin 99, Bouvin 00].

Distributed Link Service [Carr 95] is an extension of Microcosm [Fountain 90]. It uses the Web standard protocol to hyperlink selected context with other contexts. Links (their source, destination) are stored remotely and they are provided as a service.

In 1996, Remembrance Agent [Bradley 96] is presented by Bradley Rhodes from the MIT media lab. It works only in the Emacs environment. The service starts when a user uses emacs. Emacs supports strong programming tools and one can use it to change the Emacs environment. It is easy to get a buffer for the current window and from the cursor location; one can find the strings surrounding the cursor. Remembrance agent finds useful words by throwing away stop words [2] It then uses these keywords to find matches in the pre-indexed database. The database includes the corpus of “Boston Globe”, archived e-mails, INSPEC and

[2]Stop words are words such as “it”, “be”,“with”, and so on.

Year	95	96	97	99	00
Name	Distributed Link Service	Remembrance agent	Leitiza	Index-Based Hyperlinks	Margin notes
Applicable software	Netscape	Emacs	Netscape	IE	Browser
Activating method	Start of program	Start of program	Change of URL	Start of program	Change of URL
Contextual unit (text)	Manual selection	Cursor neighborhood words	Webpage	Webpage or manually selected text	Webpage
Context retrieval	Selection	String buffer	API	API	Proxy
Contextual analysis	?	TF, IDF, database	TF, IDF, history	Search Engine	TF, IDF, database
Content delivery	Client Server	Local	Local	Local	Client Server
Result presentation	Webpage	Subwindows	Multiple windows	Mini-browser	Frame within a page

Table 1.1: Extended Semantics 1

the like. The match for relevance is based on a formula using a variation of term frequency(TF) and inverse document frequency (IDF) [Salton 75, Salton 89]. The term frequency of a word in a document is the number of times a word occurs in a document over the total number of words in the document. Take the word "agent" as an example; if "agent" appears 15 times in a document of 300 words then the TF of "agent" with respect to this document is 0.05. The inverse document frequency (IDF) of the word "agent" in a collection of documents is the inverse of the ratio of the number of documents containing the word "agent" over the total number of documents in this collection. For example, if we have a collection of 500 documents where 50 of them containing the word "agent", then we have 10 ($= 1/\frac{50}{500}$) as IDF. Remembrance agent will provide two extra subwindows within the Emacs. One shows the keywords it has extracted; the other lists the relevant documents from which users can choose to read. Remembrance agent is named "Remembrance" because it has saved database (such as archived email) as its memory.

Leitiza [Lieberman 97] is by Henry Liebermann, and it is also from the MIT media lab. Leitiza works only on Netscape because it uses Netscape's APIs. It comes with three windows. The first one is the content browsing window, the second one shows the webpage under consideration, and the third displays the actual recommended webpage list. Leitiza does a depth-first search starting from the currently viewed webpage. It collects and analyzes the pages from this search, and it scores each collected page. If a page's score is higher than a recommending threshold, this page will be shown in the suggested list window. The score of a page is computed from the TF and IDF. It also has memory about a user's viewing history as a reference for relevance matching. Leitzia is also a demonstration where

an information agent and interface agent are integrated.

Index-based hyperlink [Hartman 96] can be viewed as a precursor of Xlink. It defines implicit hyperlink, which resembles the DLS but allows indices. A phrase can be associated with a group of URL options. It can have indirect indices and hierarchical indices. Indirect indices are indices whose content is located elsewhere; hierarchical indices are indices can contain other indices. It removes the duplication of inserting the same links content when a single term occurs several times in one document.

Margin notes [Bradley 00] works on all browsers. It shows recommended list of webpages with keywords in the same window as the webpage. The list is on the right side of the page. It has changed the layout of the page by adding a new HTML tag such as <**FRAME**>. The content from the web server goes through a proxy where the content is intercepted, analyzed, and further processed. After the contextual analysis, Margin notes injects new HTML tags and content to add new information to the original page. It does not change the original feel and look of the page except by adding a "Margin notes" (thus named) on the page.

Table 1.2 shows services that have more broadly applicable software. Most of them are proprietary, and there is no way to determine how they do contextual retrieval. However, from testing the service we can tell the size of the contextual unit (word, sentence, or paragraph) they are serving. Most of them identify only a single term. [Finkelstein 01] claims that Zapper "looks around" the words in the neighborhood of interest, but they do not indicate how it is done and what the coverage of the "neighborhood" is. They also claim to have a Semantic engine which performs contextual analysis, but the details of this engine are not clear.

Year	99	00	01	01	02
Name	Atomica (Gurunet)	Flyswat	Zapper	Smart tag	Enriched content
Applicable software	Almost all	Almost all?	Almost all?	Word, IE, Outlook, Excel	Almost all
Activating method	ALT + Left click	ALT + Left click	CTRL + Right click	Mouse over	ALT + Left click
Contextual unit (text)	Single term	Single term	Neighboring words	Single term	Single term, sentence, and so on
Context retrieval	?	ActiveX	?	Programmed	Simulated messanging, Graphics context
Contextual analysis	Search engine, database	Search engine, database	Semantic engine(?)	?	Search engine
Content delivery	Client server	Client server	Client server	Client server, apps.	Client server
Result presentation	On spot mini-browser	On spot menu	On spot menu?	On spot menu	On spot mini-browser

Table 1.2: Extended Semantics 2

The response time for this service is not ideal (less than 10 seconds). It indicates clearly that a good and efficient contextual analysis is difficult.

Smart tag is a Microsoft product; it differs from other services in that the context availability is programmed. You have to "write codes" for each phrase and word that you want as a keyword (key-phrase). Further, it does not show the existence of extra content until user does a mouse over on the word. If the underneath word has information, then a menu icon will appear above the word. The user can drag down the menu and select an item for viewing. A common characteristic of the systems in this table is that they all provide the client-server model as part of the content delivery method. Enriched Content has a specific implementation that captures context unit at a designers' designation.

Reverse Link: Link Type (Table 1.3)

The programs that use reverse links are of two types; those based on search engines and those based on a specialized server.

A search engine has archived the pages and their links; therefore, it has direct information about which pages are linked to other pages. A database query will find the reverse link information directly. Alta Vista and Google are examples of this group.

In 1996, Pitkow et. al. [Pitkow 96] proposed a new protocol and a server maintaining the links information. The reason to run an extra server parallel to the existing web server is for two functions: one is for reverse link information, and the other is to solve the dangling link problem (hyperlink whose destination page has been moved or removed). The updated link information is broadcasted distributedly. Because it is decentralized, it is possible that some of the information

Year	96	97	99	00	02
Name	Pitkow et. al.	Alta Vista	Chakrabarti et. al	Google	Enriched Content
Applicable software	Browser	Browser	Browser	Browser	Almost all
Activation method	Change of URL	Manual request	Change of URL	Manual request	ALT + Left Click on link
Hyperlink retrieval	Automatic	Submitted request	Automatic	Submitted query	Proxy, Javascript
Links acquisition	Real time, local	Archived	Real time, local	Archived	Real time or archived
Service requirement	Protocol, extra server	CGI	Protocol	CGI	CGI
Result Presentation	Separated Window	Browser	Separated Window	Browser	Separated Window
Note	Not implemented	Not operating now	Real time not implemented		

Table 1.3: Reverse Link

may not be spread quickly enough to all servers. The design was proposed but never realized.

Chakrabarti et. al. [Chakrabarti 99.1] did implement the reverse link function but only for the archived link information. They also proposed an extended protocol for accessing the real time reverse link information. In their implementation, three windows are used. One shows the current page, the second displays the navigation history, and the third lists reverse links of current by viewed pages.

In enriched content, we will "force" the reverse link information to be provided even if the web server serving the page does not install the reverse link module. When there is no reverse link service from a web server, enriched content will simply resort to the archived information (for example, from Google) and pass it along to the client.

The main difference between the archived and the real time is the following: in the archived approach, all reverse links will be shown; if the search crawlers have indeed crawled all the web pages. In the real time case, the reverse link is only shown when there is at least one "reference" instance from a browser's visit. When a browser follows a link to a page, the server maintaining the page will record the visit in a log. In other words, page A may be hyperlinking to page B but if this A-to-B link has never been traversed then it will not be in the "real time" log.

Version Control: Link Type and Text Type (Table 1.4)

We consider a "dangling link" (or broken link) a subproblem of version control; therefore, papers related to *fix* a "broken links" [Ingham 96, Creech 96, Pitkow 96] are not listed here.

Using a software management package to control webpage versions appears

Year	95	95	98	02
Name	Pettengill et. al.	Vitalli et. al.	Chakrabarti et. al.	Enriched content
Applicable software	Browser	Browser	Browser	Almost all apps.
Version control mechanism	RCS	VTML	Extended URI	CVS
Service requirement	URI, server side module	Client side module	Server side module	Server side module
Result display	Browser	Browser	Browser	Mini-browser

Table 1.4: Version Control

as early as 1995 when Pettengill et. al. [Pettengill 95] proposed a solution that includes a URI [3] implementation with a server-side module. This solution makes each request for webpage into a CGI request.

Vitalli et. al. [Vitali 95] headed in another direction. They used a markup language to describe various changes on a document and suggested that the integration of a software module on the client side could realize this version control function. Thus development of the client side makes version control possible.

Chakrabarti et. al. [Chakrabarti 99.1] implemented a version of the proposal by Pettengill et. al. and recommended an extended URI protocol. The software demonstration is, however, not "real time" information as it requests its link information from a search engine archive [4].

[3]URI: Universal Resource Indicator

[4]They use DirectHit.com each engine for reverse link information.

In Enriched Content, version control can be applied to both link type and text type when we use the same version management system for documents (text type) as well as webpages (link type). Applying the version management system to all documents requires modification on some system commands. If we only want to manage webpages, then we will only need to install a server-side module, there is no need for a client-side module or any new protocol. In both webpages and non-webpage documents enriched content can show not only the current content in the browsing window, it can also display a list of past versions in a separate mini-browser. The reader can choose any version from the list and display the past version on the browsing window.

Composition Assistant: Text Type (Table 1.5)

Remembrance agent works not only on reading mode, it also works on writing mode. It will look for matched documents using the keywords it has found in the neighborhood of the cursor.

The Watson project [Budzik 99] was first designed as composition assistant (on writing) and it is later extended to the Microsoft Internet Explorer (on reading). Watson uses the API for the WORD and obtains contextual information (full text), then it runs simple heuristics to weight the importance of words using cues such as the size of font, color of font, whether or not the words are part of the title, and so on. Watson then used these gathered weighted words to construct queries to search the Web. It then gathers and removes duplications from the collected information. Because it integrates contextual analyses and search functionalities, we consider it a "Fat Client". It is expected that as contextual analysis becomes more sophisticated and search volume surges, consumption of computing resources

Year	96	99	01	02
Name	Remembrance agent	Watson	Smart Tag	Enriched content
Applicable software	Emacs	WORD	WORD, Excel, Outlook	All editing apps.
Activate method	Start of program	Full document, manual selected paragraph	Term	term, sentential, paragraph, or full document
Context retrieval	String buffer	MS application automation	Pre-programmed	Simulated messanging
Context analysis	TF, IDF	Simple heuristics	Keyword match	Simple keywords extraction
Content delivery	Local	Local	Local, Client Server	Client Server

Table 1.5: Composition Assistant

and network traffic will increase. These will make this approach less favorable. Watson analyzes the article produced so far. In order to narrow down to smaller text units, such as a paragraph, it requires users to select it manually.

Enriched content can focus its context unit at a designated size (term, sentence and the like.). It delegates the contextual analysis and search analysis processes mostly to the server side.

1.6 Contribution

The World Wide Web produces a wide channel for information. The problem is it also opens up chaos of different plug-ins and formats. For every application, its programmer offers ways to write a plug-in. You see SDK (Software Development Kits or 'Standard' Development Kits) all the time. This is especially serious when it comes to the browser. Everyone wants to write a plug-in for the browser so it can work with her application.

Enriched content architecture shows how extended semantics, reverse link, version control, and composition assistant can be done in a single framework rather than designing diverse plug-ins for each application program. This framework/architecture can also be applied to different GUI [5] system. By applying enriched content, we can minimize various complexities in our digital document experience.

1.6.1 Unified Framework

The solutions to the aforementioned problem scenarios are provided under the same unified framework: uniform contextual information acquisition with respect

[5]GUI: Graphics User Interface

to context type and consistent client-server architecture/methodology for different GUI systems.

Uniform Treatment for Context Unit Type

Enriched content uses a single mechanism to obtain contextual information (text type or link type) from the client side. In other words, regardless of the application we are using, as long as the context unit, text type or link type, remains consistent, we will deploy the same mechanism to obtain the text context (or link context). We do not have to write an individual plug-in for each software application from different vendors to enforce the mechanism.

Consistent Architecture/Methodology for Different Windowing System

Enriched content employs client-server architecture to simplify content delivery and presentation. The client-server model is not new but given our presentation and information collecting method, enriched content will get extra benefits such as lightweight add-on modules, the reuse of contextual analysis, and delivery on demand, which lowers network traffic. These extra benefits will be described in later chapters. Uniform treatment on the context unit type, especially text type, is possible only when we are using a GUI system.

The inspiration of such a unified framework is derived from two ideas: back to basics and aspect programming.

Back to Basics

Software development has 50+ years of history. Since the PC movement in the 80's, many software environments and paradigms have been proposed and built to ease

the programming process. Terms like functional, module, imperative, logic, object-oriented, pattern, and component programming give programmers many leverages for different tasks. Tools and artifacts are designed to help programmers; software is implemented on top of layers of packages. Programmers can wander in the forest of programming paradigms and components: they can grow a tree(program) by watering it, fertilizing it, trimming it, and shaping it. While general programming difficulty has been removed, there is another kind of barrier: programmers see the trees but not the forest.

What we propose is : "back to basics" - the "seed".

It doesn't matter what kind of tree it is; it starts from a seed. It doesn't matter what type of GUI system it is; it uses event messanging and handling. To make a better tree within the same GUI, we need to design it at the seed level - change its DNA so it will grow to what we like it to be. No need to prune, trim, or shape.

For a GUI system, we investigate the message routes we want to create and alter the message paths in order to add new behaviors to applications. We need to change the organization of its "seed" level - message paths.

Aspect Programming

From a different perspective, enriched content architecture is like aspect programming [Aspect]. Aspect programming allows you to change or to add behaviors to a class method before and/or after the method executes its own code. It does so without physically adding codes to the method definition. Aspect programming accomplishes this task by determining which classes are going to accept the "ad-

vice", and defining what "advice" is. Just like the aspect programming's advice, enriched content architecture intercepts the event message, pre-processes it, and then passes it to the application. It adds no codes to the original application's program but it changes the behavior of applications.

1.6.2 Adaptability of Text Unit and Necessity of Client Server Model

In the implementation of enriched content, we explicitly describe how to capture each context type. Because of this particular implementation, we show how it can be adapted for different text type units (term, phrase, sentence, and so on.). In this dissertation, we also explain in Section 3.3 why the client-server model is necessary for contextual computing: the computation and processes through which the application understands the exact context a user is pursuing.

1.6.3 Minimizing Complexity in Process, Version, Data, Re-coding, and Learning

By applying enriched content we can minimize various complexities, namely, search process complexity, documents version complexity, data volume complexity, program re-coding complexity, and user learning complexity.

Search Process Complexity

As described in section 1.1, we waste time on laborious process that could be automated. Moving the mouse to the url entry space, typing in the search engine's url, entering in the query string, and waiting for the result. This procedures take

unnecessary time. In all three cases there are a series of interruptions to the working mind.

The process complexity of the above activities should be minimized. By initiating extended semantics, we could shorten the procedure, minimize working interruptions, and optimize our time usage.

Version Complexity

Many of us have had the frustrating experience of not being able to find information we have written down while reading a computer document or webpage. Even if we do find the note, it might not be the one we were looking for.

This happens to online documents as well. We read an interesting article and bookmark it. One day, we go back to look it up for reference but it is not there any more or the page has a different content than the previously visited one. We have to visit its main homepage and do a search for the page if the website even provides the search function. If we are lucky, we can find it; other times, we can not. This problem makes bookmarking seem useless and it means we may have to save a hard copy locally on our PC instead of bookmarking it. Saving a hard copy can fill up our local storage very quickly if we are avid online readers and note takers.

The complexity of versions has plagued online users since the day the WWW came into existence. In the academic world, this problem is serious when we try to refer an online paper that is no longer at its assumed location. In the business world, a missing crucial corporate data hyperlink can cause delays in operation progress and cost a fortune. By installing version control we can eliminate version complexity and ensure that hyperlinks are built to last.

Data Volume Complexity

If we use a single single word as a query and send it to a search engine, you are most likely to have a long list of returned results. To narrow down the list size, we usually add more keywords. In order to add the right keywords, we can conduct a simple contextual analysis and compose a better string for search engine. Extended semantics will give us an advantage over the simple search by enabling us to construct a context-sensitive query in order to obtain a more plausible result.

Multiple versions of the same documents without organized management can be very confusing. There are times we do need different versions; for example, for record keeping and comparison. Version control can reduce the data complexity.

Program Re-coding Complexity

As we remove the need to add new codes onto each application to which we want to provide enriched content, we also eliminate the need to download various plug-in formats adapted for this function.

User Learning Complexity

Extra content is presented in ways users are used to. For text type content, it is presented in an on-spot mini-browser; for link type, it is displayed in another window. There are no surprise elements. There are no visual gadgets that user have to learn in order to use enriched content. Users do have to remember that the enriched content is available and that they can click on the word and/or phrases to get information directly. This habit hopefully will be re-enforced by the brought benefits.

1.7 Road Map

In this chapter we have introduced what enriched content is about, its contribution, and some of its applications. Chapter 2 discusses contextual computing from the large to the small. In Chapter 3, we will describe the principle and notions of architecture. Chapter 4 details general implementation issues. From Chapter 5 through Chapter 8, more refined technicalities are explained for each application. These chapters end with each application's benefits and properties. For demonstrative purpose, we have implemented both extended semantics and obliged reversed links. Version control, and composition assistant are described in this thesis but not implemented. Finally in Chapter 9, we succinctly discuss future work possibilities and trends.

