Assisting the Hypertext Authoring Process with Topology Metrics and Information Retrieval

Gavin Gollogley

M.Sc. 1997

Assisting the Hypertext Authoring Process with Topology Metrics and Information Retrieval

By

Gavin Gollogley B.Sc.

School of Computer Applications,
Dublin City University,
Glasnevin,
Dublin 9.

Supervisor: Prof. Alan F. Smeaton

A dissertation submitted for the degree of Master of Science

Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Masters of Science in Computer Applications, is entirely my own work and has not been taken from the work of others save to the extent that such work has been cited and acknowledged within the text of my work

Signed [illegible] Date 22-9-97

Acknowledgements

Many thanks to my research supervisor Prof Alan Smeaton for his advice and direction in the undertaking of this thesis Thanks also to Forbairt for providing the funding which enabled me to carry out this research

Thanks to my parents who have encouraged and supported me for as long as I can remember

I would like to thank Fran, Ciaran, David, Michelle and in particular Dr Fergus Kelledy for their assistance Special thanks to Fiona and Áine for forcefully steering me in the right direction

Finally, I'd like to thank all my friends inside and outside DCU for providing irrefutable proof that there is more to life than work alone (and thanks for the postcards!) Thanks in particularly to those people who acted as 'guinea pigs' for the results part of my thesis, namely Darragh, Deirdre, Donal, Gary, Gerry, Martina, Noel, and Peter

ABSTRACT

As more and more documents become available in electronic format, the use of hypertext systems is becoming more common as a way to organise information However as the size of a hypertext database grows, the 'lost in hyperspace' problem may limit efficient and meaningful usage of hypertext systems

In order to increase local coherence at net level, authors should limit 'the fragmentation characteristics of hypertext' These characteristics seem to be endemic to hyper-documents and result from the segmentation of information into disjointed nodes Fragmentation may result in a lack of interpretative context and thus lead to the impression that the hyper-document is an aggregation of loosely linked pieces of information rather then a coherent whole In an attempt to understand a new node, readers try to extract information and relate it in context to other nodes that they have viewed In this thesis we describe an application which incorporates an apprentice link editor to suggest candidate information/hypertext links for the hypertext author to validate These 'suggestions' use node-to-node comparison and metrics to present the author with the most appropriate choices in adding a new node to the hypertext they are authoring

Although this project is intended to provide a hypertext author with the tools needed to enhance the construction of a large hypertext, the assumption of trusting the author, and the notion that the best quality control tool is a skilled author is never forgotten

Table of Contents

Chapter 1 - Introduction

1.1 Introduction

‘Authoring involves identifying links within information and structuring them to enhance accessibility Developers of large hypermedia systems, or systems likely to require maintenance over a period of time, must select a structured approach to authoring or risk greatly inflated development cost, a less than desired functionality, and difficulties in use and maintenance’ [Gini95]

The information revolution is upon us [Gini95] Due to the technology increases in the areas of databases, computer hardware and communications networks, our ability to quickly and easily access information has improved However there is now an overgrowth of information People today are faced with so much information in so many different media that the time required to absorb and analyse is becoming increasingly prohibitive We also find it difficult to identify areas of information that are most relevant to our needs [Gini95] Many techniques have been tried in order to sift through this sea of information Most however have looked into the possibilities of methods to search and retrieve information instead of looking at the organisation of this information Improving the organisation of information makes information easier to locate and assess its relevance and put it to use

Multimedia has now become a buzz word of the late 20th century The ability to display media such as sound, picture and video means that knowledge and information can be presented in a manner that increases understanding and facilitates education Hypermedia and hypertext lets us organise information in accordance with the ways in which we naturally access and manipulate it A hypertext is a collection of documents that contain cross references or links to each other Hypermedia is an extension of hypertext to allow graphics, sound, video, etc Both should present information in such a way that shows conceptual associations between information chunks which reveal the structure and interrelationships within the information The form requires manageable chunks that can be joined by links The link structure is important, but not more so than the composition of the disparate fragments Each time the user stops moving through the medium, the fragment he or she confronts must be able to stand alone, to make sense on its own, even though it offers multiple doorways to other places If the chunk does not give the impression that it stands alone, the users will feel as if something is missing, as if the information is inadequate, incomplete [McAd93] Both hypermedia and hypertext are handled similarly in design and refer to organised sets of information linked by semantic relationships and as such are viewed indifferently for the purposes of this thesis

Authoring is the process of creating and storing information in a fashion appropriate to its intended uses This often involves either transforming existing information sources or creating new information to populate a hypermedia information repository [Gini95] Sometimes the mistake is made that authoring is the same as publishing This is not so as publishing involves issues like layout and usability A lot of hypertext tools claim to be authoring tools but are little more then publishing tools e g HotDog Authorship involves identifying structures for information that will support accessibility and manipulation The author must adopt suitable methodologies for generating these structures Two issues must be addressed in the construction of a hypermedia application how to structure the information and what authoring approach to take

1.2 Information structure

An appropriate structures for a hypertext needs to be identified to maximise accessibility to the information it has on offer These structures affect the authoring process as it is through the authoring process that these structures are typically generated The following factors affect the design of information structures

- information retrieval and accessibility
- information security
- information reuse and maintainability
- relationships between information sources
- provision of differing viewpoints on the information

Authors should try to make explicit the knowledge representations inherent within the information In the construction of a hypertext, information is broken down into nodes which should represent one concept and one concept only and should lose all uselessness if broken down further [Gini95] These nodes must then be interlinked via what are known as anchors This is the process of identifying anchor points within each node that form the starting points for links to other nodes

Hypertexts can be classified under 3 different types of structure, linear, hierarchical and network (see Figure 1 1)

- Printed material is usually presented in a linear manner This involves users having to complete one topic before moving on to the next e g training material Using a linear structure, information can be kept in a sequential manner It may be unwise for the hypertext author to alter the original structuring of a linear document as it could remove any sort of value from the text A linear structure can be found in guided tours where information is presented in logical progression These could also serve as useful teaching tools
- Hierarchical structure resembles that structure found in books - table of contents, chapters, sections, paragraphs, etc Like a linear structure, it would be advisable for the author to keep the structure that the paper version of the document was in In this structure the root node is usually the starting point and the structure is made up of hierarchical links (see Section 2 2 8 for more detail on the different types of links available) Systems such as KMS (Knowledge Management System) use hierarchical structure as the base structure [Aksc88]
- A network structure consists of relational or associative links These are semantic in nature and are non sequential They link common or related information together An example of this may be a node on hypertext with links to a node on the WWW which in turn may link to information on setting up a web page This would suit information in an encyclopaedia best The ability to browse information that is in this particular structure, would be seen by many as the major advantage of hypertext/hypermedia

The author would pick the structure needed depending on what sort of information is being presented, what material will be included and how the information is to be accessed i e table of contents, search, index or full text

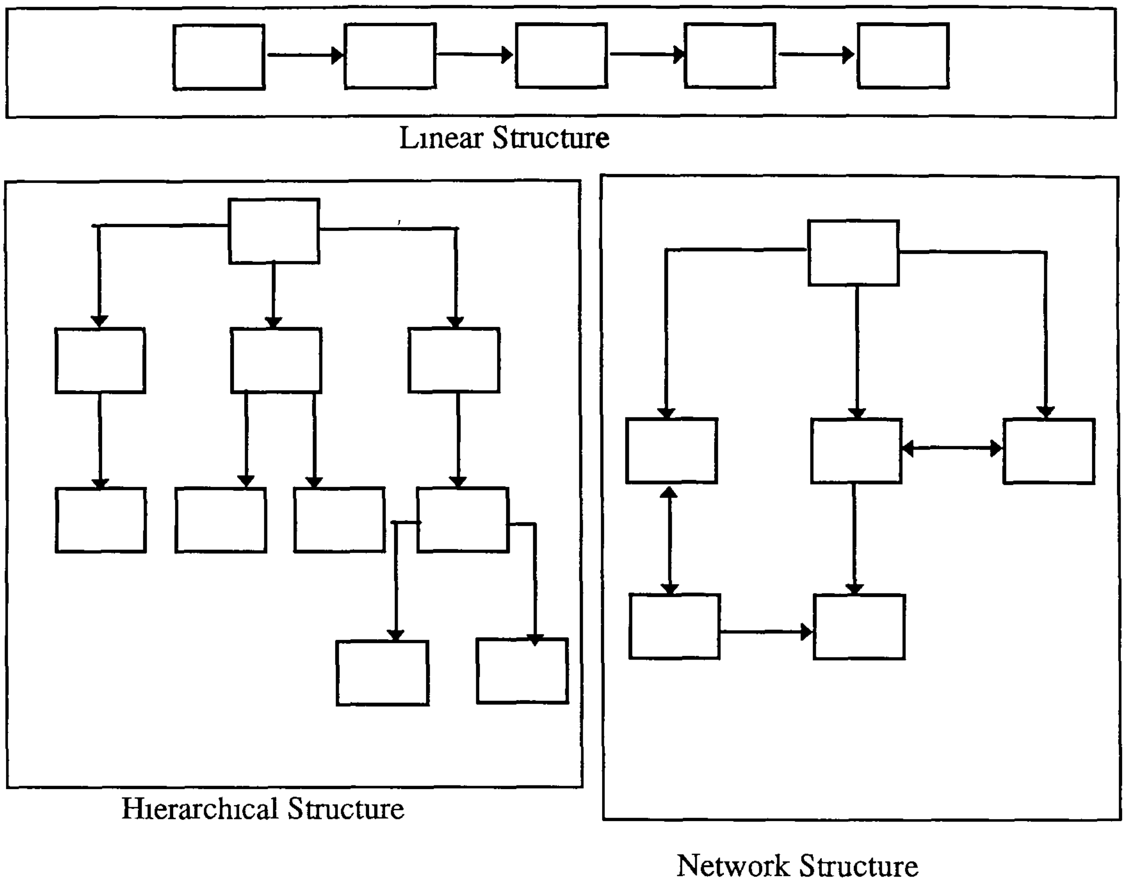


Figure 1.1 The 3 types of information structures

1.3 Hypertext

The hypertext approach to information management is to separate documents or other information units into parts or fragments, where fragments are stored and managed in a network of nodes, where each node of the network contains one or more fragments and related nodes are connected through information links [Agos96] Before we go on to give a brief history of hypertext, we must give definitions of some of the terminology which is used in the construction and reading of hypertext The following are the main terms associated with hypertext

- **Node** A node contains any kind of digitised data that can be managed and presented through a computer using its screen and any other output device. It can be a fragment of text, a graph, a drawing, a sound byte, a video clip or any other possible form of data or a combination of these [Agos96].

- **Link**: A link implements a logical connection between two related nodes; the origin is the node from which the logical connection between the two fragments emanate; the destination is where a connection between nodes ends. Two nodes that should be read sequentially should be linked. Linking nodes offers the possibility of creating multiple pathways throughout a hypertext [Agos96]. There are different types of links, reference, relational and structural (see Section 2.2.8 for more detail on the different types of links available). Links can also be unidirectional (a link going from one node to another but no link back) or bi-directional (a links exist in the reverse direction allowing inverse relationships).
- **Anchors**: The starting point for a link is known as an anchor point. By clicking on an anchor, the link will be traversed and the user will be presented with the node associated with that link. This process is known as navigating a hypertext. Anchors can be embedded within the text of a node or outside the text.
- **Path**: An ordered set of nodes which represent a sequence in which a web can be read. This path may represent the sequence of nodes the user actually read or the sequence the author recommends the user to take [W3CR97].
- **Web**: A set of nodes interconnected by links. Usually refers to the set of all nodes that are interconnected.
- **Topology**: The allowable connectivity between nodes, anchors and links e.g. a 1 to 1 or a 1 to many mapping. This displays the properties of a hypertext, the connectivity of the structure due to the number of links and anchor placements.
- **Typed Links**: A typed link carries some semantic information, which effectively increases its information content. Link types can express relationships between the information described by 2 nodes:

  A Is part of B / B includes A

  A Made B / B is made by A

  A Uses B / B is used by A

  A refers to B / B is referred to by A [W3CR97].