Chapter 2

Contextual Computing: a Preliminary Examination

In order to understand contextual computing on the Web, we will present a preliminary examination of the subject. We will look at the basic construct of the Web: webpage and website, describe the panoramic view of the Web, and discuss the following questions: How do modern search engines operate? How do we find related information? And why is search within context important?

Contextual computing is closely associated with "pervasive computing" or "ubiquitous computing". In this thesis we concentrate mainly on the context in text or hypertext [Lieberman 00]. Contextual computing and its relations with multiplicities of devices, thus pervasive computing, is not discussed here.

People usually think of the World Wide Web (or the Web) as a database source. However, while this characterization is accurate, it only captures one facet of the Web. Unlike a database which usually is a group of data organized in pre-designed table formats, the Web has links and the interactions between links and networks

is dynamic. Links are produced mostly for meaningful association. The Web contains an array of traditional databases whose information is searchable through the form submission.

A web site can have many faces. It can act as a portal which is a starting point to other locations, for instance, Yahoo. It can be a personal website (or webpage) which features an individual web presence, with information about her; such as her interests and musings about the world. Private and public organizations can construct corresponding virtual entities. The Government has websites to tell the most recent tax and law changes. Museum websites can show their recent exhibit and display their calendar of events. Universities may showcase their departments and educational prestige in order to attract prospective students. University departments can have course homepages for classes. Businesses can have intranet for business processing.

In essence, webpages are usually classified as either hubs or authorities. Hubs are resource pointers, and authorities are reputed webpages (in their own field/-topics). A webpage can be both a hub and an authority; it is a matter of degree.

2.1 The Characteristics of the Web: Superstructure, Macrocosm, Microcosm

In early Web history, there are studies concerning its quantitative growth (number of pages, sites, and links [Web]). Recent studies focus more on the connectivity, its usage on searching [Kleinberg 98, Brin 98, Bordoin 01], the size [Oclc 01], the form and shape of the Web [Kumar 00.1, Kumar 00.2] and how it grows [Barabási 01,

Barabási 99].

In this thesis, we define superstructure as the quantitative and qualitative properties of the Web as a whole. Macrocosm will be about the inter-page (thus inter-site) relations which are analyzed and used by search engines [Kleinberg 98, Brin 98]. It was not until recently that researchers started scrutinizing the microcosm, the page [Chakrabarti 01.2], and its relevance to the global structure. We will briefly discuss intra document relationships; we believe that much future research will be moved to this area because microcosm lies at the core of contextual computing. Inevitably, natural language processing will play a key role in contextual analysis.

2.1.1 Superstructure: Size, Shape, and Connectivity

Size

There is an estimate of the number of websites on the Web [Oclc 01]. In 2001, the Web had about 8.745 millions websites, which contained 8.443 millions unique IP address. [1]. The total number of webpages is believed to be around 4 to 5 billions (circa year 2000). From comparing the current growth rate and the one in the mid and late '90s, we can see that the growth of the Web certainly has slowed down (Table 2.1).

Form and Shape

Kumar et. al. [Kumar 00.2, Broder 00] draws a picture for the approximated Web: it looks like a bowtie (Figure 2.1). There are IN, OUT, and SCC (Strongly

[1] A website can have multiple different IP addresses.

Year	No. of Websites	No. of Unique Websites	Types of Unique Websites		
1997	1.570	n/a	Public	Private	Provisional
1998	2.851	2.636	1.457	0.315	0.864
1999	4.882	4.662	2.229	0.790	1.643
2000	7.399	7.128	2.942	1.494	2.692
2001	8.745	8.443	3.119	2.078	3.246

Table 2.1: Size of the Web

(numbers in millions)

Year	1997–2001	1997–1998	1998–1999	1999–2000	2000–2001
Sites	457%	82%	71%	52%	18%
Unique Sites	n/a	n/a	77%	53%	18%
Public Sites	290%	82%	53%	32%	6%

Table 2.2: Growth Rate of the Web

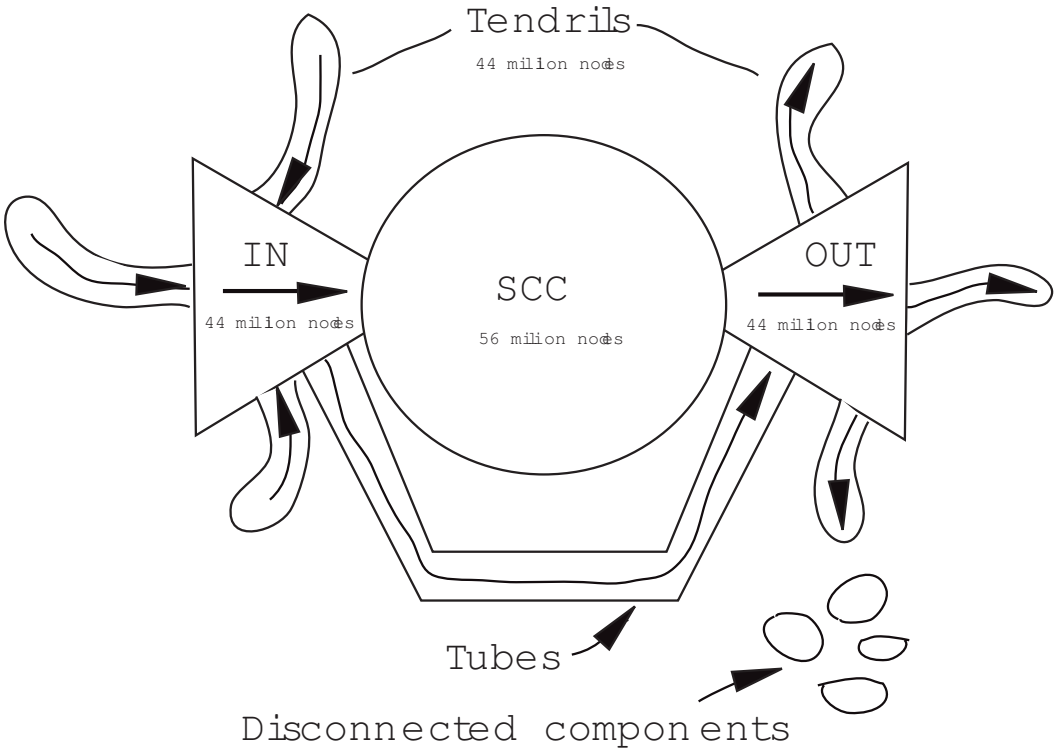


Figure 2.1: The Form and Shape of the Web

IN, OUT each has 44 million nodes. SCC has 56 million nodes, Tendrils has 44 millions nodes, and Disconnected has about 17 millions nodes.

Connected Component) parts for this bowtie. A node in IN has directed paths to the nodes of the SCC and a node in OUT can be reached by some nodes from SCC. SCC is a component where between any two nodes there is a directed path. In random graph terminology, SCC is a "giant component," named after its interwoven and colossal characteristics. Other parts include Tendrils, Tubes, and Disconnected Components. A node in Tendrils can reach nodes in IN or can be reached by nodes in OUT. Tubes are some sparse channels connecting directly from IN to OUT, and Disconnected Components are simply isolated nodes from the previous parts. SCC has around 56 millions nodes and Tendrils; IN and OUT each has around 44 millions nodes, and Disconnected Components have about 17 millions nodes. The pictures are constructed from the webpages they collected in Oct 1999. It is not the entire Web; however, we expect that the Web carries similar characteristics.

Connectivity: Power Law (or Zipf's Law)

Researchers have found out that the Web connectivities follow the power law [Barabási 01, Barabási 99] or Zipf's law [Zipf 49]. It uses a model that includes a 'scale-free' model which starts from m_0 already-existing nodes. At each time step a new node is introduced; it is connected to m nodes. The probability of connecting to node i with connectivity k_i is

$$\Pi_i = \frac{k_i}{\sum_j k_j}$$

Under a large time frame the connectivity distribution follows the equation

$$P(k) = \frac{2m^2}{k^3}$$

The experiment on a limited Web sample shows that the Web follows the power law, and it does not resemble the random graph $G_{n,p}$ where n is the size of the graph and p is the probability of the existence of edge between any two nodes [Alon 01][2]. Given the $P(k) = k^{-\gamma}$ as the probability of a document with k incoming (or out going) links, Barabási et. al. found that the "average indegree" of a node, [3] γ_{in}, is around 2.1 while the "average outdegree", γ_{out}, is close to 2.45. [4] An interesting coincidence is that the physical connectivity of the Internet, as opposed to virtual connectivity between webpages, also follows the power law with $\gamma = 2.48$ [Faloutsos 99].

There is a "copying phenomena" [Kumar 00.2] that web authors tend to copy a portion of a list from other pages as part of their content. This also contributes to "the popular gets more popular" phenomena.

Diameter, Fault Tolerance, Vulnerability

A study indicates that the diameter, i.e. the mass majority distance between any two connected nodes on the Web, is around 19 [Albert 99]. This phenomena is usually termed as "small world phenomena" [5] because despite the colossal size of the Web, two nodes are only 19 links away. The power law network exhibits

[2]Random graph has long been studied for academic interests. It was not designed to model the Internet.

[3]"Average indegree" and "outdegree" here are exponents in a power-law distribution, not real average.

[4]As experiments are conducted under sampled Web which is not the whole Web, there is a discrepancy in the estimate of γ [Albert 00, Kumar 00.2] but in all cases, the average indegree is smaller than the average outdegree

[5]The original "small world phenomena" experiments were carried out by Milgram for social networks [Milgram 67, Travers 69] in the 60's.

characteristics such as fault tolerance. Even if a large amount of arbitrary links are moved away from the network, the general connectivity is not affected much. In other words, most of the connected components of the graph will still be connected. But the Web also has vulnerability, due to the power law, there will be a few 'winner' nodes with a large number of links. Removing a series of these nodes will cause the graph to be disconnected or cause the increase of its diameter [Albert 00]. Bharat et. al. [Bharat 98] shows that the "average number" of hyperlinks of a webpage is around 7.

2.1.2 Macrocosm: Linkage Semantics and Search

We consider the two algorithms using linkage analysis. Kleinberg's [Kleinberg 98] and Brin's [Brin 98] are two main forces behind the modern search engines which model the Web as a directed graph. In their analysis the eigenvector of the matrix representing the Web plays a pivotal role in locating useful resources.

Algorithm from Authoring Perspective

Kleinberg's algorithms of finding hubs and authorities is based on linkage analysis. If a page (site) has more authority over a topic, then there are hyperlinks pointing to it. A page is a good hub if it has many hyperlinks pointing to reputed authorities. The higher the authority of a page, the higher the authority of the pages it points to. It is a diffusion process. A good hub will point to many reputed authorities. This definition sounds recursive, and it is. In fact, the computation of the algorithm itself is recursive, and the claim is that the computation will converge and will become stable.

An outline of the algorithm is as follows: first, the algorithm issues a query request to a conventional search engine (e.g. Alta Vista, in Kleinberg's case) and collects a base set of urls related to the query, then it expands the base set to include all urls such that every url in the expanded set is either pointing to or is pointed to by the url(s) in the set. Then it computes the following formula iteratively until its values do not change significantly any more. The algorithm is also known as HITS (Hyperlink Induced Text Search).

$$a(i) = \sum_{\{j:j->i\}} h(j) \qquad\qquad h(i) = \sum_{\{j:i->j\}} a(j)$$

where $a(i), h(i)$ are authority and hub value of node i respectively. $j \rightarrow i$ indicates that node (page) j has a link to node (page) i. Each page is treated as a node.

The definition can be seen from an author's view. A page author has made a decision to put links on the page; the decision is based on her evaluation of the links. It is not surprising that the result of the recursive computation is relevant as it uses many fragmented human intelligences (human intelligent evaluation) to produce an online consensus. But it is not invincible; webpage author could design links so it makes the algorithms less reliable. For instance, two densely connected web sites can increase each other's hubs and authority values quickly. An improved algorithm has been proposed [Bharat 98] to fend off this short-coming by evening out weight on the hub and authority values for links from the same site. Other than the "malicious" webpage author, the estimated human consensus on link evaluation is based on the online population, to be more precise, the online author populations. Words and phrases are written by online authors do not reflect the general human communication consensus. In online populations you could expect

more technology savvy authors. A simple example is that the word "Apple" will most likely mean the computer company but not the daily fruit we consume. This issue is addressed in section 2.2.2.

Algorithm from Both Author's and Reader's View

Webpages have authors; they have readers as well. Brin and Page [Brin 98] use this readers concept to design their page rank algorithm which can locate high quality relevant pages. The cusp of their algorithm resides in how it simulates a reader's browsing habit. Here is their assumption: a page has higher rank if the rank accumulation of pages pointing to it is high, and if a page has a higher rank, then it will contribute higher value to pages it points to. But a page may point to many other pages, so a straightforward way of distributing a page's rank is to evenly share it among the outgoing hyperlinks. This method is used to compute a page rank, which is the sum of contributions of incoming pages. PageRank algorithm also adds a human factor; it assumes a reader may do a random walk on the Web. Hence there is a probability that the reader will simply visit a page without following a link. Here is their formula:

$$PageRank(n_i) = d \times E(n_i) + (1 - d) \times (\sum_{\{j:n_j->n_i\}} \frac{PageRank(n_j)}{OutDegree(n_j)})$$

where d is a damping parameter to simulate the random walk. $OutDegree(n_j)$ is the number of hyperlinks of page n_j. E is the unit matrix. The division comes naturally when you think that a page equally distributes its $PageRank$ to its references, even though contextually, each hyperlink may weigh differently by its author. In other words, when an author builds a webpage with hyperlinks, she

has an evaluation for each link. However, this information is private and cannot be easily accessed. Therefore, evenly distributing a pages's rank is a plausible approximation. Page rank algorithm does use the author's view in that pages' links are constructed by authors. One important concept of this algorithm is its emphasis on the page ranking from a reader's view.

Comparisons

Page rank can be seen as a global distribution of all the pages. It indicates the importance of a page purely from linkage analysis. Rank values are computed before a query is made: they are pre-processed. Google, which uses the page rank method, matches up the text query from their text index barrels, evaluates the proximity of matched terms in returned documents, does a rank merge, and displays the result in descending final rank order. It performs well despite the "postponed" textual matching. From a practical point of view, it makes sense, since for each query, there is a linear order of quality which is computable from statistics. The question is why do other pages reference this page? It must be, or has a very high probability, that this page contains important information about the topics it presents. Even if this page may not be the highest ranking of all pages on the web, it can be the highest ranking for the requested topics.

On the other hand, HIT algorithm initiates the text query at the beginning of the searching process. It asks a "tradiational" [6] search engine, e.g. Alta Vista, for a set of pages containing the query term(s). It separates webpages into two categories: hub and authority. Each page has two values, one for the hub and the

[6] "Traditional" search engines are search engines that use no linkage analysis. Most of them use the term frequency and the like. [Salton 89]

other for the authority. It is pointed out [Kleinberg 00] that competitive websites are less likely to link to each other. As HITS has included the hubs in the searching process, the algorithm will discover competitive webpages even if there is no direct hyperlink between them.

2.1.3 Microcosm: Zooming In

Webpage is the basic unit of the Web. The search engine's main purpose is to "understand" each webpage in order to serve users' requests. In future when a user types in a query, e.g. "What is the best java in the world?", the search engine will search the previously computed "understanding" of webpages and give us a list of good resources. So far, researchers have paid very little attention to the webpages other than processing them for term indexing, term frequency, and collecting its hyperlinks and anchor texts.

Chakrabarti et. al. [Chakrabarti 01.1, Chakrabarti 01.2] started looking at the role of a webpage in linkage analysis. Rather than treating a webpage as one single unit, they break it down as if it is a small web itself and used a modified HITS algorithm to evaluate both the authority and hubs values.

From our perspective, an effective tool will be a contextual program capable of understanding human language and therefore, the webpages. This will fall into the field of natural language processing. Ultimately, when natural language processing does comprehend human languages fully as a human being would be, then the Semantic Web's formal approach will be either an automatic middle process for the purpose of storage representation and transaction, or it will become obsolete.

2.2 Problems Raised by Sense Ambiguity

Research has shown that the average search query by a web user is about $2 \sim 3$ words [Jansen 98]; with short vocabularies, it is hard to ask a search engine to give us the exact answer we have in mind. In fact, we give limited denotational information and ask for connotative precision from a search engine.

2.2.1 Many Faces of Meaning: 20 Questions Game

We can start a guessing game with a friend. First, we think out a "thing" (anything): she has to guess what we have in mind by asking us questions, and then we answer her questions using only one word. We repeat this Q&A until she gets the answer. It is like playing 20 questions. We may find our answer eventually; however, human correspondents are much smarter because they can infer and filter out the possible results.

While 20 questions may be enjoyable during leisure time, Web users are in no mood to play games when searching for information. Besides, this game is simply not fair or efficient, even for a human, let alone for a search engine.

2.2.2 Sense Ambiguity and Descriptive Precision

Part of the problem of a search engine's inability to capture the full spectrum of a query's meaning is rooted in the sense ambiguity of human language. We will discuss some of its aspects in terms of polysenymy, hypernymy and hyponymy, meronymy, synonymy, troponomy, and entailment.

Polysenymy

Polysenymy are words that have multiple meanings. Contextual information is hard due to linguistic ambiguity. Take the word "character", for instance, it could mean an alphabet when we discuss language. It could hint at the trait of a personality when it comes to psychology. It could imply a role in a play when it is mentioned in show business, and yet it is commonly associated with the function that transforms one Abelian group to another when it appears in a commutative algebra context.

One frequently asked query "Jaguar" (the big cat, starting from [Kleinberg 98]), when submitted to Google in late 2001 [7], returned many results, but we did not see the pages related to "cat" until the 3rd page. By then, we have gone through 20+ urls including links to "cars" and "car model", a bridge named "Jaguar", and an Atari game device. The first "Jaguar" referring to cat does have a high quality description of the animal, and it even tells us how to distinguish between Jaguar and Leopard. It is not surprising given that with the query "character", we do not see its relevance with the Abelian group even when we browse through 20+ pages (200 urls) from Google's returned results. However, when we type "character Abelian," we get the correct answer right away.