Annotation: This is the linking of a new commentary node to an existing node via an annotation. If users can annotate nodes, then they can immediately provide feedback if the ınformatıon ıs mısleadıng, out of date or plaınly wrong Thus the qualıty of the ınformatıon ın the web can be ımproved [W3CR97]

1.3.1 Hypertext History

The ıdea of hypertext has been around sınce the 1930's It may have been Ted Nelson who 'coıned' the term hypertext ın 1965 but Vannevar Bush had conceıved the ıdea ın the 1940's Hıs artıcle was publıshed ın the Atlantıc Monthly under the tıtle of 'As We Mıght thınk' [Bush45] He proposed a MEMEX devıce that was desıgned around the concept of assocıatıon, whıch would facılıtate the large number of scıentısts who had to be redeployed after World War II Informatıon would be selected by assocıatıon, rather then conventıonal ındexıng, a means of sharıng ınformatıon and ensurıng thıs ınformatıon was never lost He also came up wıth the notıon of 'traılblazers', ındıvıduals who ıdentıfied lınks or traıls between pıeces of ınformatıon and could store them ın the MEMEX devıce [Dunn93] Thıs system was never buılt but proved to be the foundatıon of hypertext

In the 1960's Doug Engelbart worked on the AUGMENT project to buıld tools to help humans and they ınvented the mouse, word processıng and wındows on VDU's As part of the project, Engelbart developed NLS (oN-Lıne System) whıch allowed hıs researchers to store all theır papers and reports ın a shared lıbrary Thıs system had several hypertext features but was not desıgned to be a hypertext [Morr94] It was first demonstrated at a conference ın 1968 and up untıl recently, NLS was marketed as a hypertext-lıke commercıal product

In 1965 Nelson first used the term 'hypertext' He vısualısed a system called Xanadu whıch would be a common access world deposıtory of computer readable lıterature and other medıa and ın fact would ıncorporate everythıng that was ever wrıtten Not surprısıngly thıs never happened but a small, reduced versıon was released ın 1990 [Nels80]

1967 saw the first real hypertext system beıng produced by Adrıes van Dam at Brown Unıversıty Thıs system was known as HES (Hypertext Edıtıng System) and incorporated a text based user interface and also had linking and jumping from text to text Van Dam later designed FRESS (File Retrieval and Editing System) [Donn95]

ZOG was the name of a well known hypertext system developed at Carnegie-Mellon in the 1970's It was a frame based text database with hierarchical links Each frame contained a single screen of information and a menu in the form of link options providing multi user access The commercial version of this is now known as KMS (Knowledge Management System) [Donn95]

The Symbolics Document Examiner was a hypertext system whose subject matter was the Symbolics LISP machine which is a workstation for LISP processing This contained 8,000 pages of printed text Developed by Jan Walker it included 10,000 nodes and 23,000 links [Donn95] This was the first hypertext application to see real world use

1986 saw the first hypertext system that was commercially available and could run on personal computers This system was GUIDE from Office Workstations Limited (OWL) and was developed by Peter Brown at Kent University

In 1987, Apple Computers decided to put a hypertext system called HyperCard on every Apple machine Originally it was not meant as a hypertext system but as a rapid prototyping tool and it contained a lot of hypertext features It combined a relational database with the ability to integrate other media (sound and video) using the Mac's graphical interface HyperCard was given away virtually for free and this tremendous marketing strategy caused a sudden increase in the levels of interest in hypertext and brought about the first ACM Hypertext Conference in 1987 at the University of Carolina Since then there have been 8 major conferences on Hypertext in total, with the most recent conference, Hypertext 97 being held in Southampton in the UK

In 1989 an idea was proposed to create a user-friendly interface to the Internet, the World Wide Web (WWW) At a basic level the Internet is a set of computer networks interconnected with routers It is a three level hierarchy composed of backbone networks (e g ARPAnet, NSFNet, MILNET), mid-level networks, and sub networks

These ınclude commercıal (com or co), unıversıty (ac or edu) and other research networks (org, .net) and mılıtary (mıl) networks and span many dıfferent physıcal networks around the world wıth varıous protocols ıncludıng the Internet Protocol [Howe97] The Internet ıncludes FTP (Fıle Transfer Protocol), TELNET, e-maıl, Gopher, Archıe and the World Wıde Web

The WWW was developed at CERN ın Geneva It ıs a dıstrıbuted, heterogeneous, hypermedıa ınformatıon system. It ıs dıstrıbuted because there are mıllıons of Web servers around the world provıdıng access to web documents and heterogeneous because ıt ısn't bound to any partıcular hardware and ıs accessıble to any computer wıth an Internet connectıon [Morr94] It was released ın May 1991 and has sınce seen phenomenal growth Thıs meteorıc rıse ıs due to a number of factors ıncludıng the fact that all software needed to set up a Web Server or vıew Web documents usıng a Web Browser ıs free Another of ıts major advantages ıs the fact that HTML (Hypertext Markup Language, see Sectıon 1 3 2) the mark-up language used ın wrıtıng a Web node ıs very sımple to wrıte and graphıcal dısplay of ınformatıon ıs easy to produce Fıgures 1 2a and 1 2b show two surveys, one from 91-97, the second from 93-97 of the number of hosts connected to the Internet, where a host computer ıs a computers connected to the Internet

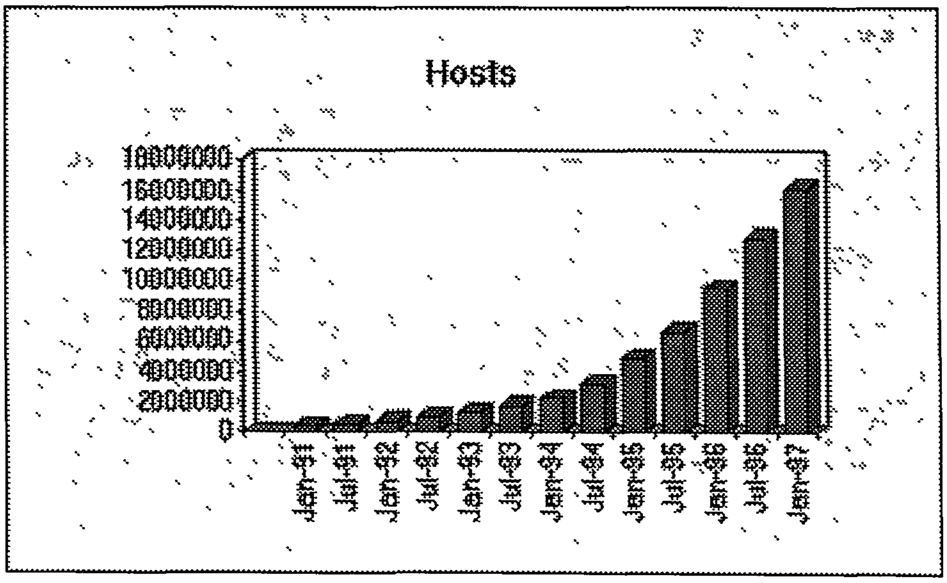


Figure 1.2a Host Count Graph (1991-1997) [Netw97]

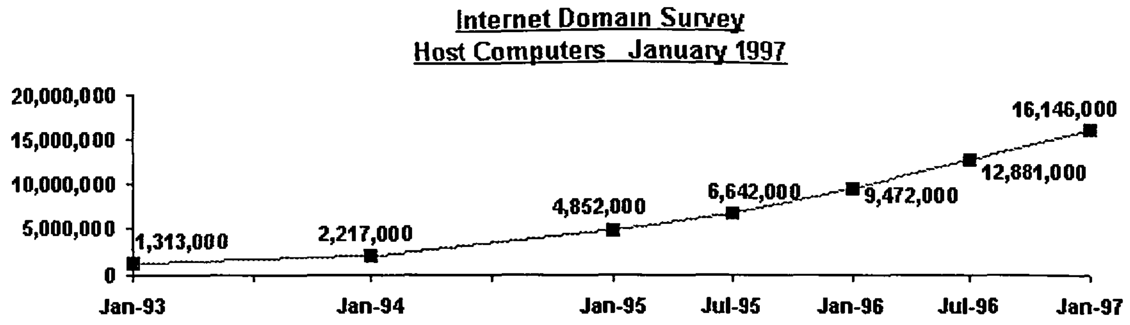


Figure 1.2b Host Count Graph (1993-1997) [Netw97]

- Netree's Internet Statistics (www netree.com/netbin/internetstats) (based on Internet statistics estimates from Internet Business Centre) reports 102 0 million people on the Internet, and 1 6 million WWW sites on the Internet, world-wide, 16 January 1997 ('on the Internet' means all persons with access to WWW, e-mail, ftp, gopher, and Telnet services, and 'sites' means domains offering web pages.)
- International Data Corporation (IDC) (www idcresearch com) reports 31 4 million users of the WWW, world-wide, 31 October 1996

The WWW conforms to the HTTP (Hypertext Transport Protocol) which is client/server HTTP servers provide Web clients with hyperlinks to nodes on the same Web-Site or anywhere else on the WWW Resources available to WWW servers are named using a method known as Uniform Resource Locator (URL) which can be used to identify almost any object available on the Internet URLs require the protocol necessary to obtain the information, the system where the information is stored and the location of the information on that system e g

`http //www.compapp.dcu.ie/index.html`

URL's can also have other protocols apart form HTTP

- `ftp://ftp funet.fi/pub/Linux`

  FTP, the File Transfer Protocol which allows a user on one computer to transfer files to another over a TCP/IP network

- `gopher.//gopher.utirc utoronto.ca`

  Gopher, a popular distributed document retrieval system (Gopher was largely superseded by the WWW)

- `news.news announce.newusers`

News refers to the protocol USENET, A distributed bulletin board system supported mainly by UNIX machines and the people who post and read articles thereon

To look at the WWW and the way it is used for presenting and structuring information on a variety of topics, it would be impossible not to explain what the underlying format was for all Web documents, namely HTML This next sections gives an overview of HTML and some examples of its tag elements structures

1.3.2 Hypertext Markup Language (HTML)

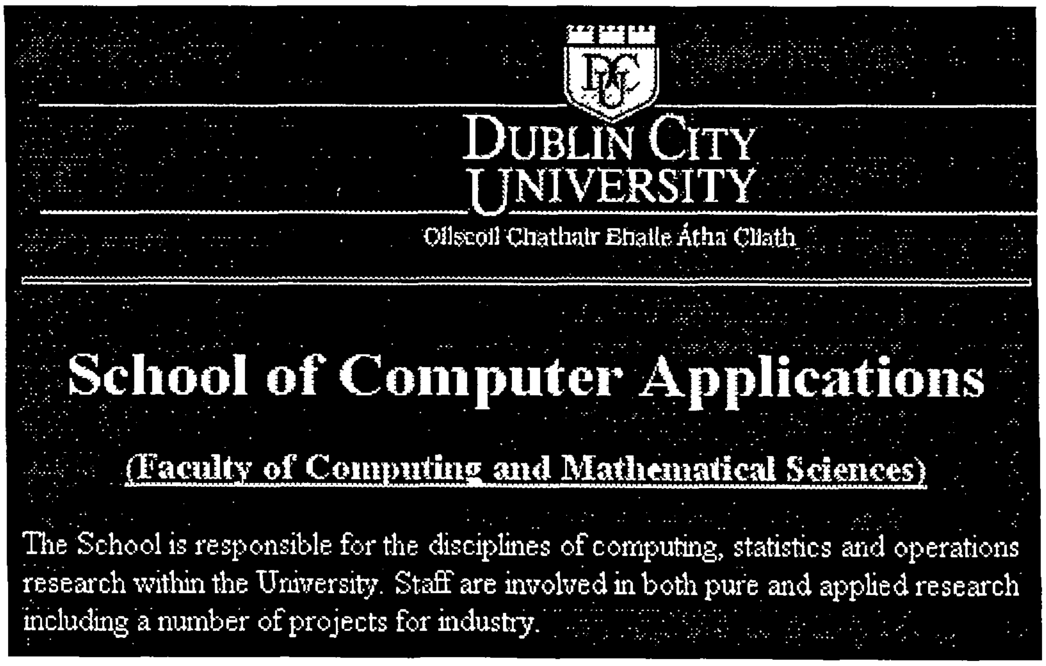
Hypertext Markup Language, or HTML for short, is a subset of the International Standards Organisation (ISO) Standard Generalised Markup Language (SGML) which is used for authoring and publishing large structured documents [Mich94] (see Chapter 2 Section 2 2 6 Standards) HTML provides formatting information such as codes for the node title, headings, paragraphs and general document structure as well as allowing the encoding of hypermedia anchors which contain embedded URL links [Donn95] HTML also provides full hypermedia facilities like graphics, sound, moving picture and animation Some examples of HTML tags are as follows

<HTML> </HTML>	Delimits the HTML portion of a node
<TITLE> </TITLE>	Delimits the title of the node
<H1> . </H1>	Delimits a font size which would appear as the largest to the reader (H6) is the smallest font size
<B> </B>	Wraps a piece of text in a bold format
<A HREF = “URL”> NAME </A>	This defines a link to a URL with the anchor defined after it until the terminating </A>
<IMG SRC = “URL”>	Displays a graphic image found at this URL address
<HR>	Draws a horizontal line

An example of a tagged document and the way it looks through a Web Browser (Netscape) is shown in Figure 1 3a and 1 3b

```
<HTML>
<HEAD>
<TITLE>School of Computer Applications</TITLE>
</HEAD>
<BODY    background=". /images/dcublue gif"    fgcolor=#000000    text=#FFFBF0
link=#FFFF00 alink=#FF0000 vlink=#FFBB00>
<CENTER><IMG ALIGN="Center" SRC=". /images/masthead gif" ALT="Dublin City
University"></CENTER>
<HR Size=3>
<CENTER><H1>School of Computer Applications</H1></CENTER>
<CENTER><H3><A   HREF="../comp_math html">(Faculty   of   Computing   and
Mathematical Sciences)</A></H3></CENTER>
<P>The School is responsible for the disciplines of computing, statistics and operations
research within the University. Staff are involved in both pure and applied research
including a number of projects for industry </P>
```

Figure 1.3a HTML Markup of a Web page [Scho97]



School of Computer Applications

(Faculty of Computing and Mathematical Sciences)

The School is responsible for the disciplines of computing, statistics and operations research within the University. Staff are involved in both pure and applied research including a number of projects for industry.

Figure 1.3b Marked up HTML page as it is viewed through a Web Browser (Netscape) [Scho97]

1.4 Hypertext Systems

Although the WWW has taken over as the most popular format for presenting hypertext, there are other hypertext systems Hypertext systems are emerging as a new class of complex information management systems These systems allow people to create, annotate, link together, and share information from a variety of media such as text, graphics, audio, video, animation, and programs Hypertext systems provide a non-sequential and entirely new method of accessing information unlike traditional information systems which are primarily sequential in nature They provide flexible access to information by incorporating the notions of navigation, annotation, and tailored presentation [Bieb93] A hypertext system should have the following capabilities

- A Graphical User Interface, with the help of browsers and overview diagrams which helps the user to navigate through large amounts of information by activating links and reading the contents of nodes
- An authoring system with tools to create and manage nodes (of multiple media) and links
- Traditional information retrieval (IR) mechanisms such as keyword searches, author searches etc There are also attempts to incorporate structured queries along with content queries - retrieving a part of the hypertext network based on some user-specified criteria
- A hypermedia engine to manage information about nodes and links
- A storage system which can be a file system, a knowledge base, a relational database management system or an object-oriented database management system [Bala94] We will now give examples of some of the best known hypertext systems

1.4.1 Hyper-G

Hyper-G is a large scale multi-protocol, distributed, hypermedia information system which uses an object orientated database layer to provide the following

- information structuring and link maintenance facilities
- fully integrated attribute and content search
- hierarchical access control scheme
- interactive link editing
- interactive preferences
- consistent look and feel
- point-and click document insertion

Hyper-G combines the intuitiveness of top-down hierarchical navigation with the immediacy of associated hyperlinks and the power of focused attribute and content searches [Orch95] Hyper-G is being developed jointly by the Institute for Information Processing and Computer Supported New Media (IICM) of Graz University of Technology, Austria and the Institute for Hypermedia Systems of Joanneum Research, Graz, Austria [Prad95] HyperLinks connect a source anchor within a document to a destination anchor within another document, an entire document, or a collection of documents The links are stored within a link database, so they are bi-directional, updatable (no dangling links), and visualizable

Hyper-G has been developed carefully to ensure cross-operability with WWW Hyper-G databases have gateways to WWW and Gopher Hyper-G introduces a sophisticated authorisation mechanism defining for each user the rights to read, create links, modify and annotate This provides the basis for sophisticated customisation, and even CSCW (Computer Supported Co-operative Work) within Hyper-G that have to be, like all other more sophisticated features, built on top of WWW (potentially creating confusion and incompatibility) Hyper-G, as a late-comer in the field, has been able to profit from and incorporate experience from earlier projects such as Gopher and WWW [Maur94]

1.4.2 Hyperties

Hyperties was designed as TIES (The Interactive Encyclopaedia System) under the stewardship of Ben Shneiderman at the University of Maryland's Human-Computer Interaction Laboratory It provides both authoring and browsing capabilities A node in

Hyperties may contain an entire article that may consist of several pages Links are represented by highlighted words or embedded menus which can be by pointing at them. Readers can preview links before actually traversing them. When a user chooses to select a node, the node in question is not immediately displayed Instead a short summary of the node appears first, allowing the user to decide whether it is worth their while viewing the full node The authoring and browsing capabilities of Hyperties are separated The authoring tool allows the author to import text or graphics into nodes Links are embedded into the text, with the author selecting the text chosen to be an anchor and the system responds by automatically looking up a table of node names to establish a link Hyperties has been used by NASA as a hypertext on the Hubble Space Telescope, as well as many interactive encyclopaedia's

1.4.3 NoteCards

NoteCards was developed at Xerox PARC by Randall Trigg, Thomas Moran and Frank Halasz It is a hypermedia system for authors and researchers to analyse information, construct models, formulate arguments and process ideas [Hala88] Its two main components are notecards and links The notecards are used to store and organise information and have the same functionality as a node The links between the various notecards are typed and directional The user specifies the type of link to communicate the nature of the relationship [Dunn93] The browsing facility of NoteCards is displayed on a notecard that contains a diagram of the complete structure of the notecards being used Users are allowed alter the structure and the general makeup of these diagrams by adding links and nodes NoteCard also offers a function that uses 'fileboxes' to categorise and hierarchically structure notecards

1.4.4 Microsoft HtmlHelp System

This help system which is planned for release in August 1997 by Microsoft is designed for displaying context-sensitive help, table of contents and indexing and will be viewable using Microsoft's Web Browser 'Internet Explorer' or most other Web

Browsers as it will be written in HTML It is aimed at authors who create online Help for software or Web Sites [Hear97] and is made up of 5 components which are briefly

- **HTML ActiveX Control** ActiveX is a medium in which one can write small code modules that run on the Internet It is made up of C++ and Object Linking and Embedding and is a rival to Sun Microsystems Java HtmlHelp provides a wizard to place ActiveX control into a HTML document [Hear97]
- **Compressed HTML** Compressed HTML allows for the quicker transfer of HTML over the Internet as well as taking up less space on a computer hard drive This adds extra functionality to the ActiveX control by allowing text search It also has the ability to determine links at runtime, known as associative linking An associative link contains a list of keywords that the Help System uses to search for topics referring to the keywords This allows topics of a similar nature to be accessed from a pop-up window presented to the user
- **Layout Engine** The layout engine can be any Web browser that supports ActiveX The engine is hosted by the HTML Help Window and started through HTML Help API or indirectly through a HTML Help executable
- **HTML Help Window** This displays HTML in a customised resizable window, independent of the user's browser (similar to WinHelp) The window that appears is owned by the application that created it and when the application is minimised, so is the Help window Users may have many Help windows open at one time, unlike the solitary WinHelp window
- **HTML Help Workshop** The workshop is an authoring kit containing all the tools necessary for creating and maintaining an HTML Help project It provides online help and many wizards which automate the process of decision making [Hear97]

1.5 Hypertext Models

In the last ten years a lot of work has gone into developing models for hypertext These models are useful as they help to distinguish true hypertext systems [Fris92] suggest three different types of hypertext models that are required

- Interchange formats defined by means of a formal model of hypertext are required to facilitate the interchange of information between hypertext systems
- A semantic model to co-ordinate intellectual tasks
- A model of hypertext browsing semantics to be used for learning or sequential information seeking tasks

[Fris92] cites the following three models as being of the type just mentioned, the Dexter model, the gIBIS model and the Trellis model Each of these are now briefly presented

1.5.1 The Dexter Model

The Dexter Hypertext Reference model evolved out of two small workshops It attempts to capture the main abstractions that exist in a range of hypertext systems Its stated goal is to provide a basis for both comparing hypermedia systems and improving interoperability [Gini95] The Dexter Model represents a hypermedia system with three layers

- The Within-Component Layer The bottom layer, the within-component layer represents the contents and structure within the system. Developers are given the freedom to define new component types This facilitates the construction of systems that just contain text, to ones that have text, audio, video, etc.
- Storage Layer This defines how the nodes and links of the hypertext are connected to form a network This is the core of the Dexter Model A database is composed of a hierarchy of components, which are either atoms (primitive components) or composites (composed of other components), connected by links Links can be directional by using a 'from' and a 'to' anchor Anchors are described by a combination of a unique global component identifier with a unique intracomponent anchor identifier [Morr94]
- The Runtime Layer This is the top layer of the Dexter model This captures the ability to access and manipulate the data structures Design decisions concerning the appearance of the hypertext system are made here Options on link anchors are also

presented here and include bold text, icons and the change of cursor shapes [Morr94]

Runtime Layer Presentation of the hypertext, user interaction, dynamics
Presentation Specification
Storage Layer A 'database' containing a network of nodes and links
Anchors
Within-Component Layer The content/structure inside the nodes

Figure 1.4 Layers of the Dexter model

1.5.2 The gIBIS Model

The gIBIS model (graphical Issue-Based Information System) was presented by [Fris92] as a hypertext model based on a semantic model that allows one to relate to issues, arguments and positions by means of a pre-defined set of components and semantic links It was devised by Conklin and Begeman to support the system design process The model allows several people to collaborate on a design project to discuss the design through a process of deliberation and argumentation [Dunn93] The central idea behind this is to help each design team member to understand the problem, focus on important issues and provide a trace of the arguments and discussions within the context of hypertext In gIBIS, issues, arguments and positions are represented as hypertext components and semantic relationships by links Arguments can 'support' or 'object to' positions, positions can 'respond to' issues and issues can 'be suggested by', 'expand on' or 'challenge' positions [Morr94]

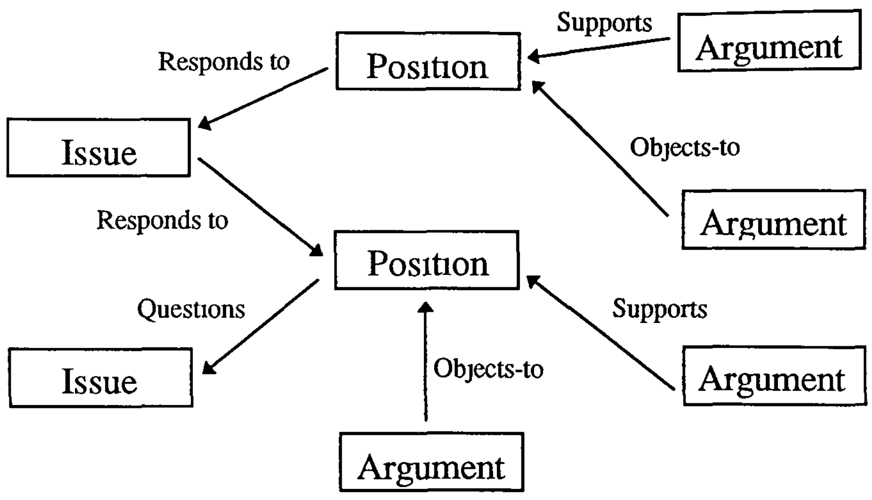


Figure 1.5 A gIBIS network [Dunn93]

1.5.3 The Trellis Model

The Trellis model is a hypertext model that incorporates browsing semantics The author controls and modifies the way a browser presents the information by representing the various states of browsing session and the conditions necessary to transfer from one state to another [Fura89], i e we can specify the manner in which the nodes are visited This model is based on Petri nets A directed, bipartite graph in which nodes are either "places" (represented by circles) or "transitions" (represented by rectangles), invented by Carl Adam Petri A Petri net is marked by placing "tokens" on places When all the places with arcs to a transition (its input places) have a token, the transition "fires", removing a token from each input place and adding a token to each place pointed to by the transition (its output places) [Howe97] In a Petri net graph the nodes are places When a place contains a token, a information element is displayed (node) Transitions represent links in the hypertext and the hypertext is explored by executing the Petri net