Hyponymy and Hypernymy

Hyponymy is words representing a special case of some other words; hypernymy is words representing a generalization of some other words. In one scenario, if a

[7] Since then, Google has improved the returned results. It is because either the Web has changed the rank by itself or Google has tuned their rank merging algorithm a bit. We can get "cat" information on the 2nd page.

user wants to know more about cars but she submitted the query "Porsche", then search engine will most likely return countless of pages related to "Porsche" rather than cars. On the other hand, if she wants to know more about "Porsch," but she knows nothing except its look, then "car" would be a better start for a query. However, the success of this generalization approach depends on the existence of a thorough and organized website/page for the specialized knowledge/information domain because search engine cannot find non-existent webpages.

Meronymy

Meronymy is words that are part of concepts (objects) represented by other words. A user may want to search a part of an airplane, for example, the trim tab, but she knows not much about the structure of the airplane. She can only enter search for "airplane" or "airplane structure" in the hope that she could find the needed information.

Synonymy

Synonymy are words that are similar to other words, for instance, "car" and "vehicle." A user can also make the "mistake" of placing the "wrong" query term simply because she is not familiar with the jargons of a profession, or she is not well versed in the natural language (English) used in query. For instance, instead of searching for "pipe," she searches for "tube." Despite the fact that they are synonyms, they may be used differently in a specific community.

	Jaguar (cat)		Java (coffee)		Apple (fruit)		Character (Abelian function)	
Web	20+	Y	100+	N	70+	Y	200+	N
Image	1st page	Y	100+	N	1st page	Y	n/a	n/a
Directory	1st page	Y	90+	Y	70+	Y	100+	N
Discussion Group	100+	N	100+	N	1st page	Y	100+	N

Table 2.3: Search and Database Type

Note: Y means "found", numerical indicates examined URLs. Search conducted in early 2001.

Troponomy

A user could be searching for "horse galloping," but if English is her second language, then she might enter "horse running." People do not gallop. Entailment is another possible confusion, for example, 'drive' and 'ride' are entailment, so are 'marry' and 'divorce'. There is a action sequence. This confusion would occur most likely for non-native speakers. WordNet is a good source to detect these differences [Fellbaum 98], and they may be used in search engines.

2.2.3 Media Type and Searched Database Orientation Matters

Curiously, if we search "Jaguar" through the image search facility of Google, we will see 3 cat images in the first 10 pictures. The first page includes images of cars, cats, and the Atari device. This implies that the search on different media types

will make a difference. An inquiry into the "Group" section of Google, yields many postings on an IBM software named "Jaguar." A directory search on Google will return both "car" and "cat" on the first page. See Table 2.3.

As many products or projects are named after objects with no direct semantic association, you can anticipate similar phenomena. A few other examples would be terms such as "Java" and "Apple." In our experiment on Google, we did not see a true "Apple" (as the one that hit Newton on the head) until 30+ urls had gone by, and "Java", the coffee drink did not appear until 50+ urls had been viewed. Again, we tried image media type. For "Apple" both computer and fruit were displayed on the first page. With "java", however, there was no reference to coffee even we had gone though 200 urls.

A search engine can categorize in advance a few possibilities for a given query and ask users to pre-select the orientation of the search. Inquirus2 [Glover 99] provides a preliminary version of this pre-selection. This process mimics the response from human correspondents. As we have seen in the query word "character," we will be swamped with a lot of pages if we do not specify the field we are interested in. A web search engine will most likely return character functions related to Abelian groups when it searches under constraints; for instance, one can constrain the domain to be in mathematics. In the previously suggested Q&A game, the correct answer "character function on Abelian group" is impossible to gain unless the answer party has majored in advanced math. We even suspect that unless the answer party is aware of the questioning party's mathematical background, it is highly unlikely she will associate "character" with "Abelian group" quickly despite her advanced mathematics knowledge.

Search engines can also categorize the search results. In the query for "Jaguar," it can display category "car," "game" (Atari), "cat," and so on as long as there is sufficient contextual evidence to cluster the resulting urls. One example is to converge the IP sub-domain. In this case, sales.apple.com and fun.apple.com will be converged under apple.com. The number of matched results can also determine if a converging urls process is justified. For example, the result may have a lot of urls related to "cars" and a few about "cats"; in this case "cats" should stand out as a different category.

2.2.4 Educate Users for the Correct Use of Search Engines

The previous section describes situations for users' query. Users not only have to know what they are looking for, they have to be more precise in forming a query. A user has to be educated about how to use the current search engine correctly. This education may not be easy because it is generally expected that the user interface for search engines is supposed to be smart enough to figure this out. In future, when the Semantic Web does take off and when many web authors adopt it as THE standard, then search engines could infer "Porsch" with respect to "car" as "$a - kind - of$" or "$is - a$" predicate. Search engines equipped with this information could provide a host of options for viewing. For example, the inquiry "car" could have a list of "consumer report," "dealer," "showcase," "user group of car," "how to drive a car," "moving vehicles department," and so on. If a search is requested within the context, then an analysis on context can assist pin-pointing the exact search purpose for the "car."

2.3 Context in Hypertext

Now we learned that when a program (search engine) is supplied with contextual information for a query, then it will be easier to correctly locate information. Let us examine what context there is in hypertext.

2.3.1 Knowledge Domain, Role, State of Mind, Environment

In Web computing, to truly capture what a user wants, we have to determine the knowledge domain and profession category of the website or the webpage, the role of the user in this field, and her state of mind. Personal computing history can imply trends and traces and it is good to have them. Sometimes, the users' environment, both physical and computational, has to be taken into consideration. This is obvious in mobile computing when location matters and the limited devices display and higher cost for content delivery make presenting relevant information all the more important. Environmental consideration (such as physical presence and location) would rely on other inputs (such as GPS [8]), which are not in the scope of this thesis.

The state of mind of users correlating with their roles, i.e. user model, is frequently omitted from most analysis. A user visiting a news site may not be just a general news reader; she could be a journalist herself and wants to find out what competitors are doing, or she could be an employee of the website publisher who is verifying whether the webpages are working properly. If a user visits a computational biology website, she could be a beginner who just drops by and want

[8] Global Position System

to know what it is about. In this case, she would be more satisfied with pages of short and quick descriptive facts accompanied with pictures and diagrams. If she is a Genomics expert working on SNP [9] identification, then what she wants is more likely to be the most advanced SNP techniques, updated SNP databases, and recent news in the community. Even a user visiting a company website may have multiple purposes, she could be a consumer, a reporter, a potential investor, a competitor, a provider, or a partner. Many roles are possible, one of the most thorny problem of website design is how to identify the needs of visitors. Conducting a contextual analysis is helpful to untie this knot.

The role of a user is crucial, but most contextual analysis does not and can not put it into consideration. In the previous SNP example, even if the user is a novice, she can be an experienced scholar, and she would like to have a panoramic and encyclopedic view of computational biology rather than a simple paragraph with facts. Regardless of all the possible role variations, when a search query is supplemented with context, such as the sentence and paragraph the query term resides, it can narrow down the possible sense orientations.

2.3.2 Context Type: Text and Link

We focus on our context unit type to see what context cues we can acquire from these two types: text type and link type.

[9]SNP: Single Nucleotide Polymorphisms is essential in identifying the genetical difference between individuals.

Text Type

In the pre-Web arena, in the pursuit of designing a better electronic text [Dillon 94], Dillon emphasized "why", "what", and "how" a reader reads. They classify nine categories of texts: (examples within parenthesis are not the original ones)

1. newspaper (New York Times)
2. manual (Mac OS X User Guide)
3. textbook (Introduction to Computational Biology)
4. novel (Harry Potter and the Scroccer's Stone)
5. journal (Journal of Theoretical Computer Science)
6. catalog (Saks Fifth Avenue Fall Season Sale)
7. conference proceedings (Foundation of Computer Science 2002, IEEE)
8. magazine (Vogue)
9. report (Investment guide for 3G, Yankee's Group)

We can now add more as we use email frequently:

- private email for personal information exchange/courtesy
- business email for deal negotiations
- knowledge discussions
- business email for information exchange/courtesy

There are other documents involving figures, diagrams, and charts such as construction plans, apparatus designs, or spread sheet. These contexts are much harder to be explained because they have a lot of localized and case-by-case information. Even though each may have a standard format for each industry, we do not discuss them here.

The Web has liberated all forms of documents and articles to be published by anyone who has Web access. We will not categorize the webpages using the above classification. No one has successfully categorized of the Web. However, when a search is requested with text unit, such as a paragraph, we can gain insight from the paragraph's keywords to narrow down the knowledge domain.

Among the criteria, knowledge domain, role, state of mind, and environment, it is much easier to decide the domain of knowledge and profession of the website or webpage. It is especially true when the visited website is dedicated to a particular field, for instance, Health.

Link Type

One major difference between printed text and hypertext is the semantic links that have been added by authors directly. These links are a good source for clarifying sense disambiguity. Through these links, we may identify the webpage's domain knowledge; for instance, the destination pages may have been classified under the Yahoo directory, they may reference pages classified by Yahoo or may be referenced by pages classified by Yahoo. Further, the anchor text of hyperlinks on this page are good indicators of knowledge domain. These are all good clues about the orientation of the page context.

For our purposes, we will focus on knowledge domain from context to see if we

can fulfill the multiple needs of user's role in this domain and her state of mind. We consider text or HTML based input and examine these inputs for contextual clues.

Chapter 3

Architecture

The structure, characteristics, and benefits of the enriched content architecture is discussed in this chapter. There are three simple notions: cross system, client-server, and add-on module. One implementation principle: build on existing standards and systems.

3.1 Principle: Build on Existing Standards and Systems

3.1.1 Reinventing the Wheel: When Not to Do it

The principle is easy to understand. But why do we choose to do this?

There are already too many systems and protocols, and too much software around. For each single entity that is used as a mean for communication, there is a wide variety of standards and formats.

Imagine when we hope to send documents from Windows to a Unix system so

that receivers can read it immediately. Or when we want to make TDMA-based handsets talk to CDMA-based handsets? Not to mention when you want to write a single program in C language and hope that it will be compiled smoothly in any operating system.

Observing the pitfalls of existing systems, many software programmers and architects launch ambitious projects; they want to set a new standard, or establish a new paradigm completely from the ground up, with the hope that the 'new standard' can replace the currently used system.

This is not likely to happen if it is not an innovation which can stand alone. A few successful examples in computing history are personal computer and the browser (or maybe we should say browser-server or the World Wide Web). Both have innovations. PC provides a small stand-along computing unit and browsers supply hypertexts via the network. Neither existed prior to their debuts.

Another criticism of starting a new standard is that there are already many standards set to solve parts of the problems. They work very well in their own individual domain. These solutions are not well utilized; although they simply requires a careful design and certain components to integrate them. Much like a car that is assembled, its parts are made by many different providers. So why reinvent the wheel when there is already a wheel and many methods to improve it? The wisdom is to use existing tools and not to reinvent them.

3.1.2 Standards? Which to Choose?

There are standards for practically everything. The original purpose of a standard is to converge the diversities and make the communication process seamlessly.

Standards are made for people to exchange information more easily, but the unfortunate effect is that there are many standards even for a single entity. Also, there is an array of organizations making standards. A short list includes IEEE (Institute of Electrical and Electronic Engineers), ISO (International Organization for Standardization), and also ANSI (American National Standards Institute) and so on. You can multiply each possible standard setting institution in the Unites States by the number of continents, if not countries, and come up with a modest estimation of how many organizations there are in the world. Now imagine the standards they are going to approve and add this to the 'pseudo' standards such as RFC (Request For Comment) documents.

We wish to implement a standard, but which one?

In our definition, the 'standard' adopted is one that is most widely used, rather than the one with a compelling theory or the one with only community verbal agreement. This constraint will narrow down the scope of coverage. In operating systems, there are UNIX, Windows, and VxWork [WindRiver] [1] The former two come with their own GUI environments, X window (Cocoa for Apple Macintosh), and Windows respectively. VxWork is not 'visible' to the general public, but it is fairly ubiquitous in appliances. It dominates the equipment and appliance operating systems. Despite the fact that we can pick the popular top 3 operating systems, the majority usage share still differs according to demographics. For operating systems, the academy (especially engineering schools) use UNIX more than Windows and there are more home and business PCs running Microsoft Windows

[1] Mac OS X which is a version of BSD UNIX has move Apple Macintosh to UNIX camp.

system than any other system. It is well known that there are more UNIX servers than Windows servers and there are more Window-based PCs than UNIX-based PCs (Linux, to be exact).

For our purpose, we choose Window client and Unix Server for demonstration, but our architecture could be applied to other operating systems without much changes as it is based on the object oriented model properties. For network protocol we use TCP/IP and for the World Wide Web standards we follow the W3C's (World Wide Web Consortium) recommendations.

3.2 Cross System

The World Wide Web presents numerous opportunities for application interpolation as we will be able to download and install through the network any software we want. Frequently, applications are made to work well with browsers so that they can be launched within the browser client window. Downloading software is easy and every developer wants to make their software work with browser. Designing a cross system architecture and schema can reduce the laborious process of integrating every single application with browsers. Through cross-system architecture, all applications can deploy the same function without the need for specific modification.

3.2.1 Object Oriented Window System

Modern operating systems running graphics user interface system, such as Solaris, Linux, Microsoft Window, and Mac OS X, all have their own object oriented windowing models. Object oriented windowing models involve message sending

and handling. Users generate messages by moving or clicking the mouse and typing. Messages are sent to the window system which dispatches them to the currently focused application. To enable cross application architecture, we will select the most common messages that are essential in recognizing hyperlink and capturing text.

By intercepting the event message sent from the system before it reaches the designated application, we can change the behavior of the application at the cross system level. In other words, in the enriched content application, we can have the same effect on all applications: be it a browser, a text editor, or an e-mail program and so on. When the mouse is clicked over a text, enriched content will be activated. Naturally, at locations such as the window caption, the scrollbar, and the menu, a mouse click should not evoke enriched content.

3.2.2 Software Component from Window System

Applications are written by programmers who have manifold styles and preferences in their choice of data structures and algorithms over which we have no control. In order to make capturing text possible, we may have to resort to the internal data structure containing the text. Fortunately, window systems frequently come with a default text component that can be used in programming applications. Also, popular software components may be used in application when programmers do not want to reproduce the existing component repeatedly. If programmers are using default text software components that come with hyperlink support, then recognizing hyperlink and text around the location of the mouse is easier. However, there are cases where software developers design their own data structures, and we

have to decipher messages passed from the system, to 'trick' the application and window systems in order to obtain the text information. These techniques will be illustrated in Chapter 4.

3.3 Client-Server

In the enriched content, we design a module to show a menu displaying options for further exploration. This small client module will communicate with the server module which supplies the content of the menu. This module is small in size; it is a thin client. The size of the server module is also small, but the size of the semantic database and context analyzer will vary depending on their sophistication and purposes. For example, if the context analyzer uses WordNet database then the size will be at least that of lexicons collected for WordNet.

The client-sever model is not a new idea. The benefits of employing client-server architecture are freeing client from computational bottlenecks, lowering network traffic, enabling universal data availability, and providing continuous updating of the newest contextual service.

3.3.1 HTTP and CGI

HTTP protocol is the de facto standard of transmitting hypertext on the World Wide Web. It may be cumbersome when it is used for state-changes processes, but we have a simple query-answer session which the protocol can manage.

CGI (Common Gateway Interface) is the common mechanism for conversing with the server. CGI is frequently used in passing parameters and their values to servers in order to get the result from the servers.

3.3.2 Avoiding Computing Bottlenecks and Network Traffic

Enriched content utilizes web data resources and requires heavy data mining on demand. The general client-side devices and the method of accessing networks such as PCs and dial-in modems will consume computing power and generate high network traffic that can slow other background processes. Client server architecture designates heavy duty computing operations to the server side and releases the client from the CPU and network burden, After the server analyzes and filters Web information, it will send in a much smaller packet of (menu) data to the client. The user on the client side can decide what action she wants to take upon receiving the menu display. She can choose to browse one of the recommended items or completely ignore the menu; in the latter case, the computing power on client side is saved, and the server can store its analyzed result without wasting its computational efforts. When in future a query with the same contextual cues recurs, the server can send the menu content directly without repeating the searching-analyzing-filtering process. On both the client and the server ends, we have economically utilized the resources.

3.3.3 Enabling Universal Data Availability

Client-server model makes data available online regardless of where we are. The server acts as a repository for data. The convenience brought by the model is obvious. For example, extended semantics can keep our profiles containing our interests and preferences in order to provide more satisfactory recommendations. If the profiles are on the network, then the servers can identify us upon our login.

Servers can provide us more personalized recommendations by consulting with our profiles.

3.3.4 Providing Continuous Updating of Services

Extended semantics needs to have a precise analysis on the context in question. [2] If the contextual analyzer resides on the server side, then it can be upgraded whenever a new algorithm is discovered. Suppose we had moved all analytic computations to the client side, then constant upgrading on the contextual analyzer will be less likely. Users will have to check from time to time to see if there is a new analyzer and if so, download and install a newer version. By placing contextual computation to the server end, we benefit by getting the newest contextual analysis service at all time. Naturally, there are options to install software that periodically checks if there is new contextual software patch from the server. If there is, then automatically install the new software patch. We do not take this approach because contextual analysis is required at understanding the context in interests on the client side as well as analyzing the information gathered from the network during the answer-reference assembling time on the server side. Also, to make enriched content on the client side a real stand-alone application we have to install the network query-search mechanism to gather from the network the information needed in answer-reference assembling. Massive network query-search operations will cause possible CPU exhaustion and network congestion as described in the previous section. It will also consume more working memory space while still suffer from not getting the newest contextual analysis. We can install contextual

[2] For our definition, a high quality contextual analysis should answer correctly for the questions from a human about the context it has analyzed. None will exist, in foreseeable future.

analyzer on the client side to reduce the network traffic in the case when a whole document has to be analyzed; however, the server side will handle the answer-reference contextual analysis. Therefore, client service architecture is inevitable and it is more beneficial to use this architecture.