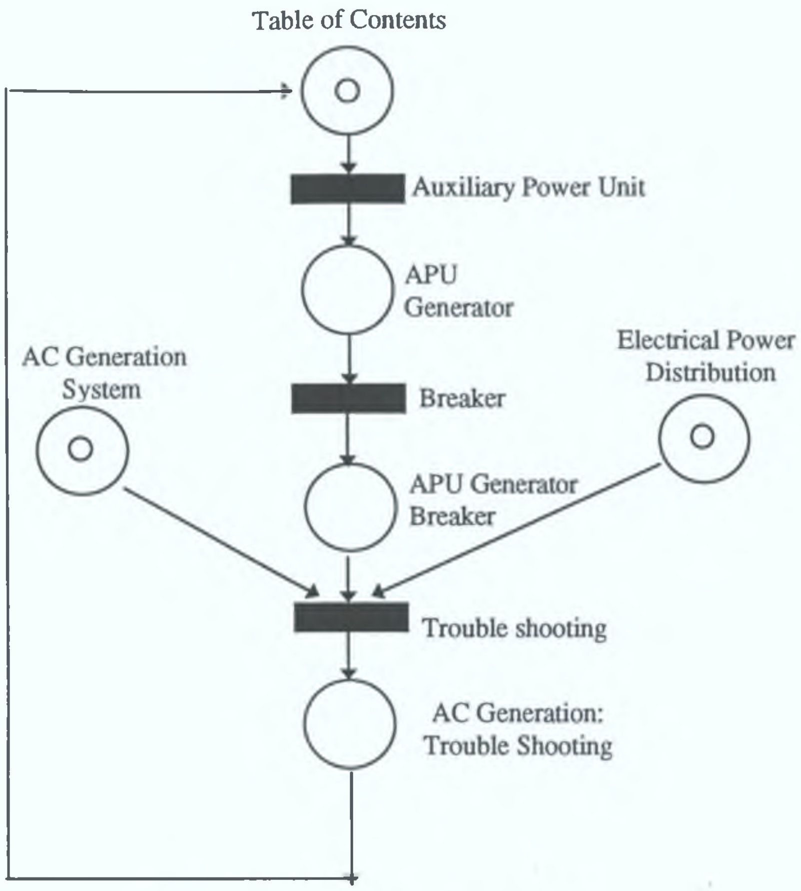


Figure 1.6 A Petri Net [Dunn93]

Figure 1.6 of a Petri net is an example of a hypertext describing the maintenance of the Electrical System of a Boeing 737. If the Auxiliary Power Unit link button is pressed the APU Generator node is displayed. The only way to get to the node 'AC Generation: Trouble shooting' is to traverse every node before it [Dunn93].

These are not the only models of hypertext available but are the most common. Of the three models described, the Dexter model is the most popular. This is because it leaves many of the design decisions to the author and is highly flexible. It also is the most popular choice among well known people in the hypertext community which has added to its credibility. However, these models have little to do with authoring and are more directed towards managing and storing information instead of the process of authoring. They assume that the hypertext already exist and are more concerned with the browsing aspect of hypertext instead of the authoring.

1.6 Summary

In this chapter we discussed the concept of information, the information revolution and structuring information into manageable building blocks This then gave an explanation of hypertext and the way it structures and presents information We discussed the theories, concepts and history of hypertext This led us to the history of the Web, its rapid growth, its structure and the way information is marked up on it, namely HTML This was followed by a look at hypertext systems Finally we looked at various hypertext models

Chapter 2 - Related Research

2.1 Introduction

In this chapter we will discuss related research that may be of benefit to the hypertext author This will take the form of discussing the various problems which affect hypertext structure, authoring and browsing Solutions to these problems will also be described This is followed by an in-depth look at 3 different authoring tools available for the construction of hypertext/hypermedia Section 2 4 will discuss possible methodologies that a hypertext author could use to enhance current authoring habits which are more geared towards presentation of hypertext nodes and not structural design of complete hypertexts This section 2 4 will cover the topics of link structures, hierarchies and global and local metrics

2.2 The Problems with Hypertext

In this section we will describe the current problems that are faced by people writing or browsing hypertext As most of these are problems that face hypertext users every day, they reflect the lack of authoring techniques used in the construction of hypertexts By creating a hypertext structure that recognises these problems, the author could enhance the browsing and learning capabilities of a hypertext reader The first problem we will look at is disorientation

2.2.1 Disorientation

The disorientation problem is also been referred to as being 'lost in hyperspace' As a user browses through a hypertext they can be faced with questions such as

- where am I ?
- how did I get here ?
- how will I get back to where I started ? [Smea93]

In traditional structured texts where there may be chapters, table of contents, page numbers and bookmarks, it is less easy to become lost, however in a complex hypertext network, a user may easily become disorientated It is very important that the hypertext author in constructing a hypertext deals with these issues or at least makes sure that the reader has access to some facilities that will ease these problems. [Conk87] defines the disorientation problem as the tendency to lose one's sense of location and direction in a non-linear document In a browsing session, hypertext users can come across links to 1000's of nodes and numerous anchors embedded into each of these nodes The World Wide Web has not helped this problem. Right now there is little or no structure to the Web and this problem continues to grow as more and more unstructured mini webs of hypertext are added Authors should reduce the fragmentation characteristics of hypertext [Thur96] sees these characteristics as being endemic to hyper-documents and result from segmentation into disjointed nodes

The problem of disorientation on the WWW has been tackled recently by a number of visualising tools to show authors and readers the structure of the hypertext they are using, in a graphical form. A sweep of the Web returns visualising hypertext tools like

- Atlas of Cyberspace - http //www geog ucl ac uk/casa/martin/atlas/atlas html
- Cybertools - http //www.cd dartmouth edu/~langmead/CyberTools/
- Dynamic Diagrams - http //www dynamicdiagrams com/
- Hyperspace - http //www cs bham ac uk/~amw/hyperspace/
- Interactive Graph Drawing - http //www cs rpi edu/projects/pb/graphdraw/
- Netscope - http //www merzcom com/demos/netscape/demo html
- Sitemap - http //lislin gws uky edu/Sitemap/Sitemap html
- Webview - http //www cs washington edu/homes/glinden/WebView/WebView html

These are to name but a few Figure 2 1 shows an example of the Dynamic Diagrams visualising tool to show the reader a map of the local web site using a Java Applet This map changes as users enter different subsections of the Web site

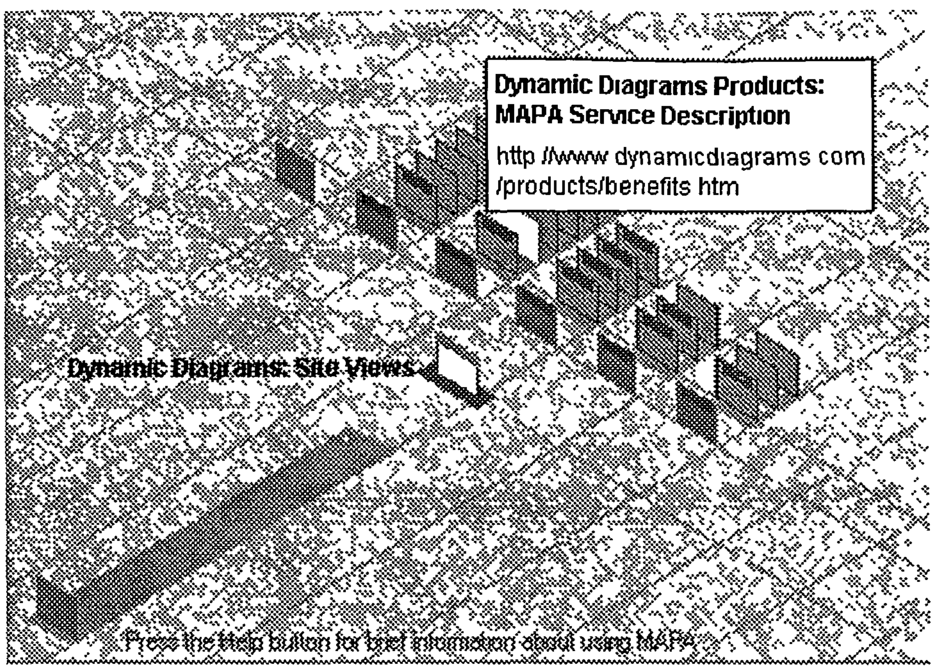

Dynamic Diagrams Products:
MAPA Service Description
http //www dynamicdiagrams com /products/benefits htm
Dynamic Diagrams: Site Views
Press the Help button for brief information about using MAPA

Figure 2.1 Dynamic Diagram Products - MAPA Service Description

There have been other solutions to the disorientation problem. The following is a short description of each

- **Guided Tours:** A guided tour is a presentation of a series of nodes associated with a particular subject or topic [Dunn93] The browser suggests to the user what is the best route to take through a selection of nodes on a particular topic At any stage the user may change their path and deviate to view another node by following a hypertext link If the user gets lost then they may easily return to the tour they were viewing

- **Bookmarks:** These allow a user to mark a place that they were interested in They are then free to browse elsewhere, knowing they can return to this node at anytime The ability to bookmark a popular site or individual node is one of the built-in features of most WWW Web Browsers

- **Thumbtabs:** The same as a bookmark except it may be viewed by all users in a multi-user system

- **Annotates:** The same as margin notes The reader is allowed make personalised notes about what they are reading, perhaps iconised

- **Breadcrumbs:** These show the user that they have already visited this node during this browsing session Also known as coffee stains Web browsers do this by changing the colour of an anchor to a node after that node has been visited Breadcrumbs help in two ways, (1) the user can avoid time wasting by realising they have been there before, and (2) remind the author of what direction they may have gone in before, where they came from and in what direction they may be heading in [Morr94]

- **History List:** The presentation to the user of a list of the most recent nodes they have visited in browsing the hypertext This allows the user to return very quickly to a node previously visited An example of this is the 'Go' button on the WWW Browser, Netscape Navigator Gold

- **Tabletops:** Tabletops are a means of capturing the layout of a set of nodes and turning it into a clickable icon This allows the user to browse through the hypertext knowing they can use the icon to return to a certain place in the hypertext

- **Fish Eye Lenses:** This concept is presented in a lot of the visualisation work done on improving hypertext disorientation These are based on the idea of the fisheye camera lens which distorts the apparent distances between objects in the picture Objects that are close at hand appear in detail, while objects that are further away are far off in the back of the picture Fisheye views were first purposed by Furnas [Furn86] as a means of overcoming the problems of general graphical browsers In terms of hypertext it gives the reader a disproportionate emphasis on the current node being displayed, showing the more remote nodes in lesser detail As the reader moves through the hypertext, the view presented by the lens changes in line with the current node being viewed. This however could lead to further disorientation by presenting to the user too much information about the hypertext. The number of nodes shown must obviously be limited [Dunn93].

- **Map:** A Map refers to the tourist metaphor for hypertext navigation [Dunn93]. A local map would show the nodes in the immediate locality of the node being viewed, whereas a global map would show all the nodes in the hypertext.

- **Flyovers:** In SemNet, [Fair88] uses a tool to present a three dimensional graphical view of the hypertext. This allows the user to fly over and among the nodes in the hypertext.

- **Graphical Browsers:** The idea of a graphical browser was around long before the appearance of any form of web browser. These use graphical images to allow the user to navigate through the hypertext. This idea is now being tackled by the recent spate of visualisation tools on the Web (discussed earlier in this chapter).

- **Backgrounds:** This is the idea of changing the background colour of certain nodes in a hypertext depending on their subject matter or their particular positioning in the hypertext structure e.g. different colours to represent different chapters.