3.4 Add-on Module

The client and server modules are add-ons; they are not replacements for existing webservers and browsers. The mechanism is built on what has been widely used; we are not attempting to replace existing Web server, Apache for example, or browser clients, such as Internet Explorer or Netscape. By adding modules on both the server and the client side we can fulfill the functional needs. On the client side, module is added to the window environment. The client side add-on has to be downloaded once; the server add-on can be integrated into the existing Web server, or it can run singly as a service.

3.4.1 Lightweight Module

Because add on modules are built on top of the existing system and standards, they tend to be small in size and consume little memory space. They usually communicate through a limited number of channels with the existing system so the interactions do not complicate the normal processes. Since it is lightweight, the design is simplified.

3.4.2 Composite Menu Presentation

On the client side, we have a menu presentation module which handles the display of menu items. The menu presentation module knows nothing about the meaning of the menu items provided to them. The module simply renders and displays to users the content. In other words, regardless of applications and sources of data providers (extended semantics, reverse link, composition assistant, or version control), the menu presentation module will show whatever are sent from the server. If the server has multiple services, they will be organized at the server end. The client side do not have to be concerned with anything other than presenting the menu. New services can be added without taking the client side operations into account. For example, at the beginning of enriched content service, server can provide only one service such like extended semantics. But later, when server installs a new module such as obliged reversed hyperlink, there is no need to inform or change the client side module. The arrangement of the menu context is made on the server side; the client side will simply display the menu.

Chapter 4

Implementation

The process for accomplishing enriched content include: intercepting events, re-routing window procedures, filtering out and handling intended messages, network communications, server operations, and menu presentations. Interception of events, re-routing window procedures, and filtering out and handling intended messages will be presented here. From Chapter 5 to Chapter 8, we will detail network communication, server operations, and menu preparation and presentations for each application separately.

4.1 Cross System: Text Type

We show how to make the enriched content work for all document applications across the system level. When document applications have an open object model (or component model) that expose its methods and properties, then it can let us tap into the document format structure and make use of it. When there is no open document architecture available or the exposed interfaces are limited, we have to

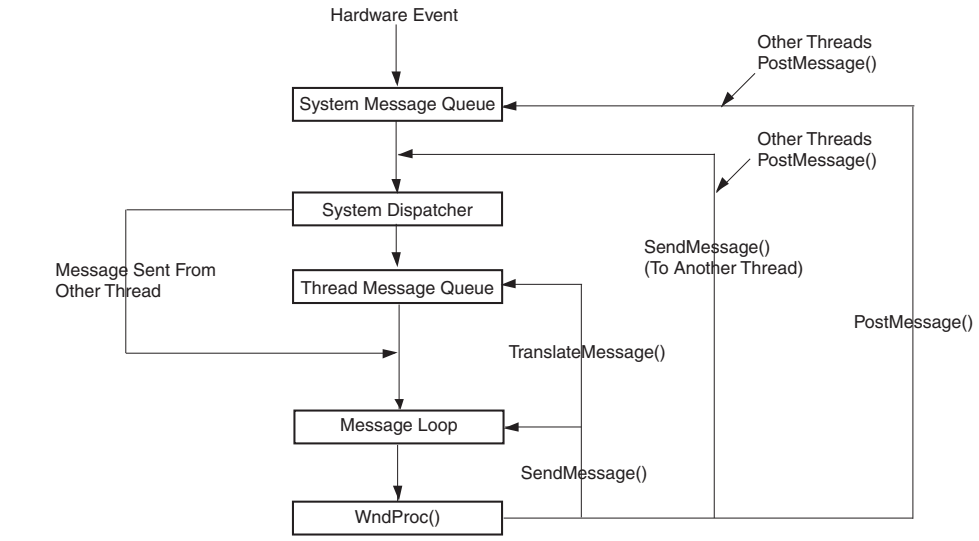
resort to the most common feature of any document application and invent new methods in order to capture the text or hyperlink around the point of the mouse click. Simulated selection is one method we use.

4.1.1 Intercepting Messages: System Hook

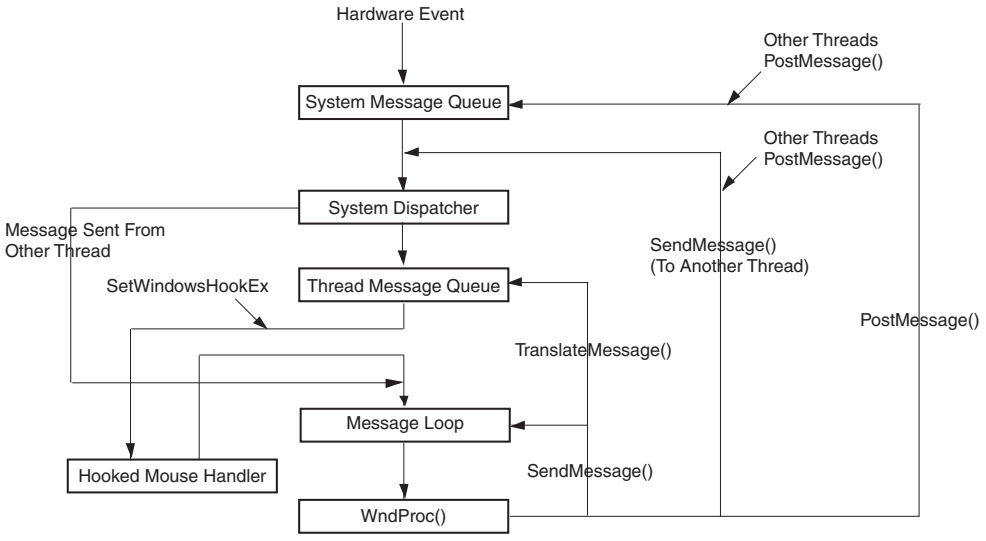
We first use the Windows system for illustration since most clients are PCs, and Windows is the most used operating system for PCs. An immediate idea for the method of intercepting events would be to list out all currently running processes and applications and to find out their respective handles. When a mouse click event occurs, we will identify the window application on focus and intercept the message sent to it.

4.1 is the diagram for a message life cycle. We show how the text capture can be done. When a user clicks on the mouse at a location or presses a key, it generates a message via `SendMessage()` or `PostMessage()` to the operating system. The message arrives at the system message queue, and the system will determine where it will be sent for further interpretation. The message is then delegated to a specific thread or process queue where it will be waiting for `WnProc` to retrieve it, as shown in (a) of Figure 4.1.

The enriched content program will set up a hook function `SetWindowsHookEx(int nFilterType, HOOKPROC hkprc, HINSTANCE hMod, DWARD dwThreadID)` to the system so it can inspect each message passed to the system. Specifically we can set the hook so it will only inspect the generated mouse messages by selecting option `WH_MOUSE` at the `nFilterType`. `hkprc` will be replaced by the name of the procedure (e.g, `HookedMouseHandler`) which has operations we want the system to carry out



(a)



(b)

Figure 4.1: Message Route Diagram

before the message is passed to the `WndProc`. When a mouse hook is installed, all the messages produced by the mouse will first be sent to the `HookedMouseHandler`. See (*b*) of Figure 4.1.

We will describe what `HookedMouseHandler` does with the message in the following sections. To have a clean exit from the enriched content program, we have to call `UnhookWindowsHookEx` when terminating the enriched content program.

4.1.2 Re-routing Window Procedure Path

Each windows application comes with a window procedure loop responding to various event messages. The window procedure is part of a pointer data structure; we can save the current address of this pointer and redirect the window procedure to our program address. In this way, any event message would be sent to our program first for preprocessing and then be passed to the previously saved procedure address to finish its journey.

`SetWindowsHookEx()` can establish the aforementioned requirement. In our installation, `HookedMouseHandler` carries out the operations to capture the text underneath the mouse. The technique we are deploying is "mouse action simulation." The inspiration is from our observation of users' behavior. How does a user usually tell the program what part of the text she want? She selects it. First she moves the mouse to the starting point of the desired text, clicks the mouse button, drags the mouse to the ending point of the desired text, and releases the mouse button. The text chosen will be highlighted. `HookedMouseHandler` will simulate exactly what the user does and deceive the application to believe that the user has done it.

4.1.3 Modify Intended Event Behavior: Filtering and Handling

When an event is sent from the Windows system to an application, it will call the application's window procedure and then pass parameters and data to that procedure. As we have replaced the original procedure with ours, our procedure will be carried out for processing the parameters and data. Upon receiving a mouse click event, which contains its location information, our program can determine if the mouse is over a text. If so, it can obtain the segment of text around the mouse location.

Mouse Action Simulation

We will use capturing a text line as an example. The generalization for capturing a paragraph and whole document can be followed similarly. Note that we only manage the "document application" whose main function is to view and edit documents. Applications of other kinds, such as for illustration and the like, are not considered "document application" here.

Upon a mouse action, `HookedMouseHandler` will examine if the message is a `WM_LBUTTONDOWN` and if a meta key (either a `ALT` or `CTRL` key) has been pressed at the same time. If it is the case, then the handler will obtain the beginning and ending boundary position of the line text described later in this section. It sends a series of messages simulating that the user moves the mouse to the beginning of the text boundary, left button clicks the mouse, drags the mouse to the end of the text boundary and releases the mouse button. The simulation is carried out by `PostMessage` calls. `PostMessage` will put messages into a queue.

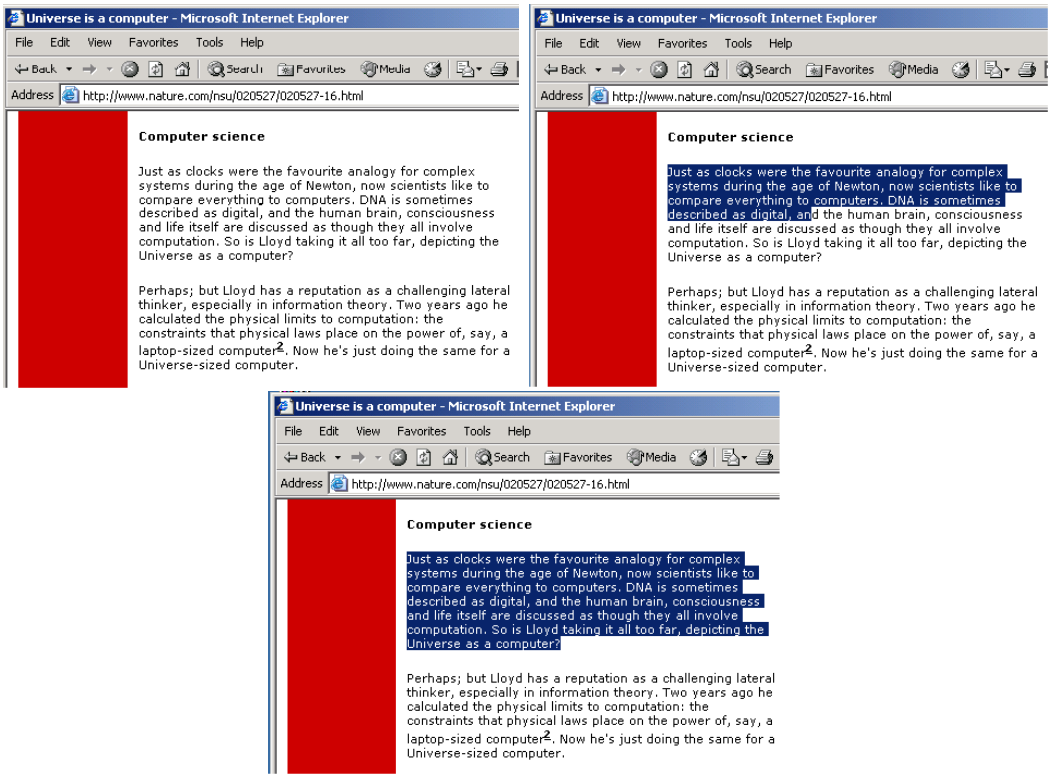


Figure 4.2: Text Selection Process

Figure 4.3: Copy Shorthand

Keystroke Simulation: when CTRL is the meta key

As we are dealing with only "document application," we know every document application comes with standard text handling functions, such as select-all, cut, copy, paste, delete, and so on. These functions all come with their shorthands. In Windows, the shorthands can be managed by the "keyboard accelerator." For example, CTRL-C is the shorthand for copy function, and CTRL-V is the shorthand for paste function. The shorthands have become part of the GUI program culture and are prevailing on almost all GUI applications.

We can capture the selected text by sending the focused application a character message 'C'. Since the CTRL has been pressed already, this combined keystrokes become CTRL-C and will be translated into a COPY action. The captured text will be in the clipboard then.

Detecting Text Boundary

To enable enriched content application to detect the boundary of text, we need to find out what consists of a boundary. It is clear that document applications will not

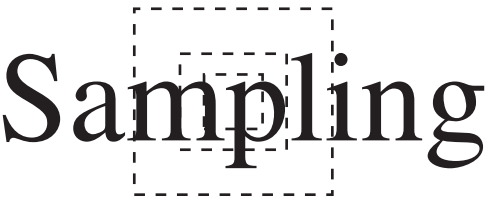


Figure 4.4: Background/Foreground Color Sample

have a mosaic-like background, and most likely the background color is monotonic and so is the font color. We can sample pixels around the neighborhood of the mouse click, count the pixels, sort the counts, and take the color with the highest count as the background color, and the color with the second highest count as the font color. We need to gradually enlarge the sampling area, and if the choices of colors persists, then we have high confidence that we have chosen the right background and font colors.

Once we acquire the font and background color, we can tentatively find out the approximate line height by sweeping a horizontal line segment vertically. By examining the color variation sampled from the sweeping segment, we can tell the location of a text boundary. When the sweeping segment goes across the margin of the text, the segment pixels will only contain a background color without any font color pigment. From this, we know it has just crossed the boundary. Using this method, we can detect the height of the line text, which may be higher than the local font height sampling since capitals may occur in a line. We then can sweep left and right with a vertical segment of text height to detect left and right boundaries using the same argument. Now we have a rectangular box area containing the line text. Enriched content needs only to know the upper left and bottom right corner coordinates of the box containing the text. These two coordinates are passed to

Figure 4.5: Sentence Boundary Detection

the `HookedMouseHandler`. `HookedMouseHandler` will simulate the mouse move to the left boundary and then drag it to the right boundary to complete a selection simulation.

The detection of the paragraph boundary will be similar. After finding the text height, we can estimate the distance between text lines. When the sweeping line exceeds the between-lines distance, we can expect there is a new paragraph. Once a paragraph's first line and last line are found, we can locate its upper left and bottom right boundary. We need only the paragraph's approximate coordinates of the upper left and bottom right. There are minor points that we have to take care of. For example, the paragraph's shape may not be rectangular and it may be intersected with a picture. However, as we will force the "copy" function to output text only, we need not worry about including images that will not be included.

One might ask, why not simply deploy a OCR (optical character recognition) program. It is true that OCR will serve the purpose of detecting characters directly but it has the shortcoming of not being able to obtain the full text whose most parts are not visible to the window client area. OCR's approach is at best a screen scraper, and it is pretty a straight forward solution at first sight but an inadequate one. A very good OCR can, however, detect any character on the screen, but we are interested in document applications, in other words, information oriented

applications. A screen scraper is not what we want. Further, OCR are prone to errors when the text fonts vary in a document.

To obtain the whole document, we simply simulate the keystroke `CTRL-A` shorthand which stands for "select all" and this will select the whole document whether or not they are all in the window client viewing area. This is not achievable with an OCR application. OCR can be used as a helper application for enriched content when it encounters documents that are represented as images, i.e documents are scanned as images.

4.2 Cross System: Link Type

4.2.1 Intercepting Messages: Client-Side Proxy

We first classify different types of proxy and give reasons of why we choose client-side proxy as an ideal implementation approach.