2.2.2 Authoring Costs

The problem at the moment is that the cost of authoring hypertext is not properly considered and the consequences of this is a poorly authored hypertext structure [Smea93]. The problems of cost arise because not enough research has been carried out on how to start authoring and achieve good functionality. When it is decided that the hypertext is badly structured, then the costs of rewriting and righting the problem are immense. This is one of the main arguments between WWW developers and the rest of the hypertext community. The WWW group are more concerned with developing the Web as an enabling technology for other applications and to support other features such as advertising and other interests, than to be concerned about the structure of the Web. This is where the hypertext community should help in adapting their considerable knowledge towards applying a structure to the Web The money is certainly in the Web to do this and pay for the research needed

2.2.3 Jumping into Hyperspace

The 'jumping into hyperspace' metaphor comes from the need to supply the user of the hypertext with a good starting node to browse from. There are two ways to find a specific topic in a hypertext The first is to browse through the nodes from the start until the user finds what information they needed, while the second is to use some form of search to find particular subject matter Many hypertext systems provide boolean or string matching searches against a user query. A good starting point is a node that is connected to other nodes that are of interest to the user The 'goodness' of a node should not be measured by the similarity of a node to a query, but instead measured by the goodness of area around that particular node [Smea93] Solutions to this problem include search facilities based on vector space modelling [Coom90], statistical approaches in information retrieval [Crou89], probabilistic retrieval models using Bayesian Inference Networks [Crof89] and drawing graph patterns [Cons89] Other work in this are is the use of guided tours [Guin92], which dynamically generate a guided tour in response to a user query

2.2.4 Cognitive Overload

Cognitive overload is the feeling of being unable to retain all the information that has been presented to a user This feeling can be felt by both author and reader of a hypertext Due to the complexity of some hypertext systems, the user may feel bombarded with information, little of which is being remembered For an author, ideas may spring to mind when constructing a hypertext, ideas which may be lost if not written down in the form of notes or written into the hypertext immediately A reader of a hypertext not only has to comprehend the information in the node they are reading, but also contend with multiple anchors which link off to other pages Conklin describes it as 'the additional effort and concentration necessary to maintain several tasks or trails at one time' This problem arises because of the limited capacity of human information processing, with every effort additional to reading reducing the mental resources available for comprehension [Thur95]

There are two specific manifestations of cognitive overload

- **Embedded Digression:** where a user has to remember a list of side tracks of information that they would like to explore at some point, but not right now
- **The Art Museum Problem:** coherence and understanding are diminished as the reader takes in none of the information on offer as there is too much of it The name comes from the metaphor of visiting an art museum and seeing so many great works, but being unable to remember any in great detail

[Foss89] describes some tools which have been used in NoteCards to tackle the problem of cognitive overload

- **Graphical History Lists:** These lists record every node the user has visited during a browsing session They also provide a trace of multiple digressions allowing the user to revisit an area they have already been The information here is stored linearly
- **History Trees:** These trees also record a user's path through a hypertext, storing the information hierarchically A tree like structure is built up representing all the nodes and the connections between them that the user has visited Users are allowed to attach annotations to the nodes of the tree in order to remind them of some idea or detail that came to mind while visiting that node
- **Summary Boxes:** These allow an author to add annotates to nodes that they are creating to record ideas At a later stage they can return and view these
- **Summary Trees:** These are similar to a history tree with the added mechanism of allowing the user to add annotates in the form of text or drawing to any page that they may have visited on their travels through the hypertext This allows the user to build up a conceptual map of the hypertext

[Conk87] suggest the following solutions to the problem of cognitive overload

- **Have the reference node appear immediately:** By returning a link in as short a time as possible the extra cognitive load is minimised However, due to the traffic on the WWW and processor power this may be a difficult task to achieve

- **Provide short summary of node content:** Used ın Hyperties, thıs dısplays a small summary of the content of a node when a lınk to that node ıs selected. The user may then follow up on thıs lınk or decıde from the summary that ıt ıs not worth vısıtıng that node
- **Using a graphical browser which shows the local sub-network into which the link leads:** Thıs dısplays ınformatıon about the destınatıon node as well as ınformatıon about the surroundıng nodes From thıs the user can decıde whether a node ıs worth pursuıng

2.2.5 Integration

Before the Web all hypertext systems were closed Thıs means that they dıd not ıntegrate wıth other software on a computer and there was a fixed set of encapsulated applıcatıons A open hypertext system ıs one whıch allows any applıcatıon to partıcıpate ın hypermedıa envıronment The Web ıs an open hypertext/hypermedıa system. Its browser's can launch many dıfferent applıcatıons on a clıent's computer desktop However ıt ıs not fully open as ıts hypertext lınks are embedded ın the data whıch ıs stored ın a proprıetary document format Another open hypertext system ıs Mıcrocosm wıll be mentıoned later ın thıs chapter

2.2.6 Standards

Wıth the huge ıncrease ın Web usage, HTML (hypertext mark-up language) has become the new standard for wrıtıng hypertext Before the Web many of the PC based and Wındows products avaılable where not compatıble As of now all Mıcrosoft help files are beıng wrıtten ın HTML ınstead of the standard RTF (Rıch Text Format) The pre-Web Hypertext '89 conference produced a survey whıch reported that the lack of standards ınhıbıts the use of hypertext It was belıeved then that there was a need for standards ın the followıng hypertext rhetorıc, ınformatıon ınterchange, browsıng semantıcs, ınteroperabılıty, multımedıa data and synchronısatıon. Although HTML has ıntroduced some standards, the ımpact and sıze of the Web on the whole world has dimimshed some of these At present (July 1997), W3C (World Wide Web Consortium) recommends HTML Version 3 2

The W3C was set up in 1994 to develop common protocols for the evolution of the WWW W3C started at CERN and now are represented by consortia in America, Europe and Asia W3C are vendor neutral and offer a number of public services, (1) a repository of information and specifications for WWW users, (2) a reference code to implement standards, (3) various protocol and applications to demonstrate the use of new technology HTML 3 2 was developed in early 1996 together with vendors including IBM, Microsoft, Netscape Communications Corporation, Novell, SoftQuad, Spyglass, and Sun Microsystems It adds widely deployed features such as tables, applets and text flow around images, while providing full backwards compatibility with the existing standard HTML 2 0 HTML 3 2 is an SGML application conforming to International Standard ISO 8879 – Standard Generalised Markup Language As an SGML application, the syntax of conforming HTML 3 2 documents is defined by the combination of the SGML declaration and the document type definition (DTD) This specification defines the intended interpretation of HTML 3 2 elements, and places further constraints on the permitted syntax which are otherwise inexpressible in the DTD [W3CR97] The problem with the HTML 3 2 Standard is that not all browsers are capable of handling these standards, so people don't get to see a lot of the presentations provided by certain Web pages on offer This is particularly true about Java Applets As platforms differ, browsing capabilities differ Although there are versions of Web browsers for every platform, a lot of people don't have the hardware necessary to run these browsers Recently there has been a spate of browsing extensions added to many pages e g Real Audio, Macromedia Shockwave, Real Video, Microsoft ActiveX, etc These all add to the presentation of Web pages on offer but also diminish a Web page as few people bother to download the software necessary to run these add-ons or just aren't able to run the software from their platforms On top of this, download time is increased considerably, slowing up access even more

2.2.7 Text to Hypertext Conversion

In an ideal world, all hypertexts would be designed and written from scratch so that the layout and structure does not hinder an author's work A simple process that would convert text documents to hypertext nodes would considerably reduce the amount of time an author would have to spend on creating the hypertext structure There would be no need to learn another hypertext language if a decent tool existed that could do this conversion This would give an author more time to deal with structure and presentation with existing nodes becoming immediately available The following are a list of possible problems [Drak94]

- Word processing formats contain information that is just not convertible into HTML e g font types
- It is difficult to determine the granularity for each node, how much information makes up a page of hypertext, too little or too much may equally disorientate a reader
- Some documents may require navigation aids due to the complexity of the structure of the text, or possibly the complex nature of the subject
- Conversion may be inflexible with the converted documents not fully availing of a hypertext's capabilities An example of this may be that in a text document, a referral to another section takes the form of 'See Section 2', whereas this would just be a link in a hyper-document
- Another problem may be that a document may be too linear for a hypertext Sometimes breaking a document down to suit a hypertext may lead to the loss in detail of the subject matter

For large hypertexts this problem is an impossible manual problem as the time taken to create a proper hypertext from text would be huge Our authoring tool described in this thesis, uses similarity measures to create a cross referencing structure which is one possible way of countering some of the problems of improving text to hypertext conversion Of course there is still the problem of embedding anchors into the text as well

2.2.8 To Embed or not to Embed

When creatmg a hypertext node not only does the author have to deal wıth the problem of decıdmg what nodes should be lınked to each other and what structure to keep, but they must also face the problem of whether to embed the anchor for each lınk ın the text or to keep them separate as 'See Also' lınks Fırstly however ın talkıng about lınks between nodes we must poınt out the 3 dıfferent lınk types These are

- **Reference link** - these are used to relate part or whole of a node to another node(s) These are usually unı-dırectıonal and are generally used to create cross-referencıng structure commonly found ın hypertext systems
- **Relational link** - those that have a strong semantıc closeness, usually bı-dırectıonal They relate to nodes that address sımılar subject matters
- **Structural link** - those that express a hıerarchıcal relatıonshıp, formıng a tree ın the hypertext structure These would be dıfficult to generate and would be more of a concern when topology and the structural concerns are brought ınto a hypertext

If anchors for lınks are embedded ın the text of a node, then the author has a number of problems to overcome One of these problems ıs plurals or tenses The same term can be represented ın a number of dıfferent forms For example, computer, computerıse, computabılıty, etc To reduce these words to a common form, a stemmıng algorıthm would be needed as ıs done ın conventıonal ınformatıon retrıeval (See Chapter 4 Porters Algorıthm)

Another common problem wıth embeddıng anchors ıs the use of abbrevıatıons An example of thıs would be HTTP or Hypertext Transfer Protocol Here the author has to recognıse the fact that both terms mean the same A predefined lıst of terms and theır relevant abbrevıatıons may be the only way to solve thıs problem, otherwıse manual anchorıng throughout the nodes wıll have to be done There may also be a problem wıth words that are hyphenated e g. tuples, n-tuples

In choosıng a possible posıtıon for an anchor, there may be more then one occurrence of the term ın a node Here the author must decıde whether to make all of these occurrences anchors which is not particularly wise as too many anchors may confuse the author, or whether to choose one term to be the anchor

[Dunn94] suggests the following 9 steps to embedding anchors in text

- Find terms that are an exact match
- Remove '-' from node content
- Remove stopwords
- Apply normalisation of terms and reduce all plurals to singular form
- Remove any phrase which appears in more than one or more anchors
- Accept a string in the node text as a suitable location if it contains all but one word from the search string
- Select local stopwords based on frequency of occurrence within node titles and their anchors
- Follow step above but reduce the selected frequency threshold by 1
- Select global stopwords

In our project we decided to keep the anchors in the nodes themselves but external to the text The reason for this is given in Chapter 4 Section 4 8 Another way to embed anchors however is to keep the links external from the text Microcosm, developed at Southampton University, is a system that embeds the anchor in the text but stores the link externally A link database is used to store the information about what nodes are connected by what links, but the application is responsible for maintaining the persistent sections which mark as hotspots With the link structures held locally, tools to show link structures, dangling links and tools for navigation can be built [Davi96] This could also means that moving and deleting links becomes less of a problem

2.3 Hypertext Authoring Tools

At present there is a massive range of hypertext authoring tools on the market or freeware ones available to download The list of Web-based HTML authoring tools is immense. However most of these deal with the construction and presentation of the text in HTML format instead of the actual cross referencing and structure building of a hypertext web. The authoring tools are closer to being individual node editors than hypertext web constructors. In the next section we will provide an overview of three authoring tools. Two of these are web based, Netscape Gold and Microsoft FrontPage, while the other, Microcosm, is an open hypermedia system that provides an authoring tool and although it is separate from the Web it can still interface with it. We choose these three at random as they are representative of the state of the art of hypertext authoring tools.

2.3.1 Netscape Navigator Gold 3.0

Described by [Nets97] as ‘a powerful solution for information creation, access and sharing with the seamless integration of Netscape's Web, News, and Mail applications’ Netscape Navigator Gold 3.0 is a WYSIWYG (What You See Is What You Get) editor that allows easy mark-up of HTML nodes. The editor is built into the Netscape Web Browser and can be accessed by the click of a button from the Netscape Browser (Figure 2.2 displays the editor toolbox). This means that it is cross platform.

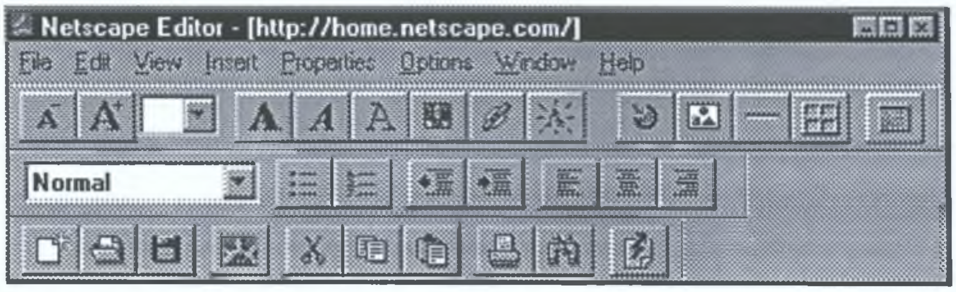

Netscape Editor - [http://home.netscape.com/]
File Edit View Insert Properties Options Window Help
Normal

Figure 2.2 Gold’s Editor Toolbox

The author, in creating new nodes in Gold is offered 3 choices, start from a blank document, use a template (in order to follow existing examples) or use the wizard available to take the author on a step-by-step construction of an HTML page. It provides all the features of HTML (e.g. style guides, etc.) and also allows the user to download existing pages and re-edit these. However, although it allows linking, the author has to choose the anchor for this link manually. There is no in-built link suggestion process and no attention paid to the overall structure of the web being constructed.

2.3.2 Microsoft FrontPage

MS FrontPage is more of a mini web editor then a node editor Like Gold it contains a WYSIWYG editor avoiding the need to know HTML Some of the functions of FrontPage are as follows

- **Create FrontPage Webs:** This allows the use of templates and wizards to create Web pages including a corporate presence wizard, discussion web wizard, personal web template and project template
- **Administration:** There are 3 types of permissions that an administrator may add, browsing, authoring and administering permissions All permissions are hierarchical ranking from administrative to authoring and then browsing
- **View the Entire Web:** A user can view the web created graphically, viewing all nodes and links between them. This view may be expanded or contracted to facilitate user specification (See Figure 2 3)

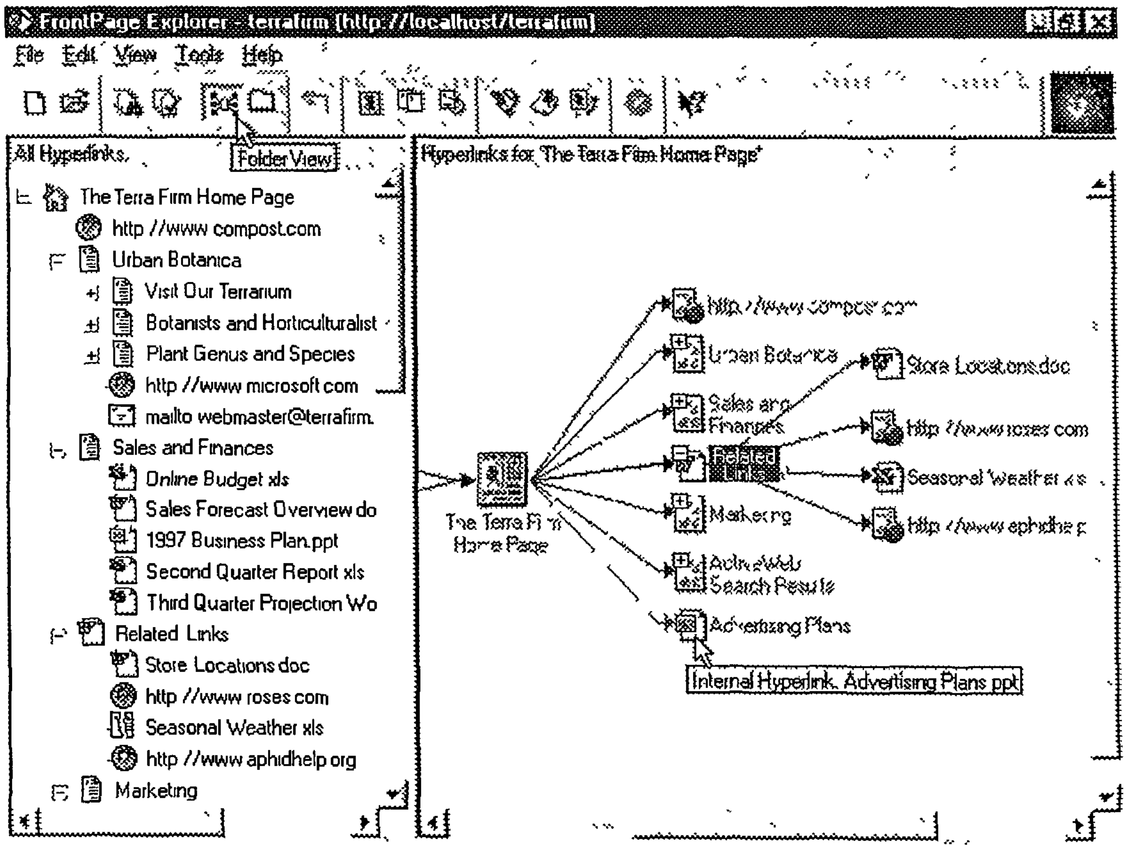


Figure 2.3 Visual of FrontPage Web

- **Test and Repair Hyperlinks:** If any node is removed or renamed, all links are updated in the entire Web There is a 'Verify Hyperlinks' command which checks that no links are dangling This can be done on internal and external links A Verify Hyperlink dialog box allows a user to repair any broken links Another function of FrontPage is a 'Recalculate Hyperlinks' command which will show any recent changes in the Web if another author has been working on it concurrently
- **Configuring Editors:** This allows you to configure editors for any of the file types in a Web
- **To Do List:** This is a sort of annotation facility that tracks and names all unfinished work tasks in a web construction
- **Full Text Index:** A text index is kept of the web being created for use with the WebBot search component This allows users to find nodes in a Web without the need to program a search facility WebBot allows a user to insert dynamic objects into a web This provides complex functionality including search FrontPage also supports Java, ActiveX, database connectivity and Visual Basic

2.3.3 Microcosm

Microcosm provides an open and resource based approach to the development and delivery of learning materials [Davi97] Microcosm is a hypertext system, but here however we will describe the authoring process that it supports Authoring using Microcosm involves two different processes The first is the building of a resource base, the second is the act of authoring within that resource base There are 4 steps needed to build a resource database These are as follows

- **Collection:** This is the process of collecting data to use in the hypertext This means text files, files of other formats, paper documents, pictures, bmp's, gif's, etc Microcosm suggest adopting the approach of rejecting nothing - it is important that all information relevant to the particular subject area is gathered [Hall97]
- **Digitising of Information:** This involves the process of turning paper documents into electronic documents and also the scanning of texts and pictures

- **Organising:** Information is organised into the structure needed using Microcosm's Document Management System e g list documents according to author, themes, title, physical shape, etc
- **Indexing and Generic Linking:** In order for nodes to be found, generic linking is necessary This defines how to get to a node but not where to go from it A generic link database is created that is sorted alphabetically and presented in node format This can act as a guide to the author as to what links have been created and also serve as a powerful tool for the user Different links are kept in different linkbases to avoid confusion, depending on their subject matter These links provide no specific connection between source and destination node Instead these destination nodes, either labelled or indexed, are a set of external information resources that bind, on the fly to specific occurrences of source node selections These bindings are an expression of authored links [Hall96]

The next phase is authoring within the resource base When the resource base has reached a certain size then the authoring process can begin [Hall96] describe this as building a coherent body of information with a particular purpose The process of collecting information is still continued in parallel with the authoring process One of the key features of Microcosm is its ability to incorporate information using third party software like Authorware, Toolbook and Director Effective authoring should give pointers to important parts of information, answering the questions of

- what am I doing here ?
- what are the objectives and aims ?
- how am I going to go about achieving this ?

[Hall96] suggest the best 4 applications of Microcosm are

- educating and training
- geographic and urban information systems
- technical documentation and manufacturing support
- multimedia archives

Microcosm's authoring environment provides hypertext linking, logical structuring of material, direct access to any material, search facilities, glossary keywords and hidden links (these enable the user to focus on the material while only activating cross references on essential materials) There are different sets of links to the same resources, allowing the information to be reused by different users who have different requirements and interests Another function of Microcosm is the dynamic link generator using Prolog Links can be generated or accessed in response to dynamic processes such as a database or rule based query This allows clause evaluation resulting in the return of different interpretations depending on the query

2.4 Methodologies For Structural Analysis

In this section we will explore how other methods may effect hypertext structural analysis in an authoring environment This study will be mainly based on work in this field by Botafogo, Rivlin, and Shneiderman [Bota92] who analysed hypertext in two ways, the first was to provide authors with different views of the hypertext, the second, examined useful metrics to reflect the properties of nodes and the whole structure First however we will discuss the application of developing link structures in the creation of a hypertext and the way these could effect the authoring process

2.4.1 Link Structures

A common problem with hypertext may be that in static hypertexts there is an abundance of links, many of which are inappropriate at times [Calv97] We will take a brief look at the work of three different approaches to this problem, to see whether this research could aid the hypertext author in the creation of nodes and the cross referencing that is involved in the authoring process

[Chua95] developed a hypertext training system (HTS) which enabled a user to browse through a hypertext, provide judgement on node traversal and activate the system to update link structure Although HTS is not an authoring tool, we will give a brief explanation of it as it deals with link creation between nodes The HTS consists of 5 major modules The first is the link generator which builds an NxN Similarity Matrix between N hypertext nodes in a web, to generate an average of k links per node This uses the same weighting scheme as that used in our authoring tool's node to node comparison, namely tf*IDF (See Chapter 4 Section 4 6) The Start Node Advisor module computes a ranked list of possible start nodes and the Next Node Advisor builds a ranked list of next nodes for the user to select at each stage of browsing This sequence of the nodes browsed and any possible feedback which the user gives on each node traversed are stored and used by the link trainer to update the weights in the SIM matrix Finally the Query Processor provides a query-based search function that modifies the query based on user feedback

[Chua95] suggest 3 ways of evaluating the usability of a hypertext to support effective user browsing and information retrieval These 3 methods may also be of use to an author in evaluating the hypertext they are creating The first is to compute the average cost of locating the first relevant node, F, and subsequently retrieving the rest of the relevant nodes

$$\text{Cost} = F + S = F_i + \sum_{j \in R_i} path_{ij}$$

where F_i is the cost of locating the first relevant node i in the relevant node set, $path_{ij}$ is the path length from node i to node j, R_i denotes the rest of the relevant set excluding node i This measure will show the number of links needed to be traversed by a reader to retrieve all relevant nodes in a hypertext The author may find that there maybe too many links to follow to find all the information on a particular topic and may choose to change this The second suggested measure is compactness which we will discuss later Their third measure is recall, which measures the fraction of relevant nodes in a collection that are retrieved

$$\text{Recall} = \frac{\text{Total number of nodes retrieved}}{\text{Total number of relevant nodes in the hypertext}}$$

Although this is more of a measure of the effectiveness of a retrieval engine, it may also provide the author with useful knowledge as to how relevant the subject matter of their hypertext is against a user query.

[Adam97] used another method known as object decomposition to structure an existing hypertext. This method provides a consistent framework for breaking course material into manageable and interrelated nodes. This method was tried on a CS383 hypermedia course 'that attempted to enhance the educational resources available to a student outside of the classroom' [Adam97]. The initial hypertext had a loose structure around course objectives and schedule. This structure was then re-created into a object oriented text decomposition based hypertext (OOTDH) [Talb89]. This method uses object decomposition to develop a hierarchy of relationships between concepts in the material. As a main concept is decomposed, the explicit relationships serve as links between the conceptual components. This could force an author to rethink several key development issues:

- How do I expect students to access this concept ?
- What level of detail is needed at this level of abstraction ?
- Can the student access material which supports or expands on this concept directly, or do they have to navigate through different hierarchical levels ? [Adam97].

[Adam94] developed graphs and flow charts to assist a hypertext author in dealing with these issues. The graphical display of cross references and hierarchical links meant that a complete system map was developed. To evaluate how big the change in structure was from the original hypertext, a Relative Out Centrality (ROC) was used [Bota92] to measures a node's accessibility. The difference between the new structure imposed on the hypertext as against the old structure showed that node access is made almost 5 times easier by using Object Decomposition. This method was also found to enhance student performance in exams but also increase the amount of time required to visit the number of increased nodes.