Categories of Proxies

In the Web, proxy server is a module that stands in between the webpage server and the requesting client (browser). Proxy servers have multiple functions: they filter the incoming and outgoing messages and content to ensure security and privacy; and they cache content locally for a faster delivery in future when a repeated request is issued. In general, there are three types of proxy:

[Wide-Are-Network (WAN) Proxy] WAN proxy is a proxy server on the global network; a client needs to issue requests through public networks in order to reach the WAN proxy. A WAN proxy are physically located in a different

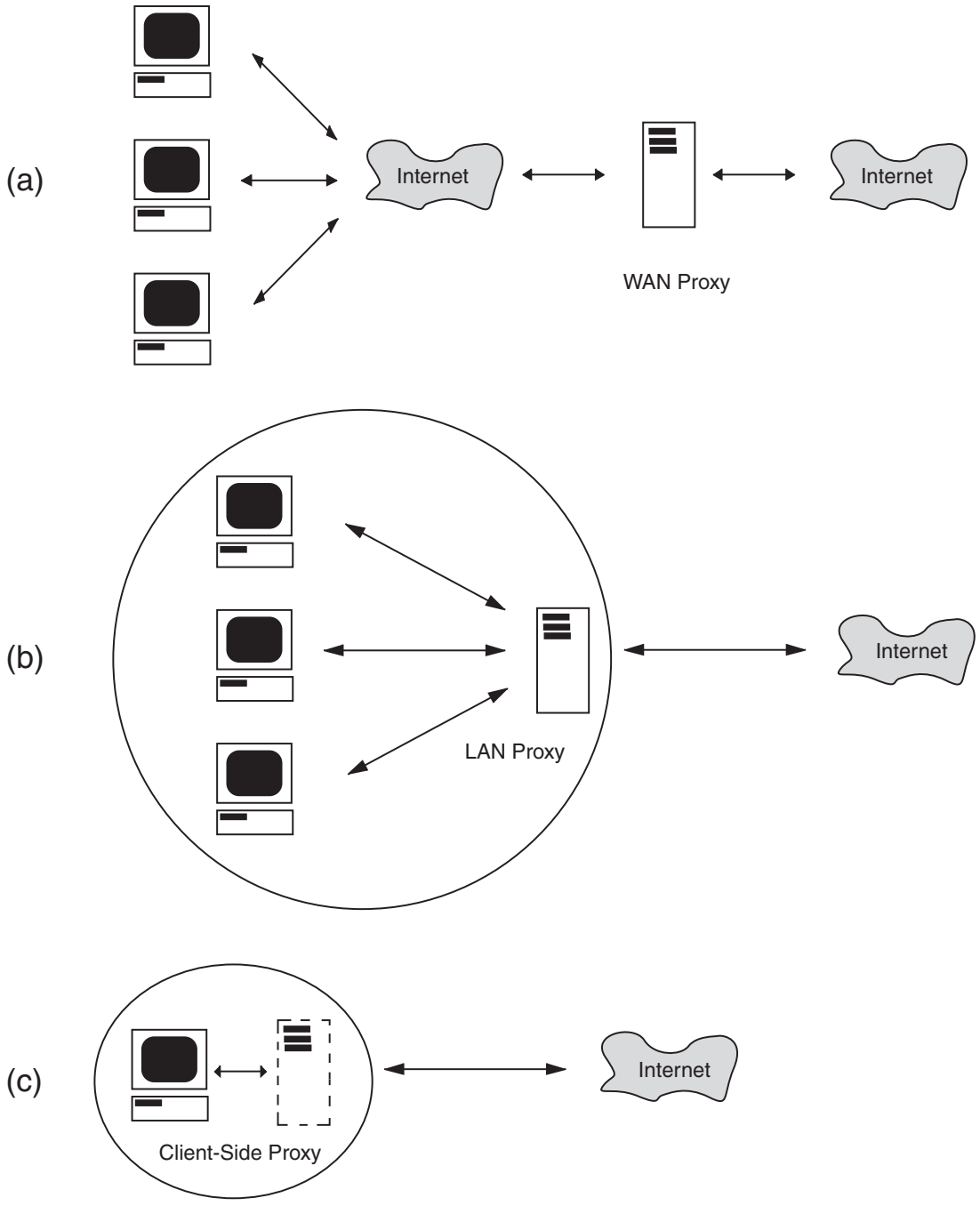


Figure 4.6: Categories of Proxy

place from the clients.

[**Local-Area-Network (LAN) Proxy**] LAN proxy is also a server on the network. The main difference between a WAN proxy and a LAN proxy in that LAN is on a local network and usually the clients are not physically too far away from the proxy server; they can be in the same building. LAN proxy are used mostly in corporations as a way to set up their firewall [1] in order to protect their corporate servers. WAN and LAN proxies have one thing in common; they serve more than one client.

[**Client-Side Proxy**] Client-side proxy, on the other hand, serves locally and has only one client: the machine where client-side proxy resides. The delay in communication between the client-side proxy and the client is minimal; it is a local messanging which is no different from inter-process communication. In fact, UNIX operating system uses the same protocol for process communication. Unlike WAN and LAN proxies, client-side proxy's request-response limitation on network is inherited from the original network capacity. WAN and LAN, however, have to include their responses capabilities. When facing with massive requests, WAN and LAN proxies may be overloaded. The main purpose of the proxy used in our demonstration is to change the behavior; therefore, the caching function is not implemented.

Why Client-Side Proxy?

We picked client-side proxy as the implementation choice for several reasons: the first reason is that it avoids network overload. The network traffic will be localized;

[1] Firewall is a mechanism to filter and reject unwanted messages.

the message packets issued for communication are restricted within a single local machine unlike a remote proxy, which may be burdened with massive requests if it serves too many clients. The second reason is for programming. Modifying a browser so it will use a proxy server involves only changing its configuration. Changing configuration requires overwriting entry in configuration files, which is a much more simpler task than programming plug-ins for different browsers. Besides, we need to write only once for one configuration overwriting module in one cross platform language, such as Ruby or Java, and use it in other operating systems as long as the same browser is available. In plug-in programming, however, one first has to write a more complicated code calling browser's APIs and to re-write the same code just to make it native to a different operating system. The third reason is financial: it costs to buy a proxy server.

4.2.2 Re-routing Content Path

Once the client-side proxy is established, all the messaging and content between the client and the server will first go through the proxy. Enriched content can filter out intended messages, much as we have done in the section 4.1, and handle the messages according to our designed behavior. We can also change the content sent from the server before it reaches the client.

4.2.3 Modify Content and Link Behavior: Replacement and Codes Injection

Enriched content can modify the content (of messages) sent between the client and Web server. Modification can be done by either replacement or injection.

Enriched content can replace the unwanted segments with blanks, or replace it with desired content. It can also inject new segments of content or codes into the original content. Obliged reverse link application in Chapter 6 will show a list of reverse link URLs in a mini-window. This mini-window will be generated from the injected code into the original content. Enriched content can inject a segment of JavaScript code. When the browser receives and executes this code, the browser will spawn a new mini-window, if it has not already been spawned, and within that window another JavaScript will make a request to the client-proxy for reverse link information. Client-proxy serves not only as a proxy but also as a content enhancer. In future, client-proxy will be the places where contextual agents can interact with each other (Section 9.2.1).

4.3 Client Server: Network Communication - Common Gateway Interface(CGI) Service Request

After the client side obtains the words and phrases and their residing sentence, the client issues a network request to the server for data service. For network communications, we use existing standard HTTP and if necessary HTTPS which is a secure channel protocol. HTTP is based on TCP/IP which is the standard for Internet network protocol. Could we just use TCP/IP and not HTTP? Yes, but we do not think there is a need to invent another transaction protocol just for this purpose. No need to reinvent the wheel here.

Common Gateway Interface (CGI) is a well defined interface for requesting service. The network communication design will be based on CGI, using the **POST**

or the **GET** method for accessing menu and content. In future, the request for data service may be in the form of Web Service (SOAP, WSDL), which is an emerging W3C standard.

4.4 Add-on Module

4.4.1 Server Operations: Server Module and Ruby Scripting Language

The server module will identify the requirement, process it, and send back the result in menu format. Each application will have its own server operations and they are illustrated in individual chapters. We use Ruby [Ruby] scripting language as the programming language. Ruby is an object-oriented scripting language that come with class libraries in network, GUI, and server [mod_ruby]. It has cleaner syntax when compared with other scripting languages, such as Perl or Python.

4.4.2 Client Operations: Cross Applications - Dynamic Link Library (DLL)

Client module includes a message detection and message re-routing functions that can diverge the message destination temporarily so the client module can obtain textual information. This has been described previously in Section 4.1. Text capturing module on the client side is written in dynamic link library (DLL). DLL can be loaded dynamically into any running process; it can use the same memory spaces (and name spaces) of the process without causing memory violation. System hooks are implemented as DLL.

4.4.3 Client Menu Presentation: Reuse Browser Component

When the program receives menu content, it parses out menu items and visually displays them to users. The enriched content menu will show up right at the point of the mouse click. Menu presentation source format is based on HTML. We can fully utilize the render engine provided by the browser component without duplicating the effort. Each application will have distinct menu items, and they will be explained in later chapters.

4.5 Other Considerations and Alternatives

4.5.1 Cache for Efficiency

One thing all web users have in common is that they do not want to wait. Of course, a local cache storing previously requested results would help to present a menu more promptly. On the server side, we could also cache on the search query results indexed by the contextual cues and sort them according to their query rate. The overall retrieval time for repeated contextual queries will be saved. This is a technical possibility; we will not implement it.

4.5.2 Fatter Client

Many of the tasks performed by the server can be executed on the client side. The degree of independence from server operations relies on the add-on module designers. Contextual analysis and process to search for reference can be done locally on the client side without making requests to a server. Integrating the

server functions with the client side will increase the add-on module program size, and it will become a fat client program rather than a thin client program. We can have a independent fat client unit; however, if we want to access and share data which we have saved through the network, then using a server module is necessary. Again, fatter client design is mentioned as a technical possibility, and it is not implemented.

Given the generalities of the architecture and process, a wide range of applications can be spawned.

Part II

Applications and Conclusion

Chapter 5

Extended Semantics

Extended semantics, obliged reverse link, composition assistant, and version control are four applications of enriched content. In each application we need a different content menu and sometimes we have to identify the properties of a text to see if it is part of a hyperlink. The preparation of each menu will be described in individual sections.

5.1 Existing Standards: Xlink

5.1.1 Multiple Destinations Hyperlink

Currently each hyperlink has only one destination, if many destinations are relevant, the author must create a separate link for each reference. There are many situations where multiple destination hyperlinks are useful. For instance, we need them for the purpose of comparison and contrast. For example, for news we could have CNN.com, ABCNews.com, and NYTimes.com. For entertainment, we can have Mickey mouse, Pokemen, or Simpsons. We can use multi-destinations hy-

perlinks when we need evidence for an argument, specific details, or quotations if there is more than one origin.

The need for the multi-destinations link is abundant. The purpose of hyperlinks is about the association of information. This association is more like a map than a single threaded line. Hyperdocument such as webpage has many hyperlinks. When a webpage itself is considered as a single entity, its links would be the associations. The necessity of the multi-destinations link is most evident as we are already using one of its alternative: the option <**SELECT**> tag in HTML.

5.1.2 Xlink

The Xlink standard from W3C will provide more than one option to a hyperlink and it is up to the browser designer to determine how the feature will be implemented. In future, we expect there will be more links having multiple destinations.

However, if an author tries to 'hyperlink' a lot of phrases in a webpage, the size of the webpage will grow significantly. Xlink does provide a method to point to the menu resources that contain menu items. Even if the links only point to menu resources, the page size will still increase dramatically if there are many reference links.

5.2 Cross System: Client Side All Document Applications

Extended semantics focuses on text contextuality. Capturing text is explained in the Section 4.1. There are cases when we may run into problem. When users are

using the PDF viewer or other document viewing program whose text selection is not set as default.

5.3 Client Server

5.3.1 Network Communication: Common Gateway Interface (CGI) Service Request

In enriched content we are dealing with two types of text: plain text and hypertext, specifically, hyperlink.

For plain text, we want to capture text so we can analyze them. The text content will be sent through the network via HTTP protocol. Enriched content program can issue a GET request

`http://www.server.org/cgi-bin/enriched_content.cgi?text=INTERESTED_TEXT`

and the server will respond. `server.org` is the domain name of the server. `extended_semantics.cgi` is the CGI interface for the extended semantics. `INTERESTED_TEXT` is the text users are interested in. For example, it can be "The Nobel Price laureate for Physics in year 2003".

For hyperlink, enriched content program can issue a GET request as well

`http://www.server.org/cgi-bin/enriched_content.cgi?link=INTERESTED_URL`

where `interested_url` can be any URL, for example, `http://www.pbs.org/programs`. The prefix "`http://`" can be dropped off; it relies on the communication design.

5.3.2 Server Response

In essence, server response will be in HTML format; it can be tailored in XML for more specific usage. In our demonstration, we are mainly concerned with the display of the information. We do not deal with the problem of enabling the client to use the server response. Obviously, with XML client can conduct more sophisticated manipulation on data formatted in XML.

5.4 Add-on Module

5.4.1 Server Side: Harvesting and Pruning

There are two main processes on the server side: harvesting process and pruning. Harvesting manages the construction of plausible query terms and query submission to the Web. Pruning process manages collecting and summarizing the returned results from the Web. We use the returned result from Google as an example and show how we can apply simple lessons that we learned from Chapter 2 on Contextual Computing.

Harvesting: Query Terms Construction

Servers can run a simple contextual analysis on the sentence received from the client side, extract, and construct a series of queries. One example of simple contextual analysis is to throw away stop words such as “it”, “they”, “is”, “a”, and “the” which play a lesser role in providing essential information. A group of combinatorial queries could be composed easily.

Example 1 "... U.S. Marines and Afghan opposition groups were preparing Sunday for a possible final assault on the remaining Taliban stronghold of Kandahar, with the number of U.S attack and support helicopters on the ground nearly doubling overnight. ..."

Quote from CNN.com Dec 2, 2001 Posted: 3:24 PM EST

We have a set of words : "U.S.", "Marines", "Afghan", "opposition", "assault", "remaining", "Taliban", "stronghold", "Kandahar", and "helicopters".

Search combinations can be:

- "U.S Marines" + "Afghan"
- "Afghan opposition" + "Taliban stronghold Kandahar"
- "U.S" + "helicopters"
- ...

A simple rule of thumbs is to combine any word starting with a capital with other words and preferably sequentially in the order they appear. The first three searches all generate related results from Google.

In future a more thorough NLP will make the requests more precise, and we can obtain more accurate information. Further more, we can identify the url of the webpage currently visited and the title of the page from the browser, then we can obtain webpage category from portal (Yahoo) and this will add to the contextual understanding. It is conceivable that the more queries is imposed on the search engine, the fewer results will be returned. Results may be fewer but they are more precise. We include the word(s) in the close vicinity of mouse click and use 2 $\sim$

3 terms combination as query terms. What follows are possible aspects for query submissions. We show what to do with three different types of databases.

[**Generic Search Engine (Google) and Encyclopedia (Brittanica)**] Sending the query string will work as we have shown in the previous examples. Longer query terms may be applied for generic search engine; however, it is too restrictive if we want to match multiples terms in encyclopedia site.

[**Dictionary (Merriam Webster)**] Combinatorial patterns might not help much here as the purpose is to understand a single word or phrase (adjacent words). We may incorporate synonyms, antonyms, hypernyms, and meronyms using database such as WordNet.

[**5W1H (What, Why, Who, When, Where, and How)**] One of the ideal search is to find information and knowledge in all five aspects of questioning. We can have a naive formation for a query: add "How" with a verb shown in the context and add "What", "Where", "Who" and "Why" with a noun or with a complete sentence. "Who" and "Where" are likely to be combined with words starting with a capital letter which is possibly the beginning of a personal or geographical name. "Who" could be submitted to The Hidden Web [Kautz 97].

Pruning: Merging and Collapsing Similar Results

We will search Google using its four categories: Web, Group, Directory, and Images. And we prune the results by the following methods:

[**Directory**] Directory classifies different webpages into hierarchical categories. We can compare two subdirectories names to determine if they are of related

topics. For instance, if we use the hierarchy as a guide and merge the pages when their top two hierarchy is identical then we have the following result when searching the word "Apple" on Google. [1] The number in front of the listing is the original order from Google. The format of the listing is " number : page title: category: URL."

Categories:

Computers > Companies > Product Support > Apple

Computers > Systems > Apple

The first two categories are shown by the Google directly. The merging method will collapse the following two pages:

6:www.apple-history.com:Computers > Systems > Apple:

www.apple-history.com/

1:Apple:Computers > Systems > Apple > Macintosh:www.apple.com

There are other pages not shown here. These pages are all under Computers > Systems category.

3:The Apple Store (US):Computers > Hardware > Retailers > Mac:

store.apple.com

9:The Apple Collection, Desktop Pictures, Mac Links, Bookstore and

... :Computers > Graphics > Clip Art > Apple Logos:

[1]Search conducted in Aug, 2002. Only English pages are listed here.

www.theapplecollection.com

However, merging the top two subcategories will leave the above two pages standing out independently. We consider mering the top most category as an unreliable approach; otherwise, these two will also be collapsed into the previous category. The benefits of merging is obvious:

7:Newton's Apple:Category:Kids and Teens > Entertainment > Television > Educational > Science:www.ktca.org/newtons/

15:Fiona Apple:Category:Arts > Music > Bands and Artists > A > Apple, Fiona:www.fiona-apple.com

16:Apple Vacations - America's Favorite Vacation Company:Recreation > Travel > Travel Agents:www.applevacations.com/

17:Apple Seeds Homepage for Inspirational Quotations, Motivation: Science > Social Sciences > Psychology > Self-Help > Motivation

18:Washington State Apple Commission:Business > Industries > Food and Related Products > Fruit and Vegetables > Associations: www.bestapples.com/

The merging method can quickly list out several well-known concepts about "Apple" which include the apple that hits Newton, the apple that we eat, and the musician Fiona Apple. And some lesser related concepts: a travelling agency, and self-help quotation page. Merging the same top domain name is also plausible but it may be unreliable as there are many pages in AOL.com which are not really about the company AOL. The same reasoning can be applied to webpages under the same internet service provider. Merged pages

can be further grouped together by their titles. We also find out that the directory listing from Google is mostly extracted from the Google's search result with the same listing order as long as the page has been classified in Google's directory. The pages that are not classified will not be listed in the directory search.

[**Discussion Group**] For discussion group posting, we focus mainly on the group category. For "Apple" on Google's discussion group, we can find out the following for the top 20 posting sorted by Google's relevance metrics:

```
1:comp.sys.apple2:Apple II Csa2 FAQs:  Users' Groups, Part 22/25
5:comp.sys.apple2.marketplace:FS Lots of Apple Manuals
13:comp.sys.apple2:Apple ][ font
17:comp.sys.apple2:Mac stuff for Apple II stuff - Swap comments?
20:comp.sys.apple2.marketplace:FS/T- Apple ][ items and more...
```

Other groups under `comp.sys` are `comp.sys.apple, comp.sys.mac.forsale, comp.sys.mac, comp.sys.mac.advocacy` and two more under `comp` is `comp.lang.java.gui, comp.publish.prepress`. We also have the apple that is related to fruits (or food).

```
7:rec.food.recipes:Simpson and Vail Spiced Apple Tea and
Metropolitan Apple:rec.food.drink.tea
12:rec.food.recipes:Apple Cakes (5) Collection
```

Issues related to Microsoft and Apple are discussed in the social section,

and one posting related to the relation of the method of growing an apple to the environment.

```
8:soc.org.nonprofit:Microsoft deal hurts Apple in education market:
soc.org.nonprofit
19:soc.org.nonprofit:Apple cringes at Microsoft school deal
14:sci.environment:Organic Apple Growing Is Best Thursday
```

We can also use the overlapping on subject titles to guess the relation between two postings. If two subject titles have more than two none-stop-words repetition, then chances are that they are about the same topic and can be merged.