[Calv97] describes research into creating dynamic hypertext structure and content to facilitate the learning process of a student. They point out that variations in link structure are needed to advance learning material. They suggest that although metrics like those applied by [Bota92] are 'able to find link structures that are unusable they cannot guarantee that link structures having all suggested values for different metrics will actually belong to highly usable hyper-documents' [Calv97]. Links are often introduced without consideration of their actual usefulness. Users often find themselves confused as to why a piece of information is at the end of a particular link meaning that linking can hamper instead of benefiting the user [Calv97] developed a technique which creates link structures based on the following requirements

- **Required Knowledge:** knowledge needed in order to view a node The system automatically hides (conditional) links to nodes if the user has not acquired this information by having ‘visited’ prescribed nodes in advance
- **Forbidden Knowledge:** knowledge that makes viewing uninteresting The system automatically hides (conditional) links to nodes if the user has acquired this information by having ‘visited’ prescribed nodes in advance
- **Generated Knowledge:** knowledge the user gains after reading this node

This technique was found to enhance the student learning process while still preserving as much navigational freedom as desired However [Calv97] is quick to point out that dynamic link structures must be used with care as they could confuse a student due to unexpected changes

2.4.2 Hierarchies

Centrality is the notion of how easily a node can access other nodes To do this we must find the sum of distances between one node and all other nodes A matrix that has as its entries the distances of every node to every other node is called a distance matrix If a node can't reach another node then the distance is infinite To find out which node is most central we use a converted distance matrix C defined as follows

$$C_{ij} = \begin{cases} M_{ij}, & \text{if } M_{ij} \neq \infty \\ K, & \text{otherwise} \end{cases}$$

where M is the distance matrix, K is a finite conversion constant (usually set to the number of nodes in the matrix) The matrix is made up of the central out distance (COD) and converted in distance (CID) COD is the sum of all entries in row I for node i in the matrix,

$$\text{COD}_i = \sum_j C_{ij}$$

The CIN for a node i is the sum of all entries in column i in the converted distance matrix,

$$\mathrm{CID}_i = \sum_j C_{ji}$$

The converted distance (CD) of a hypertext is given as :

$$\mathrm{CD} = \sum_i \sum_j C_{ij}$$

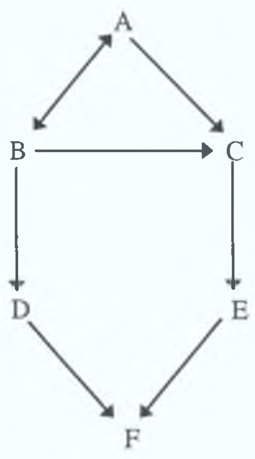
In order to define the central node for a hypertext we must find out which one has the smallest distance to all other nodes. As the distance grows the node becomes less central. [Bota92] states that for a small hypertext, COD gives a good measure of node centrality compared with other nodes but indicates little if two hypertexts are compared due to the possible differences in sizes. Two other formulae are used to compensate for this problem. First the ROC (Relative Out Centrality) metric for a node is defined as:

$$\mathrm{ROC}_i = \mathrm{CD}/\mathrm{COD}_i$$

The higher this measure the more central the node, the more it can access other nodes. The second metric is the RIC (Relative in Centrality) which is defined as:

$$\mathrm{RIC}_i = \mathrm{CD}/\mathrm{CID}_i$$

The higher the value for this the more accessible the node (see Figure 2.4). This would provide an author with the ability to see how well the node is linked to every other node and whether a user may get disorientated by the number of possible nodes and links between them.



	A	B	C	D	E	F	COD	ROC
A	0	1	1	2	2	3	9	10.2
B	1	0	1	1	2	2	7	13.1
C	6	6	0	6	1	2	21	4.3
D	6	6	6	0	6	1	25	3.7
E	6	6	6	6	0	1	25	3.7
F	6	6	6	6	6	0	30	3.1
CID	25	25	20	21	17	9	92	
RIC	3.7	3.7	4.6	4.4	5.4	10.2		

Figure 2.4 Converted Distance Matrix and Associated Metrics (taken from [Bota92]).

In constructing a hypertext, authoring tools sometime differ on the flexibility of the structure the author may create NoteCards makes an author create links hierarchically, Hyperties gives authors the freedom to create a structure of their choice But which is better ? [Bota92] suggests ways of finding hierarchies that will separate hierarchical from cross referential links [Trig87] suggests that a hypertext system should allow the author to create many coexisting organisations of the same hypertext which can evolve in parallel

To identify a hierarchy, the root of the hypertext must be found i e that node that reaches all of or most of the other nodes in the hypertext It should also have a reasonable number of children and its distance from it to any other node should not be too large To identify a good root for a hypertext, all index nodes should be removed (nodes that present an index or guide to all other nodes) and an author should select nodes that have a high relative out centrality (ROC) Once the root has been identified one can separate the hierarchical from the cross reference links This is done by assuming that vertices are flexible joints and edges are strings of constant length By picking up the graph at the root and letting the rest dangle one will be left with a tree that has the shortest path taut

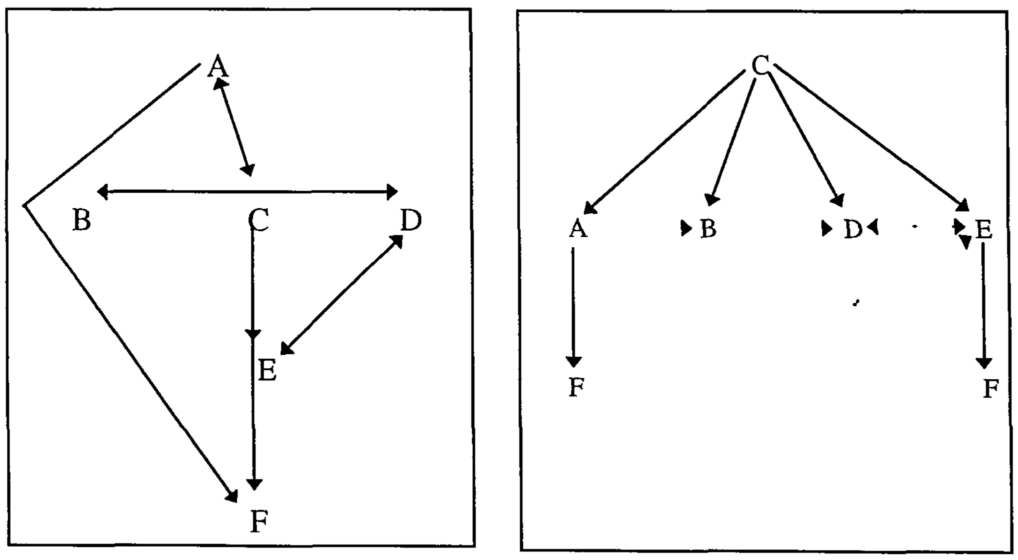


Figure 2.5 'Hierarchisation of a Graph' [Rivl94]

Figure 2.5 shows how this technique works. Hierarchical links are presented as solid lines, while cross reference links are in dashed lines. In this diagram, node F has 2 parents from the same level. Node F is part of subtree A or part of subtree E. Once a hierarchy has been found it is important to show it to the author in a reasonable way. One option is to display the whole graph. This may not be wise as the hypertext may have a large number of nodes and links. Other ways are to use maps and fisheye lenses which have been discussed earlier. Yet another way is to show the hypertext in the form of a book, with each hierarchy forming a different chapter. It may be noted that the use of hierarchies is equally an important tool in solving many problems felt by readers of hypertexts and not just authors [Rivl94].

2.4.3 Metrics

Metrics give clear and precise values to objects being studied so that subjectivity is not involved [Bota92]. Word, the word processing package from Microsoft for example uses metrics to show how readable a piece of text is by showing how much passive voice is used for example. Showing the readability of a node may be of some use to an author but in the overall scheme of things where a large hypertext is concerned it is not very useful. [Bota92] describes the use of two different types of metrics that will characterise the structure of hypertext. These are global metrics which deal with the hypertext as a whole, and node metrics, which focus on the structural properties of individual nodes. These will present to the author how complex and structured the hypertext they are building is.

There are two different global metrics, stratum and compactness. Compactness shows how well connected or disconnected a hypertext is. Values vary between 1 and 0, 1 indicating a high compactness value which means that each node can easily reach every other node in the hypertext. 0 or a low compactness value indicates that parts of the hypertext are disconnected. Since hypertexts vary in size (nodes and links), it is not easy for an author to grasp the significance of those numbers. [Bota92] tested these metrics on three different hypertexts to see if it was a good measure of complexity for a hypertext. CMSC (Computer Science Department of the University at Maryland) which is hierarchically structured with 106 nodes and 402 links has a compactness value of 0 53 HHO (Hypertext Hands-On!) is an electronic book with 243 nodes and 803 links, also built hierarchically has a compactness value of 0 55 Finally GOVA (Guide of Opportunity in Volunteer Archaeology) is structured like an encyclopaedia has 222 nodes and 1609 links and has a compactness value of 0 78 It can be seen that the differences in the structures and the differences in node and link ratio are reflected in the compactness values for each hypertext This shows that compactness could prove a useful aid in helping the hypertext author to create a better structure for their hypertext Compactness is the metric we will use in our topology maintenance module and is discussed in further detail in Chapter 4 Section 4 7

The other metric [Bota92] used was stratum, which was designed to capture the linear ordering of a hypertext Stratum is a measure that indicates whether there is a natural order for reading a hypertext Maximum stratum is achieved in a linear hypertext If stratum is zero the hypertext has no hierarchy [Rivl94] Again this metric was applied to the 3 hypertexts named above The values for CMSC, HHO and GOVA were 0 13, 0 05 and 0 01 respectively These values show that even simple hypertexts like CMSC are still far from linear and the author has used links liberally Although HHO has more nodes then GOVA its stratum is 5 times as large This is due however to the number of cross references used in GOVA

It should be pointed out that compactness and stratum are not independent measures If a hypertext as a compactness value of 1 its stratum will be 0 and vice versa However both metrics indicate different but possibly very useful properties for a hypertext author (See Figures 2 6a and 2 6b [Morr94]) They supply the author with information about the hypertext like its density, in linearity and its organisation The author may decide that the hypertext should be broken down further into different clusters of information or whether to bring some nodes closer together. What global metrics achieve is the ability to let an author impose some form of structure on the hypertext they are creating

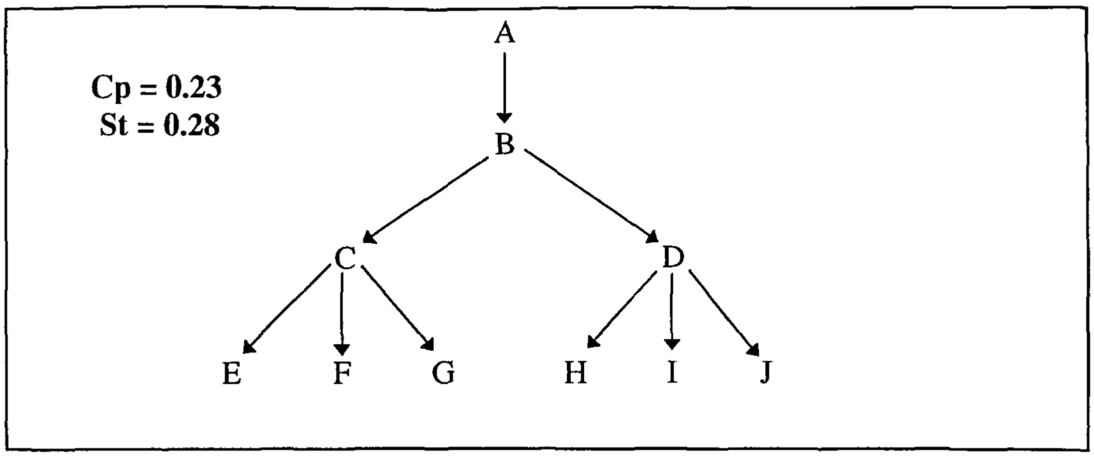


Figure 2.6 (a)