[**Web**] As we have discovered that the directory listing is the extraction from the Web search, we will list only the result that are not classified by the Google's directory search. We will merge the webpages whose suffix domain name are identical. For instance, `store.apple.com` and `www.info.apple.com` will be merged.

[**Images**] There is less linguistic clue we can use in images search so we can simply list out the first return page of Google's image search.

5.4.2 Client Side: Menu Presentation - Generic, Expert, and 5W1H

We briefly describe the process of obtaining menu content for each different resources.

Menu Presentation: Generic Search Engines (Google) and Encyclopedia

The menu presentation layout will follow the Directory, which classifies possible senses of a word (phrase), Discussion Group, which will uncover mostly discussed issues, Web, which will list important pages related to the search query, and Images. Encyclopedia search will list out the domain knowledge as a category.

Menu Presentation: Dictionary

Presentation for dictionary database is just the display of the result. We may parse and remove the possible advertisement and other decorative parts of a webpage and re-arrange the visual effect.

Menu Presentation: 5W1H

This menu presentation is for a strong and competent Natural Language Query search engines. The "Why" menu listing will be the answer passed back by the request for "Why" + Query String. "What", "Who", "Where", "When", and "How" are proceeded analogously.

We are particularly fond of the 5W1H method as it covers general questioning methods. For instance, querying "Computer Graphics", we could supply "Who" with the organization ACM SIGGRAPH or computer scientists who work in this

field, “How” and “What” with a class homepage, “When” with the date of conference, “Why” with a list of graphics applications such as CAD and animation, and “Where” with a list of graduate schools that have “Computer Graphics” program. These are all possible if a large ontologies resource is available. It could be the index list of the Encyclopedia of Britannica. From one index and a well designed (but not perfect) natural language processing, we can provide the aforementioned result. However, there are possible ambiguities among 5W1H. One of the problems the program has to solve is how to distinguish between “Who” and “What”. For example, the program has to distinguish the URL of an organization and the URL of a class homepage. A search verification from Yahoo directory on “organization” is one solution.

5.5 Benefits: Semantic Intention: User Chosen Data Source, Private Link

5.5.1 Users Choose Data Source

Users can choose the query database source according to their own personal interests and their mind. In fact, one can turn a “serious” website, CNN for instance, into a fun one by choosing the semantic data source from Disney (if they had one). All the reference on mouse could go to “Mickey”.

There is ongoing work and initiative for the Semantic Web. In future, when a vast amount of ontologies sites are available and nomenclature and semantic interpretation are defined and standardized, users can choose among these services to obtain more accurate results. Of course, at that time extended semantics will be

adapted to the standard. The key is that user can choose her data service provider according to what she considers appropriate.

5.5.2 Users' Private Links

Each user has his own personal way of organizing the world. Just as an extend semantics server can suggest meaning extensions, we can supply our own semantics. Instead of using the data from the server, we can link words and phrases to the association that we design. For example, the word "sports", can be linked to a sport center website. When we click on the "sports", extended semantics will bring us directly to the preassigned website. Or we can have our own list of menu on "sports", then extended semantics will present our own menu on "sports" and we click on the "'sports".

On implementation we can have several considerations: extended semantics needs to first check if there is a local definition on the client side, does the definition have a single or multiple items? and whether we have pre-set to inquire extended semantics even when there is local definition. Naturally, extended semantics needs to keep a client side library storing individual data definitions. These personalized semantics can also be stored on the server and can be made accessible from login-password session through the network.

Chapter 6

Obliged Reverse Hyperlink

Hyperlinks can only be followed forward. In current browsers, there is no way to go from a page P to a pages that link to P.

By applying enriched content, we can partially resolve this limitation. We can keep track of all the links that have been followed (referral links) and maintain a statistics of the traffic. We then have a real time visitor data that can be used for real time reversed links. However, we will keep only the most recently active referral links which have the highest number of references. This is reasonable; just imagine Yahoo, how many reverse links Yahoo homepage has? Certainly we want to economize the data storage and its current state.

We use the word "obliged" because before the software module that enables reverse link become widely installed, "real time" referral links will not be available for most websites. However, as Google's [1] facilities provides archived reverse links functionality; we can use its service as a temporary substitute and enforce the reverse link function.

[1] Alta Vista also provides reversed links.

6.1 Cross System: Text Type and Link Type

Recognizing hypertext across document application is accomplished through identifying "http://" prefix in texts. Enriched content module will carry this process indiscriminately for all document applications.

Enriched content will set the browser configuration so the browser will communicate through the client-proxy as described in section 4.2. The URL requested by the browser will be intercepted and be incorporated into the injected codes. The original content with the injected code will then be sent to the browser.

6.2 Client Server: Network Communication - CGI

The client sends url to the server to requests a real time menu. If the server has not installed the reverse link module, it will respond to client with "service not found" and client can issue another request to the reversed link server (client side proxy). When reversed link server receives the reversed link request, it will redirect the request to Google and then preprocess the returned result before sending back to the requesting client.

The communication between client, reversed link server, and reversed link archive server are all conducted through CGI. Suppose the user click on the URL `http://www.cs.nyu.edu/index.html`, the client will first issue `http://www.cs.nyu.edu/cgi-bin/reversed_link.cgi?url=/index.html` and request `www.cs.nyu.edu` server to send over the reversed link menu. If `www.cs.nyu.edu` does not have reversed link service installed, then the client will receive the error message. When the client receive this error message, it will issue

`http://www.server.org/cgi-bin/reversed_link.cgi?q=www.cs.nyu.edu` to the pre-assigned reversed link server `www.server.org`. `www.server.org` will look up the reversed link from the Google and pass the information to the client. Of course, client can look it up from the Google directly but in this case it will also need to add summarizing and formatting menu item on the client side.

6.3 Add-on Module

6.3.1 Server Side: Server Log Processing

Server operations involve pre-processing server log data as well as periodically updating its reverse link index. Server module pairs referring pages with referred pages; it sorts referring pages for each referred page. This information will be used in reverse link menu presentation. We want to concentrate on "real time" reverse link. Because there may be a couple hundred thousands of references. But we only care about the most recent ones only since reference links on referring pages that haven't been followed for long period of time (months) are unreliable and this implies the importance of the links are weak. However, module designers could optionally choose to keep the last access time from an unique domain name or an unique referrer page even if these links haven't been followed for a long while.

In order to obtain referrer's page, we need to enable the log entry. In Apache server file, `httpd.conf` we need to remove the # sign in front of the following two lines. The first is server log format line:

```
LogFormat "%h %l %u %t \"%r\" %>s %b \"%Referer\" \"%User-Agent\"" combined
```

%h represents hostname, %u represents user name (if available), %t is for time stamp, %r is the request method and file, %>s is the protocol number, %b is the number of bytes transmitted. What we want is "%Referer" entry. And the second is log location specification line.

```
CustomLog /home/hubert/WWW/logs/access_log common
```

A typical log entry will look like the following:

```
127.0.0.1 - frank [10/Oct/2000:13:55:36 -0700] "GET /apache_pb.gif HTTP/1.0"
200 2326 "http://www.example.com/start.html" "Mozilla/4.08 [en] (Win98; I
;Nav)"
```

The log entries can be stored in a separate database. A simple separator program. will be able to parse each log entry into log array which can be used for updating log database. We can write a module such that for every request issued to the server, the server will deliver the page, update the log file and database. This will keep the referrer information real time.

6.3.2 Client Side Proxy: Code Injection and Reverse Link Request

Client side proxy will inject a small segment of code so when the browser processes the code, it will spawn a mini-browser containing codes that will issue reverse link information. This request can go through client side proxy which will issue

request for the browser. If the Web server serving the page have reverse link information, then it will be passed to the client proxy which will then pass the same information to the browser. If the Web server serving the page does not have reverse link information, then the client proxy request will be denied. Upon receiving the denial protocol, client proxy will issue reverse link request to Google which has archived reversed link information. Client proxy will take this archived information, reformat to the pre-designed layout and then pass it to the browser.

6.3.3 Client Side: Menu Presentation - Full or Truncated LIFO Order

Reverse links can be displayed in several fashions. One way is in reverse time order. Another is to compress a day's (or a week's) references into one block and show each referring pages only once, even if it contains more references. Each number of references for each referring page can be appended as an indicator of its referential strength. The details of a reference can be expanded when users do a mouse-over on it. Of course, when the compressing time frame is customizable by users, the resulting information will be more insightful.

Still a third method is to exhibit the referring pages alphabetically together with its referencing times. To avoid displaying excessive references we can limit the length of the listing; however, we will give options to extend them on demand.

6.4 Benefits: Semantic Junction, Aid to Web Design

6.4.1 Semantic Junction (Internal and External)

A reverse link can show the link's semantic junction on the Web: where the Web link path is coming from and where it is going to. From the reversed link information, we can estimate its relationship within its hosted website. Moreover, it also reveals its popularity, usefulness, and possibly the duration of each visit.

A precise reverse link can be employed by Xpath when a document are made from XML. Xpath, or more precisely Xpointer, points to the exact location of reference, not just the web page address. Reversely, server log can show the precise source of reference origin under the Document Object Model. Indicating the exact reference locations signifies the real semantic junction

6.4.2 Aid to Website Design

Using reverse links and the analysis of their semantic junctions, we can see the points of interest from the Web usage perspectives. Most site analysis programs will show only the website authors' view and ignore the true web users' habits.

Because the reference shows true strength of the relationship from actual users' points of view (links that are followed), which might differ from the original design intent, this will help the website designers and architects to determine what elements should be moved, reinforced, edited, linked, or removed. Web site design has been heading in the direction of being more adaptive and user-customizable; semantic junction analysis can be an effective tool for a successful Web design.

Chapter 7

Version Control

Many websites update their pages periodically [1] and they often relocate or remove their webpages without notifying their faithful readers appropriate. Not surprisingly, this happens even to the websites of reputed media corporations. It is a nuisance to run into the case when a previously bookmarked page has become a broken link, and we have to look around, above, or below (in terms of hierarchy) the website in order to locate needed information. Try to bookmark a section in an online news website and you will understand what this means.

A solid website should be responsible for all its publications and maintain an archive. There are some projects for archiving the whole web [Kahle 97]; those projects are too ambitious and bound to be disappointing due to the period of time to update, and the limit of storage and processing time. Versioning and archiving is best performed by each individual website itself. For websites that do not have a version control module installed; however, the archive project could be a compensation.

[1] Version Control is only described and not implemented

Some websites solve the versioning issue by providing a link to its own "archive" page which contains all previous versions or by providing a "search" button which allows users to search all its past records. A search, however, may lead to other pages containing the same term. We can improve this in a more logical manner. We should be able to click a link and ask for a specific version; we should not have to go through the search process in order to select what is the exact page we are looking for.

With enriched content architecture and designed modules installed, a plethora of documents can be re-viewed and re-used. Readers can select from a list of versions and choose the one they would like to access. They can also compare versions of documents. Literature usage is maximized; readers are benefited.

7.1 Existing Standards: Concurrent Version System (CVS), Software Engineering in Practice

The software community has long been troubled by development cycles, project coordination, and version management. The chaos and bugs produced by lacking the proper software environment are daunting, and the more delays in fixing a problem, the more complex the problem will become. Retracting previous changes in order to identify a bug origin or to adapt to newly changed software requirements necessitates the need for an organized schema.

Software engineering practitioners have implemented various software version management tools such as RCS (Revision Control System), SCCS (Source Code Control System), and CVS (Concurrent Version System). These programs are

developed in order to coordinate team programming projects. One of their main functions is version control that allows utilities to show the difference between two versions of a software package and to retrieve the previously committed version if requested.

CVS, especially, is more popular in the software community. It does not dictate the programming language you are using, so its function is sufficiently general. We can use it for the purpose of general file control. Or we could perceive this differently. We can view natural language as a programming language. While articles, reports, and books are written by natural language, we can think of them as "programs" "coded" in natural language. In this respect, CVS can be used for version control on "natural language program".

7.2 Cross System: Replacing File Related System Commands

Version control implementation can be done as described in Chapter 5 on obliged reverse link where the client side proxy could issue request for versions on the server, if the server does not have the versioning service, then the proxy can request from the archives we have mentioned prior. As the mechanism is the same, we describe here what need to be done on the server side.

Version control requires document management software which is usually not installed in operating systems. Document management software needs to record when a file is created, modified, removed, renamed, or relocated. In this section, we use the UNIX system as an example, to show how we can incorporate an existing

software version management package to manage document versioning. The same principle can be used in other operating systems that support CVS.

In order to make version control automated, we need to replace the system command (in the UNIX server). We can do so either by aliasing commands or using wrapper programs to substitute original commands. For example, "rm" and "mv" should be replaced. "rm" is used for deleting files or for deleting files and subdirectories recursively when it is used with "-r" option. System administrators can replace original "rm" with a simple script. The script first checks if the argument filename resides under the webpage directory. If so, it will issue a CVS command to update information on the file, then execute the real "rm" command to remove the file. The same procedure can be used for "mv". Similar steps can be taken on standard editing tools such as "ed", "vi", "vim", "emacs", and "Xedit". The simplest way is to check whether a file has changed after executing editing commands. If it has been changed, then the system will update the information by issuing proper CVS commands. Note that the change of commands behavior only occurs when the argument files are within the webpage directory. But the process can be used for overall documents version control within a system. Here is a possible implementation:

[rm] "rm" is used for deleting file(s). We use a single file as an example. "rm" can be replaced by

```
original_rm FILENAME
cvs remove FILENAME
cvs commit -m ``delete FILENAME'' FILENAME
```

```
cvs release
```

where **original_rm** is the original "rm" program, and FILENAME is the name of the file to be deleted. "-m" and the string immediately following it is the comment for this modification. This comment will be recorded in the change log for the file. Removal of multiple files are carried out one at a time. The modified command has to recognize that there are multiple arguments. If the "-r" option is used by the "rm" then the above script has to be executed recursively through the subdirectories if the argument is a directory.

[**mv**] "mv" is used to rename a file or a directory.

```
original_mv FILENAME_1 FILENAME_2
cvs remove FILENAME_1
cvs commit –m ``rename FILENAME_1 FILENAME_2''
cvs add FILENAME_2
cvs commit –m ``create FILENAME_2''
cvs release
```

FILENAME_2 can be a full path or relative path for the file. For example, it can be "/home/hubert/new_directory/new_name.tex". Script has to adjust the CVS's naming convention.

[**editor programs**] There are programs to edit documents. System administrators have to write wrapper programs for them. Let's take "vi" as an example.

cvs checkout FILENAME

original_vi FILENAME

cvs add FILENAME

cvs commit –m ‘‘modified’’ FILENAME

cvs release

First, the script checks out the current version of FILENAME for the user. A user will edit the FILENAME during the “**original_vi** FILENAME” session. After the editing session, the script will automatically commit the changes to the file source depository and record the change to the log file . In terms of user’s experience, there is no difference between using the original “vi” command and using the modified “vi” command. The same process should be placed on “emacs,” “vim,” “Xedit,” and any other editing programs.

[**ftp**] **ftp** is file transferring program that will send files through the network. It can either generate new files or overwrite the existing files. System administrators can write script wrapping the original ftp program by monitoring the user’s command and modify the ftp changes on files just as we did in previous **mv, vi** examples.

[**shell redirection**] In UNIX system, users can redirect the data stream to create a file directly. For example, `echo hello > hello_file` will create a file “hello_file” with “hello” in its content. Again, system administrator can write a super-shell to wrap the original shell. Whenever user create (by “>”) or modify files (by appending “>>”), the supershell will consult CVS first. It should take care only the command line scripting. However, it is possible

that an experienced scripting user can use script to generate and modify files. This will make the supershell solution less desirable.

An alternative for the system administrators is to write a `corn` or `at` script to periodically (daily) execute a walk on the system to find and manage the newly generated or modified files. It can examine the log of the CVS to avoid repeated updating efforts. Using this scheme, we can eliminate writing individual wrapper program for editing applications. Traversing on the file system to find modified can be done at a more local scale. For example, whenever a user finishes her editing session, the system can traverse on her current working directory and her home directory to update any changes.

Because CVS handles cross site (IP address) versioning commands; the version control function can be applied to cross sites files migration. It requires, however, the installation of the version control modules on both sites. When `FILENAME_2` is an URL indicating the new location of the webpage `FILENAME_1`, modified "mv" can be executed as follows:

original_mv FILENAME_1 FILENAME_2
cvs remove FILENAME_1
cvs commit –m ‘‘rename FILENAME_1 FILENAME_2’’
cvs release

When Web server consults the CVS's change-log, the server module will discover `FILENAME_2` in the comment "`rename FILENAME_1 FILENAME_2`" is an URL and will pass the redirection directive as a response.

There are other system calls that involve modifying files, for instance, "unlink"

and “rename” are two of them. Because these are used for programming purposes, there is no need to change them.

7.3 Client Server: Network Communication - URL Address

We write a server module and attach it to a web server. For any web access request, web server will first run a CVS consulting procedure. The request for the webpage remains the same. There is no change on the client side or URLs.

7.4 Add-on Module

7.4.1 Server Side: Communicating with CVS

When a server receives a page request, it consults with CVS. CVS will list in chronological order when the requested file was changed. This information is re-formatted for menu presentation by a script. If there is only one version, the server will signal back to the enriched content client, and the content will be delivered. If the current file cannot be found, CVS still can get the previous version. Of course, if all efforts are in vain, then the message “404 file not found” has to be issued. The message could be “404 file not found after DATE” where DATE is the starting date for the version control service.

7.4.2 Client Side: Menu Presentation - LIFO Chronological Order

There are two windows for the version control. One is the content window, which is the browser; the other is the version window, which is a mini-window.

The content window shows the chosen document content; the version window has a list of different versions of this document. The user can select and view them on the content window.

When the browser issues a webpage, the server will first consult the CVS. The server module collects historical information from CVS's response, sorts it in LIFO (last in, first out) chronological order, and transforms it for menu presentation. It can be ordered in other ways: FIFO (first in, first out) or sorted by size. If the requested file was edited by a group of people, then it can be sorted by authors. Different sorting combinations can be implemented such that users select them on the client side's version window.

7.5 Benefits: Preservation, Authenticity, Comparison

7.5.1 Links Preservation

One of the advantages of version control is to maintain hyperlink validity and webpage availability. If a client browser requests a link that leads to a relocated document, then the Web server will detect it and will issue a CVS command, obtain the current location of the older version, and deliver the content. If a

browser request a page which has several past versions, then each version will be listed in the version-window so users can select and view them on the content window.

7.5.2 Archiving: Maintain Authenticity

Reference authenticity is assured if the version control module is installed. With version control, it is easy to find out what has been said in which article by whom and when.

7.5.3 Content Comparison

When doing research, we might want to compare between two hypotheses. We might also want to compare earlier and later work from the same author. Version control can assist with document comparison. Functions, such as "diff", which compares the difference between two documents, can be added. We can add other functions to compare structured XML documents.

Chapter 8

Composition Assistant

Composition assistant [1] is a phantom agent that works in the background and assists writers by suggesting URLs that are related to their writing topics. The agent first recognizes the writing application in use; it monitors the keystrokes and the text around the cursor for hints. After gathering clues on target topics, the agent seeks out information on the Web and filters the sought information for recommendations.

Goodware vs. Badware

Monitoring keystrokes of users has been used by software vendors to create user defined macros and by software companies to evaluate their product usability. Monitoring keystrokes is an old trick of hackers and crackers. On the other hand, we can use it to help writers. It could be used for good or evil purposes.

[1] **This application is described but not implemented.**

Text Context Unit

Composition agent will seek relevant URL webpage on the Web. We can only provide reference as a webpage now. In future we can reference a paragraph or a sentence through Xpointer. The reference webpage is selected by an algorithm which consults the context requested by users. The context unit can be a sentence, a paragraph, or the article produced so far. As the information will be transmitted via network, the rule of thumb of unit size is the shorter, the better. Since an article is being produced while the composition assistant is running, context information is accumulated in small size packets on the server side, gradually.

8.1 Cross System: Monitoring Keystrokes of Editor Applications

8.1.1 Identifying Editing Application

Not all document applications can edit content. We have to be recognized when this is the case. It is a convention for editing applications to show a caret when the application is ready to receive editing operations. [2]. For Windows applications, you can see the "I" shape blinking at the location where the mouse was last clicked. Enriched content program can detect the existence of the caret. `ShowCaret(HWND hwnd)` can give the last window that displayed the caret and this window can be compared with currently focused application window. If they are identical, then we know the application has editing capacity. Otherwise, it does not.

[2]Internet Explorer can show a caret as well whenever a webpage has a form submission entry. Also, the browser may show the caret at the URL input box.

8.1.2 Recording Keystrokes

Once enriched content identifies that the current application in focus is an editing application, it will start to record the keystrokes. When we set the system hook `SetWindowsHookEx` we can place not just mouse hook but also keystroke hook. By setting `WH_KEYBOARD` at `nFilterType` and `HookedKeystrokeHandler` at `hkprc` of the function call `SetWindowsHookEx(int nFilterType, HOOKPROC hkprc, HINSTANCE hMod, DWORD dwThreadID)`, we effectively pass the keystrokes information first to the `HookedKeystrokeHandler(int nCode, WPARAM wParam, LPARAM lParam)`. `HookedKeystrokeHandler` will receive every key that has been pressed through the `wParam` parameter. We can use an array to save all the typed keys. User can pre-determine the size of the context unit, and the enriched content will record the context unit accordingly.

8.1.3 Context Unit Recognition Process

The general process for context unit is illustrated as Figure 8.1. Whenever the user presses a key, the program will identify if it is a "terminal token". If it is not a terminal token, then the typed character will be appended into a string buffer. If it is a terminal token, then the saved string buffer will be copied to a new string buffer. The old string buffer will be cleared and become empty; the new string buffer will be put into the communication queue. And this process starts all over with a fresh string buffer waiting for the user to key in characters. The string buffer will save non-terminal-token characters only. Terminal tokens do not need to be a single character. Enriched content will continue sending the queued string buffers to the server until the queue is empty. Strings will be accumulated and

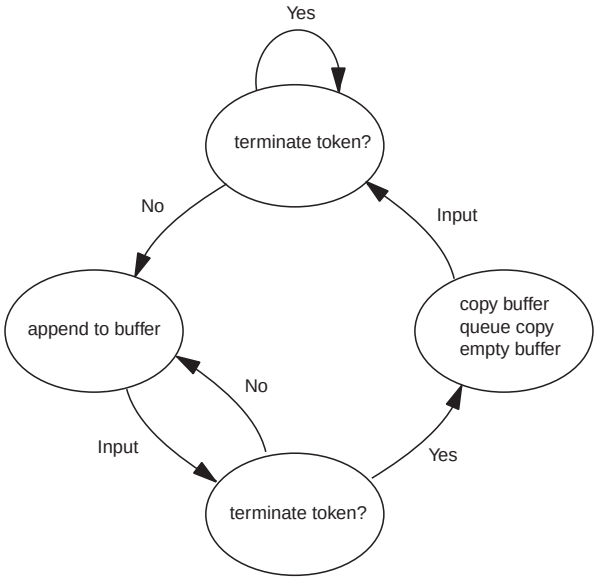


Figure 8.1: Context Unit Recognition Process Diagram

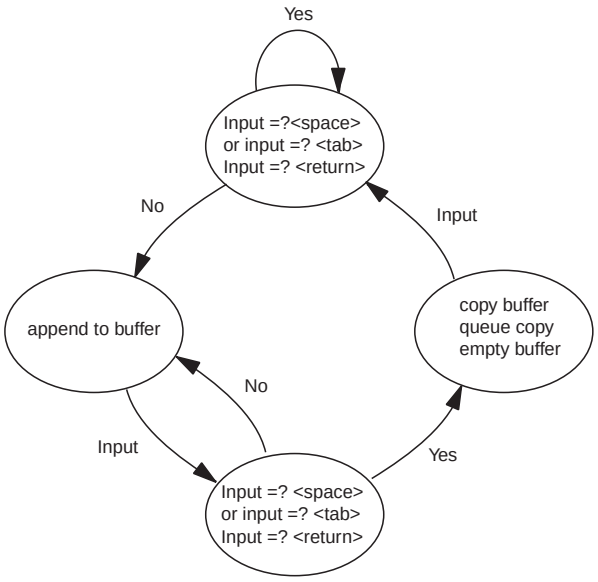


Figure 8.2: Word Recognition Process Diagram

re-assembled at the server end.

8.1.4 Text Type Recognition

We first examine how we handle the text type whose unit can be a word, a sentence, a paragraph, or the article produced so far.

Word

Let us take a word as text context unit, then we have Figure 8.2. We use "empty space" as a terminal token. Empty space can be created by a single space "\s", a tab "\t", a new line "\n" or a carriage return "<RETURN>". The string buffer will save any character until it runs into any of the above tokens.

Sentence

Recognizing a sentence is a much harder problem but we can have an ad hoc recognizer which uses period (“.”), question (“?”), and exclamation mark (“!”) as terminal tokens. We can guess that a new sentence has begun when these tokens are followed with blank space(s) and a capital letter, which is usually the indication of the beginning of a sentence.

When a user decides to change its editing location by scrolling the viewing window and choosing a location, we can use a mouse hook to trap the mouse click. Whenever there is a new mouse click, we will assume the user has relocated to a new sentential position. The new location can be identified by searching forward and backward for terminal tokens. In between the found forward and backward terminal tokens will be the current sentence. We have to take care of the case

where a "period" is used in abbreviations. An abbreviation has the characteristics that each letter is capitalized (most of the time) and each letter is followed with a period.

Paragraph

More than one "<RETURN>" can be regarded as a starting signal of a new paragraph. <RETURN> can be used as a terminal token. When a user relocates the caret location, we can use the paragraph boundary detection method mentioned in Section 4.1.3 to locate the current edited paragraph. A <RETURN> followed by a tab can be a possible beginning of a paragraph.

Article Produced so Far

A user may start to use the composition assistant when she has written some material. We can still collect the whole document on the server end. A simulation on the keystroke of "SELECT ALL" and "COPY" will copy the whole document to the clipboard. Enriched content will take the content from the clipboard and send it to the server over the network. In case the content size is too big, it can send the content by packets rather than in one bundle.

8.1.5 Link Type Recognition

Link form in text context is very simple to recognize. Hyperlink always shows up with the prefix "http://". We can use this as a cue for the hyperlinks and whenever the program retrieves a string copy from the queue, first it will examine if it is a link type and if so, it will process the context as a link type.

8.1.6 Managing Pasting and Deleting

To handle users' selecting and deleting operations, enriched content program can detect each `WM_COMMAND` using the hook with `WH_CALLWINPROCRET` option, which will execute enriched content's special codes after the `WM_COMMAND` finished its processing. Enriched content can therefore re-evaluate the sentence at the current caret location after the execution of the detected `WM_COMMAND`. This will manage the changes made by text pasting and deleting since both pasting and deleting operations are performed through the menu selection (or accelerator short hand.) and both operations will issue `WM_COMMAND` messages.

8.1.7 Basic Assistant: Synonyms Listing

Here is an example where we use a word as the context unit. A simple and yet useful composition assistant is a synonym lister. Whenever a new word is typed in by the user, the assistant program will list out all the synonyms of the word. In a composition course, we are frequently asked to find out all the synonyms for an article we have composed. Now, this process can be completely automated. Right on the computer screen, you have a group of words from which you can choose the most appropriate one. Of course, we can reduce the network traffic of requesting synonyms by calling a local application if a local thesaurus is available. For hyperlinks, we can regard "related" webpages as synonyms. For example, if the webpage requested by the link is listed under Yahoo directory, then all the rest of webpages under the same directory can be regarded as synonymous (related) webpages.

8.2 Client Server: Network Communication - CGI, Tagged Paragraph

As the service can be accessed through the network, there may be more than one user using the service. Since all context units are sent to the server to be re-assembled, there may be confusion about which text segment belongs to which user. We need to distinguish each user and her editing document by a tagging method. We tag the context unit with the user's ID (email address), the title of the currently edited document, and a time stamp.

```
http://www.server.org/cgi-bin/comp_assist.cgi?q=context_of_paragraph
&t=title_of_document&id=hubert@cs.nyu.edu
```

In this way, the service can collect correctly the text segments of each user and the editing history of the user can be profiled. Users' editing profiles help maintain consistency on suggesting relevant URLs and avoid confusion among the ownership of articles when a user changes her editing document.

8.3 Add-on Module

8.3.1 Server Side: Batched Search vs. Real Time

When a word is the context unit, searching the Web can be performed word by word. However, when a paragraph is the context unit, we have two processing methods. The first method is to analyze and extract a finished paragraph, sum up a list of searching criteria, and look them up from the Web. The second one is to extract on the fly whenever a user types in a new word. The drawback of the

second method is constant network traffic.

We can use the simple query composition method used again in the extended semantics. We can have a more sophisticated algorithm to summarize the paragraph or the whole document produced so far, and use the summary as cues to search the Web. The same summarization algorithm can be used for organizing feedback from the Web in order to respond to the user. Of course, the ideal program will be an efficient and truly intelligent algorithm that "understands" [3] human language.

8.3.2 Client Side: Menu Presentation - Keywords and Links

The window displaying suggested reference will be floating independently along side the application, as there is no fixed point of interest on a mouse click. The caret is also constantly moving as a user types. In the synonym assistant, a simple presentation will suffice: each keyword is accompanied with a set of synonyms. Keywords can be listed on the leftmost side of the window, and synonyms can be indented to the right. Clicking any one of the synonyms will bring the Webster dictionary's definition in another window whose size can be customized to save screen space.

[3] Here, "understand" is interpreted whether a program can answer question related to whatever has been provided as input. This statement, however, is vague and perhaps too generalized.

8.4 Benefits: Active Breadth and Depth Expansion

Suggestions from composition assistant can expand the width and depth of a writer's view, as the resources recommended are from other perspectives. The topic may have been written from a different angle and purpose. Suggested resources may have more citations which a writer can incorporate into her work. This process stimulates a writer to correlate her work with resources; it provokes a writer to elaborate more on her material. Composition assistant helps to keep active writing staying active.

Chapter 9

Conclusion and Near Future Work

9.1 Problems All Solved?

9.1.1 Integrating Optical Character Recognition

Though enriched content can solve many of the document application problems, we still need to integrate the optical character recognition software as many documents come in the text image format and not in digital character format. There are programs that are made for viewing purpose only, for instance, Ghostview. You can not select and copy the presented text in Ghostview.

9.1.2 Document Analysis

In extended semantics, illustrating, explaining, and associating words and phrases help but if a program knows the true context for which our words and phrases are meant, then it could pinpoint more accurate data that is crucial for us. This requires layout, syntactic, semantic, and rhetoric analysis of the document. The

effectiveness of recommended relevant information relies on the depth of understanding.

9.1.3 Higher Level Relevance from Document to Knowledge Web

Once the program has a better understanding of a document, it could read between the pages(and lines), generate a higher level abstraction of themes and topics, and drill down into details. From these analysis, it can synthesize a search criteria and locate other semantically close pages and sites and recommend them to users. It could also fill in the gaps of comprehension by offering a lower level or higher level of context. For example, it could give us an introduction to or an expertise level content. In some sense, Amazon's recommendations and Google's search are using the similar scheme, their recommendations are the result of statistically analyzing massive user data. The program eventually would construct on the fly a map of 'customized' knowledge with pages from the Web when the user request with a specific context.

9.2 Full Contextual Computing

9.2.1 Proactive Turns Autonomous

Evidently, enriched content is a proactive approach to help us to extend our "meaning domain". A conventional search is passive; an enriched content search is proactive. Studies have been done on automatically generate hypertext from text [Green 97][Allan 95]. The crucial point to successfully automatically produce hy-

perlinks is a program that can truly (or at least with high probability) understand context. So far, none exists.

Eventually, the next generation of enriched content would be agents working together to weave a new semantic web by composing a complete article or generating a solution from users' hints and Web's resources.

From Reference to Inference

From our perspective, only when a program could answer any query in regards to the context could it be considered a true contextual analyzer.

An agent would become more active by inferring for users some of the consequences given the contextual data from documents. As it is a true contextual analyzer, it could conditionally ask itself questions and answer them by drawing conclusions from collected resources. Agents would evolve from functioning in referencing capacity to functioning in an inferencing capacity.

Data is Program, Active Agent Document

Each agent program could act independently as a freelancing agent taking contextual queries one at a time and act on them. On the other hand, a webpage author could attach an agent for each document she has produced. These document agents will comprehend the given content and communicate with other requesting agents. Agents will talk to each other, access, and exchange needed knowledge. This saves time, network traffic, and storage as each agent has pre-processed its own data and understands its own mission.

By attaching a contextual agent, a document becomes "active". Each digital book, article, and report is itself a "program". Data becomes a program; document

becomes an agent.

Turing Test, Post Contextual Computing

Active agent document could continuously process on its own, it could initially get web resources and answer postulated questions raised by other views (documents, agents). It will interpret the "world" as it sees it from its own given document(s) and acquired knowledge. Making conclusions and concessions, collecting more facts, and collaborating to generate knowledge, agents will evolve and eventually are capable of passing Turing test. [Turing 50] [1]

9.2.2 Context Meets Affect

What we really desire is a fully integrated intelligent system that understands what we want and possibly before we are even consciously aware of our inclination. There is no limit for the program being empatic and not only that, it is best if a program has clues about our state of mind, intention, and emotion. This later effort is more related to affective computing. Scientists have long included "emotion" as one motivating element of intelligence. True intelligence would, and should include emotion as part of its ingrident.

[1] The Turing test was suggested by Alan Turing, the founder of computation theory. It states that when a machine can converse through teletext with a human being without being suspected as a machine, then it has the equivalent "intelligence" of a human. We think, however, the true test should include perceptual capability. The tester should show the machine a cartoon and ask what is funny or not funny about it. When the program could pass this test, it would be considered to have human "intelligence". Sense without sensation does not make sense.

Bibliography

CACM Communication of ACM

CHI Conference on Human Factors and Computing Systems

CIKM Confernece on Information and Knowledge Management

FOCS Symposium on Foundation of Computer Science

IJCAI International Joint Conference of Artificial Intelligence

IUI International Conference on Intelligent User Interface

SIGIR Conference on Research and Development in Information Retrieval

SODA Symposium on Discrete Algorithm

UIST User Interface Software and Technology

WWW International Conference on World Wide Web

[Adar 99] E. Adar, D. Karger, L. Stein. Haystack: Per-user information environment. Proceedings of the 1999 Conference on Information and Knowledge Management, CIKM, 1999.

[Adar 95] Eytan Adar, Jeremy Hylton. On-the-fly Hyperlink Creation for Page Images. ACM Proceedings of Digital Libraries '95, 1995.

[Albert 99] Réka Albert, Hawoong Jeong, Albert-László Barabási. Diameter of the World-Wide Web, Nature 401, pp. 130-131, 1999.

[Albert 00] Réka Albert, Hawoong Jeong, Albert-László Barabási. Error and Attack Tolerance of Complext Networks. Nature, July 2000, pp. 378-381.

[Allan 95] James Allan. Automatic Hypertext Construction. PhD thesis, Cornell University, 1995.

[Alon 01] Noga Alon, Joe Spencer. Probabilistic Method, 2nd edition 2001, Pritence Hall.

[Armstrong 95] Robert Armstrong, Dayne Freitag, Thorsten Joachims and Tom Mitchell. WebWatcher: A Learning Apprentice for the World Wide Web. AAAI Spring Symposium on Information Gathering. Mar. 1995.

[Aspect] Aspect programming: http://aosd.net

[Baecker 00] Ron Baecker, Sasha Jovicic, Joanna McGrenere, Gale Moore, Reducing the Gap Between What Users Know and What They Need to Know. Conference on Universal Usability, 2000, pp.17-23.

[Balabanović 95] Marko Balabanovic, Yoav Shoham. Learning Information Retrieval Agents: Experiments with Autoamted Web Browsing. AAAI Spring Symposium on Information Gathering, Mar. 1995.

[Barabási 99] Albert-László Barabási, Réka Albert. Emergence of Scaling in Random Networks. Science, Vol. 286, 15 October 1999, pp. 509-512.

[Barabási 01] Albert-László Barabási. The Physics of the Web. http://www.physicsweb.org/article/world/14/7/09

[Bharat 98] Krishna Bharat, Monika R. Henzinger. Improved Algorithms for Topic Distillation in a Hyperlinked Environment. SIGIR '98, pp.104-111.

[Bharat 98] Krishna Bharat, Andre Broder. A Technique for Measuring the Relative Size and Overlap of Public Web Search Engine, WWW7, 1998.

[Broder 00] A.Z. Broder, S.R. Kumar, F. Maghoul, P. Raghavan, S. Rajagoplan, R. Stata, A. Tomkins, and J. Wiener. Graph Structure in the Web: Experiements and Models, WWW9, 2000.

[Bordoin 01] Allan Borodin, Gareth O. Roberts, Jeffrey S. Rosenthal, Panayiots Tsaparas. Finding Authorities and Hubs From Link Structure on the World Wide Web. WWW10, May 2001, pp.415-429.

[Botafogo 91] Rodrigo A. Botafogo, Ben Shneiderman. Identifying Aggregates in Hypertex Structure. Hypertext '91, Dec. 1991, pp. 63-74

[Bouvin 99] Niels Olof Bouvin. Unifying Strategies for Web Augmentation. Hypertext 99, pp.91-100.

[Bouvin 00] Niels Olof Bouvin. Augmenting the Web through Open Hypermedia. Ph.D. Thesis, Nov. 2000, University of Aarhus, Demark.

[Bouvin 02] Niels Olof Bouvin, Polle T. Zellweger, Kaj Grønbæk, Jock D. Mackinlay. Fluid Annotations Through Open Hypermedia: Usingand Extending Emerging Web Standards. WWW 11, May 2002.

[Bradley 96] Bradley Rhodes, Thad Starner. The Remembrance Agent: A Continuously Running Information Retrieval System, First International Conference on the Practical Application of Intelligetn Agents and Multi-Agent Technology (PAAM 96), pp.486-495.

[Bradley 00] Bradley Rhodes. Margin Notes: Building a Contextually Aware Associative Memory, IUI '00, pp.219-224.

[Brin 98] Sergey Brin, Lawrence Page. The Anatomy of a Large-Scale Hypertextual Web Search Engine. WWW7, 1998.

[Budzik 99] Jay Budzik, Kristian Hammond. Watson: Anticipating and Contextualizing Informaiton Needs. Proceedgins of the Sixty-second Annual Meeting of the American Society for Informaiton Science, 1999.

[Bush 45] Vannevar Bush. As We May Think. Atlantic Monthly, 76,1, July 1945, pp.76-82.

[Carr 94] Leslie Carr. PhD Thesis, Department of Electronics and Computer Science, University of Southampton, UK. 1994.

[Carr 95] Leslie Carr, David De Roure, Wendy Hall, Gary Hill. The Distributed Link Service: A Tool for Publishers, Authors, and Readers. WWW4, 1995.

[Chakrabarti 99.1] Soumen Chakrabarti, David A. Gibson, Kevin S. McCurley. Surfing the Web Backwards. WWW8, 1999.

[Chakrabarti 99.2] Soumen Chakrabarti, Byron E. Dom, David Gibson, Jon Kleinberg, Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, An-

drew Tomkins. Mining the Link Structure of the World Wide Web. IEEE Computer, 1999.

[Chakrabarti 01.1] Soumen Chakrabarti. Integrating the Document Object Model with Hyperlinks for Enhanced Topic Distillation and Information extraction, WWW10, May 2001, pp. 211-220.

[Chakrabarti 01.2] Soumen Chakrabarti, Mukul Joshi, Vivek Tawde. Enhanced Topic Distillation using Text, Markup Tags, and Hyperlinks. SIGIR '01, 2001.

[Creech 96] Michael L. Creech. Author-oriented Link Management. WWW5, May 6-10, 1996.

[Cutting 92] Douglas R. Cutting, David R. Karger, Jan O. Pedersen, John W. Tukey. Scatter/Gather: A Cluster-based Approach to Browsing Large Document Collections. SIGIR '92, pp. 318-329.

[CVS] http://www.cbbrowne.com/info/textversion.html

[Davis 94] H.C.Davis, S. Knight, W. Hall. Light Hypermedia Link Services: A study of Third Party Integration. Proceedings of the European Conference on Hypermedia Technology (ECHT 1994), Sept. 1994. pp.41-50.

[Davis 98] Hugh C. Davis. Referential Integrity of Links in Open Hypermedia Systems. Hypertext 98, Jun 1998, pp. 207-216.

[Dean] Jeffrey Dean, Monika R. Henzinger. Finding Related Pages in the World Wide Web.

[Denoue 00] S. Denoue, L. Vignollet. An Annotation Tool for Web Browsers and its Applications to Information Retrieval. RIAO2000, Apr. 2000. http://www.univ-savoie.fr/labos/syscom/Laurent.Denou/publications/riao2000.pdf

[Dillon 94] Andrew Dillon. Designing Usable Electronic Text. 1994.

[Engelbart 84] Douglas Engelbart. Authorship provisions in AUGMENT. Proc. IEEE CompCon Conference, 1984, pp.465-472.

[Faloutsos 99] Faloutsos, M. Faloutsos, P Faloutsos, C. On Power-Law Relationships of the Internet Topology, ACM SIGCOMM '99. Computing Communication Review 29, 1999, pp. 251-264.

[Fellbaum 98] Christiane Fellbaum ed. WordNet: an Electronic Lexical Database. MIT Press, 1998.

[Finkelstein 01] Leve Finkelstein, Evgeniy Gabrillovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, Eytan Ruppin. Placing Search in Context: The Concept Revisited. WWW10, 2001, pp.406-414.

[Fountain 90] A. Fountain, W. Hall, I. Health, H. Davis. Microcosm: an Open Model With Dynamic Linking. Proceedings of the European Conference on Hypertext, INRIA, France, Nov. 90. pp.298-311.

[Gibson 98] David Gibson, Jon Kleinberg, Prabhakar Raghavan. Inferring Web Communities from Link Topology. HyperText 98, pp.225-234.

[Glover 01] Eric J. Glover, Steve Lawrence, Michael D. Gordon, William P.Birmingham, C. Lee Giles. We Search - Your Way. CACM, 2001, Dec.

[Gong 01] Yihong Gong, Xin Liu. Generic Text Summarization Using Relevance Measure and Latent Semantic Analysis, SIGIR '01, pp. 19-25.

[Google] Google, http://www.google.com

[Green 97] Stephen J. Green. Automatically Generating Hypertext by Computing Semantic Similarity. Ph.D. Thesis, Oct 1997, University of Toronto.

[Halasz 94] Frank Halasz, Mayer Schwartz. The Dexter Hypertext Reference Model, CACM Feb, '94, pp.31-39.

[Hartman 96] John H. Hartman, Todd A. Proebsting, Rajesh Sundaram. Index-Based Hyperlinks. WWW5, 1996.

[Hearst 93] Marti A. Hearst, Christian Plaunt. Subtopic Structuring for Full-Length Document Access. SIGIR 1993.

[Hofmann 00] Thomas Hofman. Learning Probabilistic Models of the Web.

[Hummingbird] Hummingbird corp. http://www.hummingbird.com

[Ingham 96] David Ingham, Steve Caughey, Mark Little. Fixing the "Broken-Link" Problem: The W3Object Approach. WWW6, May 6-10, 1996.

[iMarkup 00] iMarkup http://www.imarkup.com, 2000.

[Jansen 98]]. J. Jansen, A. Spink, J. Bateman, and T. Saracevic. Real life information retrieval: A study of user queries on the web. SIGIR Forum, 32 (1), pp.5-17, 1998.

[Jones 00] Richard M. Jones. Introduction to MFC Programming with Visual C++, 2000, Prentice Hall PTR.

[Joy 00] Bill Joy. "Why the future doesn't need us", April 2002, Wired magazine.

[Kahle 97] B. Kahle. Preserving the Internet. Scientific American, 276: March 1997, pp.82-83.

[Kahan 01] Jose Kahan, Marja-Riitta Koivunen. Annotea: An Open RDF Infrastructure for Shared Web Annotations. WWW10, 2001, pp.623-632.

[Kautz 97] Henry Kautz, Bart Selman, and Mehul Shah. The Hidden Web. AI magazine, Summer 1997, pp.27-35.

[Kenjin] http://www.kenjin.com

[Kleinberg 98] Jon M. Kleinberg. Authoritative Sources in a Hyperlinked Environment. SODA, 1998, pp.668-677.

[Kleinberg 00] Jon M. Kleinberg. Hubs, Authorities, and Communities. ACM Computing Surveys, Vol. 31, No. 4es, Dec 1999.

[Kumar 00.1] Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, D Sivakumar, Andrew Tomkins, Eli Upfal. Stochastic Models for the Web Graph. FOCS 2000.

[Kumar 00.2] Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, D. Sivakumar, Andrew S. Tomkins, Eli Upfal. The Web as a Graph. Symposium of Principle of Database, pp.1-10, 2000.

[Kurzweil 99] Ray Kurzweil. The Age of Spiritual Machines. 1999, Penguin Books.

[Glover 99] Eric J. Glover, Steve Lawrence, William P. Brimingham, C. Lee Giles. Architecture of a Metasearch Engine that Supports User Information Needs. CIKM 99, pp.210-216.

[Berners-Lee 00] Tim Berners-Lee. Weaving the Web, 2000.

[Lamport 86] Leslie Lamport. LateX, Addison-Wesley Publishing Company, 1986.

[Lassila 99] O. Lassila, R.R. Swick (eds.). Resource Description Framework(RDF) Modeland Syntax Specification. Recommendation, W3C, Feb. 1999.
http://www.w3.org/TR/1999/REC-rdf-syntax-19990222

[Lempel 00] Ronny Lempel, Shlomo Moran. The Stochastic Approach for Link-Structure Analysis (SALSA) and the TKC Effect. WWW9, 2000.

[Lieberman 97] Henry Lieberman. Autonomous Interface Agents. CHI 97, pp. 196-176.

[Lieberman 00] Henry Lieberman, Ted Selker. Computer Systems That Adapt To, and Learn From, Context. IBM, System Journal, 2000, V.39, No. 3&4, pp.617-631.

[Maes 94] Pattie Maes, Agents that Reduce Work and Information Overload. CACM, July 1994, V.37, No.7, pp.31-40.

[Maglio 00] Paul P. Maglio, Rob Barrett, Christopher S. Campbell, Ted Selker. SUITOR: An Attentive Information System. IUI 00, pp. 169-176.

[Marais 97] Hannes Marais, Krishna Bharat .Suppporting Cooperative and Personal Surfing with a Destkop Assistant. UIST 97, pp.129-138.

[Marcus 93] Aaron Marcus. Human Communications Issues in Advanced UIs. CACM, April 1993, Vol. 36, No.4.

[Merant] Merant corp. http://www.merant.com

[Milgram 67] S. Milgram. The Small World Problem. Psychology Today 1, 61, 1967.

[mod_ruby] Ruby module for Apache server. http://www.modruby.net

[Moore 65] Gorden E. Moore. Cramming More Components onto Integrated Circuits. Electronics, Apr. 19, 1965.

[Mosaic 93] NCSA Mosaic Documentation. Group Annotations in NCSA Mosaic, 1993.
http://archive.ncsa.uiuc.edu/SDG/Software/XMosaic/Annotations/overview.html
http://www.ncsa.uiuc.edu/SDG/Sfotware/Mosaic/Docs/group-annotations.html

[Ng 01.1] Andrew Y. Ng, Alice X. Zheng, Michael I. Jordan. Stable Algorithms for Link Analysis. SIGIR '01, pp. 258-266.

[Ng 01.2] Andrew Y. Ng, Alice X. Zheng, Michael I. Jordan. Link Analysis Eigenvetors and Stability. IJCAI '01.

[Nelson 92] Theodor Holm Nelson. Literary Machines 91.1, 1992.

[Oclc 01] http://wcp.oclc.org/stats/size.html

[Oetiker 01] Tobias Oetiker. The Not So Short Introduction to LaTeX 2e. http://www.ctan.org

[Page 98] Lawrence Page, Sergey Brin, Rajeev Motwani, Terry Winograd. The PageRank Citation Ranking: Bringing Order to the Web. http://www-db.stanford.edu/ backrub/pageranksub.ps

[Penrose 91] Roger Penrose. The Emperor's New Mind: Concerning Computers, Mind, and the Laws of Physics, Penguin, Jan 1991.

[Pettengill 95] R. Pettengill, G. Arango. Four Lessons Learned from Managing World Wide Web Digital Libraries. Digital Libraries '95 : The Second Annual Conference on the Theory and Prctice of Digital Libraries, Jun 1995, pp. 177-180.

[Petzold 99] Charles Petzold. Programming Windows, Fifth edition, Microsoft Press, 1999.

[Phelps 98] Thomas Arthur Phelps. Multivalent Documents: Anytime, Anywhere, Any Type, Every Way User-Improvable Digital Documents and Systems. Ph.D. Thesis, Department of Computer Science, University of California, Berkeley, 1998.

[Pirolli 96] Peter Pirolli, James Pitkow, Ramana Rao. Silk from Sow's Ear: Extracting Usable Structures from the Web. CHI

[Pitkow 97] James Pitkow, Peter Pirolli. Life, Death, and Lawfulness on the Electronic Frontier. CHI 97, pp. 383-390.

[Pitkow 96] James E. Pitkow, R. Kip Jones. Supporting the Web: A Distributed Hyperlink Database System. WWW5, 1996. http://www.image.fr/Multimedia/www5cd/www396/overview.htm

[Reckdahl 97] Keith Reckdahl. Using Imported Graphics in LaTeX 2e.

[Rissanen 89] J. Rissanen. Stochastic Complexity in Statistical Inquiry, World Scientific Series in Computers Science, Vol. 15, World Scientific, Singapore, 1989.

[Rafiei 00] Davood Rafiei, Alberto O. Mendelzon. What is this Page Known for? Computing Web Page Reputations. WWW9, 2000, pp. 823-835.

[Rafiei 01] Alberto O. Mendelzon, Davood Rafiei. What do the Neightours Think? ComputignWeb Page Reputation.

[Robertson 93] George G. Robertson, Stuart K. Card, Jock D. Mackinlay, Information Visualization Using 3D Interactive Animation. CACM, April 1993, V.36, No.4, pp.57-71.

[Röscheisen 95] Martin Röscheisen, Christian Mogensen, Terry Winograd. Beyond Browsing: Shared Comments, SOAPs, Trails, and On-line Communities. WWW4, 1995.

[Ruby] Ruby Scripting Language: http://www.ruby-lang.org

[Salton 75] Gerard Salton. A Theory of Indexing, regional Conference Series in Applied Mathematics, SIAM, 1975.

[Salton 89] Gerard Salton. Automatic Text Processing, Addison-Wesley, 1989.

[Satyanarayanan] M. Satyanarayanan. Pervasive Computing: Visionand Challenges. IEEE Computer?

[Semantic] Semantic Web. http://www.w3.org/2001/sw

[Shneiderman 88] Ben Shneiderman (Editor). Hypertext on Hypertext, Hyperties disk with 1M byte data and graphics incorporating. CACM, July 1988.

[Shneiderman 00] Ben Shneiderman. Universal Usability. CACM, May 2000, Vol.43, No.5, pp.84-91.

[Silber 00] H. Gregory Silber, Kathleen F. McCoy. Efficient Text Summarization Using Lexical Chains. IUI 2000, pp.252-255.

[Simon 96] Richard Simon. Win 32 programing, API Biblle, Waite Group Press, 1996.

[Simmonson 98] J. Simmonson, D. Berlean, X. Zhang, M. Xie, and H. Vo. Version Augmented URI's for Refernece Permanence via an Apache Module Design. WWW7, April 1998, pp.337-345.

[Third 99] Third Voice, 1999. http://www.thirdvoice.com

[Thomas 01] Programming Ruby: The Pragmatic Programmer's Guide, 2001, Addison-Wesley.

[Travers 69] J. Travers and S. Milgram. An Experimental Study of the Small World Problem. Socimetry 32, 425, 1969.

[Turing 50] Alan Turing. Computing Machinery and Intelligence, Mind, 59(236) pp.433-460, 1950.

[Vitali 95] F. Vitali, D.G. Durand. Using Versioning to Provide Colaboration on the WWW. WWW4, Dec 1995, pp.37-50.

[Web] http://www.mit.edu/people/mkgray/net/web-growth-summary.html

[Whitehead 97] E.J. Whitehead Jr. An Architectural Model for Application Integration in Open Hypermedia Environment. ACM Hypertext Conference, Apr. 1997. pp. 1-12.

[Whitehead 98] E.J. Whitehead Jr., M. Wiggins. WEBDAV: IETF Standard for Collaborative Authoring on the Web. IEEE Internet Computing, 2(5), 1998, pp.34-40.

[WindRiver] WindRiver corp. http://www.windriver.com/products/html/os.html http://www.windriver.com

[Wood 00] L. Wood et al. (eds.) Document Object Model (DOM) Level 2 Specification Ver. 1.0. CR, W3C, Mar 2000. http://www.w3.org/TR/2000/CR-DOM-Level-2-20000307

[Xdrive] Xdrive. http://www.xdrive.com

[Yee 98] K.-P. Yee. CritLink. Better Hyperlinks for the WWW. Submitted to Hypertext '98, Apr. 1998. http://crit.org/ http://crit.org/~ping/ht98.html

[Zellweger 98] Polle T. Zellweger, Bay-Wei Chang, Jock D. Macinlay. Fluid Links for Infomred and Incremental Link Transitions. Hyptertext '98, Jun. 1998, pp. 50-57.

[Zipf 49] Zipf, H. Human Behavior and the Principle of Least Effort, 1949, Addison-Wesley, Cambridge, MA.