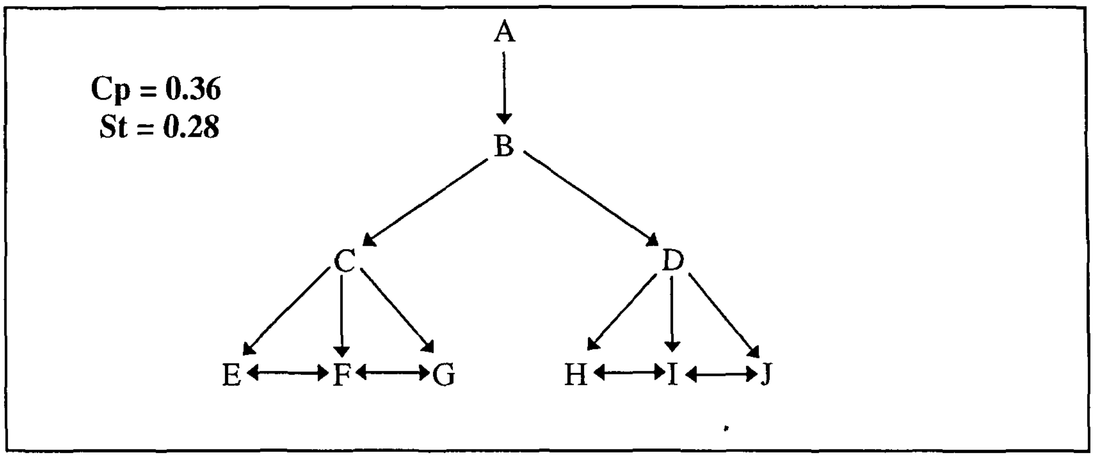


Figure 2.6 (b)

Node metrics are designed to indicate special nodes in the hypertext These may indicate to the author whether a node is difficult to reach because it is too deep in the hypertext, whether links are missing or if a node is specialised [Bota92] Botafogo at al suggest two different node metrics, depth and imbalance

Depth indicates the distance a node is form the root of a hypertext The larger the distance the harder it is to reach that node as more links have to be traversed Nodes that are deeply embedded in a hypertext are unlikely to be read by most hypertext users The author could detect this as a bug or they could purposely store this node so deep because it may contain information that may be of little consequence to the reader Only readers with a deep interest in a topic may come across it as they read further into the hypertext. This metric offers the author the chance to locate unreachable or deep nodes and verify that their placement was intentional [Bota92]

The second node metric is imbalance In a hypertext we would like each node to be about one idea and the text for idea to be similar in length i e number of words With this assumption and also the idea that each link from a node is an extension of an idea contained in the source node, the author may like to create a hypertext in the form of a balanced tree An author may use this metric to create a balanced hypertext or if imbalances are required they may overlook this information An imbalanced hypertext is not always bad and may in fact be a requirement of the author's For example our DCU Web server would purposely contain far more information about 'Faculties and Centres' then about each lecturer in the university This would cause an imbalanced tree with a bias towards information about 'Faculties and Centres'

The metrics and structural processes used to analyse the structure of a hypertext, show it is possible to indicate different views of a hypertext to an author They also identify possible errors in the structure [Bota92] suggests using them to see if one's hypertext contains the expected properties He also suggests improving results by taking into account node content and stylistic dimensions as well

2.5 Summary

In this chapter we covered the problems which a hypertext author may face in the creation of a well structured hypertext We then explained three different authoring tools available Although these provided excellent presentation facilities of hypertext nodes, they did not help particularly in adding structure and balance to a web of information To do this we found that research had been done in adding structure to nodes by using metrics and other techniques including link structure The underlying theme presented however, is that present authoring techniques available in commercial software do not possess the capabilities presented by tools and techniques developed in research This is why we see the large number of different hypertext problems Instead we find that these tools that have user interface techniques and textual analysis, do not make use of an important part of hypertexts their structure [Bota92]

Chapter 3 - Preparation of Data

3.1 Introduction

In building an efficient authoring tool, we needed to create a good test set to work on in order to get results and comment on our findings. This opened up many questions, such as finding an existing hypertext to test on, where this would come from, whether to design the authoring tool around hypertext on the Web (HTML) or to take some other hypertext system and whether to build a PC or a UNIX standalone version of the authoring tool. The first section of this chapter deals with why we choose the Web instead of another hypertext system. The next decides on UNIX as against Personal Computer. Section 4 discusses the Dictionary of Computing, the test hypertext we used. This is followed by a section covering the changes we had to make to this test set to shape it into the format we needed. Section 6 explores the problem definition, why there a need for a different approach to authoring hypertext. Finally section 7 provides a summary of the chapter.

3.2 Web versus other Hypertext Systems

Presently in the hypertext community there seems to be a backlash against the role of the Web in the construction and browsing of hypertext. Before the WWW appeared in 1989, hypertext was written for specific systems. Few would have heard of the words ‘hypertext’ or ‘hypermedia’, except people who resided in colleges or worked on these systems. It was not until the Apple’s HyperCard that hypertext really found a market for itself, however retrieval was slow and hypertext viewers were not very robust, specifically designed for one format and one format only. Then the Web took over and its growth since its beginning has been phenomenal. Not only has hypertext become a buzz word, it has also become another form of medium, with most companies presenting their business through pages on the Web. Its rise has taken the original hypertext community by surprise, leaving them with many questions and few answers. The community seems to have split into to different factions, those who are opposed to the Web’s influence on hypertext and believe that hypertext research has little to do with the Web and those who want to use the Web as the infrastructure to build hypertext on and embrace the advantages that it come with it [Smit97] in his keynote speech at the Eighth ACM Conference on Hypertext sees it as one group whom view the Web as an unwelcome guest, who have simplified the data model, ignored problems of large scale navigation and declared link integrity as irrelevant On the other hand there are those who want to embrace it and the considerable technology that comes along with it Smith feels that the hypertext community is in a unique position to give the Web some of the features that have made small experimental systems so practical Some of the advantages of smaller hypertexts include integrated authorship, global write access and reliable links Reliable links seems to be one of the major problems of the Web There is no governing body that checks to see if a link still exists or whether its address (URL) has changed Many would claim though that it would be impossible to ever have 100% reliable links as the Web is just too big

We are also of the opinion that the Web is the way to go for now as there are such vast numbers using the web, not just for browsing but also for authoring and publishing information over a wide range of topics including education, business and sport Hypertext is now not just for those in educational institutions or the Information Technology business, it has now a much wider spread and the Web must be congratulated for this event The Web provides global access, the ability to create a simple page of HTML with many links and images in minutes This is the major problem with any other form of existing hypertext system. Many of the existing systems require the user to read manuals, follow demos and look for technical support e g the Microsoft Help System, HyperCard, Hyperties and Microcosm (these are discussed in full in Chapter 2) When they have created a hypertext using one of these systems, it is usually an end in itself with no possibility to convert it to any other kind of hypertext mark up or integrate it HTML as a subset of SGML (Standard Generalised Markup Language) makes parsing and conversion easy, so converting one set of tags to another format, or the removal of these tags to get the text as a standalone should not be a problem

It could also be said however, that the ability to create nodes and links so easily has left the Web in the mess that many feel it has now become There is little or no structure to the Web as ıt stands, there ıs no one to regulate whıch nodes are lınked to what, no authorıty to check for danglıng lınks or enforce standards The Web ıs an open medıum for users to do wıth what they please Thıs ıs the attractıon and also the downfall People who browse the Web can only but acquıre skılls and knowledge from whatever ıs already there Perhaps ıf Web/hypertext authors maıntaıned better structures ın the begınnıng, Web browsers who try theır hand at authorıng may just pıck up some good authorıng habıts The defınıtıon of thıs part of the problem wıll be dıscussed ın more detaıl ın Sectıon 3 6

3.3 UNIX versus PC

Havıng decıded to select the Web as our medıum, we were then faced wıth the decısıon of whether to buıld the authorıng tool on eıther a PC or UNIX Workstatıon platform. The choıce was ınfluenced by the technology avaılable In the fırst ınstance, the authorıng tool was goıng to use HTML forms and CGI scrıpts whıch all must be run off a Web server, these could be facılıtated by our group's (Multımedıa and Informatıon Retrıeval Group) Web server whıch ıs run on a UNIX workstatıon Thıs was the technology avaılable at the tıme, ıt would have course have been possıble to use a Wındows NT server or any other platform Web server ıf they had of been avaılable The second reason for choosıng UNIX was that some other research work was already wrıtten on the UNIX platform and the tool could use thıs exıstıng data and executables from prevıous projects such as the DCU Web Robot whıch uses compactness metrıcs and Query Space Reductıon for Informatıon Retrıeval whıch contaıns a search engıne Although we have based the authorıng tool on a UNIX platform ıt wıll ın fact run on a multıple platform basıs, as ıt ıs Web based ı e UNIX, Wındows NT, DOS, etc

3.4 Dictionary of Computing

When decıdıng what hypertext to test the authorıng tool on, we looked for an exıstıng hypertext ınstead of usıng the tool to create one from scratch Thıs was done maınly because we wanted a hypertext that was ınformatıon-rıch and whıch had a wıde spread of connectivity throughout its nodes to provide an excellent test set for the authoring tool to work with. The Dictionary of Computing was ideal for this.

We had decided to use the Dictionary of Computing after viewing other possible existing hypertexts and deciding that it was not feasible to build our own hypertext from scratch. The latter decision was made due to the time period involved as well as subject matter. We needed a large scale hypertext, one that was information-rich with a good deal of connectivity. To build a hypertext incorporating these features would take to long, the scale of this task being to large. The decision to use the Dictionary of Computing over other existing hypertexts was made very carefully. We had first decided to also use a CD ROM distributed course on multimedia and a wide variety of topics known as the Multimedia Course. However as this was CD ROM based, we had no access to the actual text of the course. This meant to construct a Web based hypertext from this would have meant retyping the whole course as well as recreating the cross linking all of which would have to be manually marked up in HTML. Another hypertext we looked at was for a company who we had done previous research work for. They had wanted their private networks hypermedia course used in our research. After viewing their possible hypertext of information we found it to be too linear, very much written in the format of a book. Other hypertexts viewed included web courseware and encyclopaedias.

The Dictionary of Computing (Copyright Denis Howe 1993, 1996.), known as FOLDOC (Free on-line Dictionary of Computing), is a searchable dictionary of acronyms, jargon, programming languages, tools, architecture, operating systems, networking, theory, conventions, standards, mathematics, telecomms, electronics, institutions, companies, projects, products, history, in fact anything to do with computing. Some examples of the wide diversity of terms are

- *'Zipperhead - an IBM term for a person with a closed mind'* ,
- *'Propeller Head - jargon used by hackers that is equivalent to a computer geek'*
- *'Egosurfing - Scanning the World-Wide Web, databases, print media or research papers looking for the mention of your name'* [Howe97].

The Dictionary started in 1985 and now contains over 10,000 definitions totalling 3 7 megabytes of text. Entries are cross-referenced to each other and to related resources elsewhere on the net These include HTTP, FTP, Telnet and NEWS sites This Dictionary is not to be mixed up with the IEEE Computer Society's Computer Dictionary Project which also produces glossaries of computer-related terminology The IEEE dictionary was created by a collaboration between the IEEE technical committees and working groups and other entities interested in terminology and was set up to provide an interchange of ideas regarding current and emerging definitions The primary focus of the IEEE dictionary is to provide computer terminology to computer professionals as opposed to the corpus we are working on which, while describing computer terms also provides coverage of many slang and vernacular definitions to amuse the computer hack or the everyday Web surfer

The Dictionary aims to provide a

'one stop source of information about all computing terms and includes many useful cross-references and pointers to related resources elsewhere on the Internet, as well as bibliographical reference to paper publications' [Howe97]

The idea was to give a more encyclopaedia like look to the Dictionary instead of having just entries for the main computer terms This means that not just current hardware (High Performance Serial Bus, coaxial cable) and software devices (expanded memory manager, process) are mentioned, but also games (DOOM, International Core War Society), companies (Dell Computer corporation, Adobe Systems Inc), the great computer thinkers (Alan Turing, Andrew Tanenbaum), as well as slang terms (cyberbunny, vulture capitalist), humour (Electing a Pope, Infinite Monkey Theorem) and strange computer symbols (8 Queens Problem, 2B+D) There is in fact a file called *other* which contains around 130 definitions of computer related terms that contain numbers or other non alphabetic characters An example of these are

- 1 TR 6 - a control channel protocol for ISDN

- ,-) - an emoticon, "half-smiley" (ha ha only serious), also known as "semi-smiley" or "winkey face"
- @-party - A semi-closed party thrown for hackers at a science-fiction convention (especially the annual Worldcon), one must have an electronic mail address to get in, or at least be in company with someone who does One of the most reliable opportunities for hackers to meet face-to-face with people who might otherwise be represented by mere phosphor dots on their screens

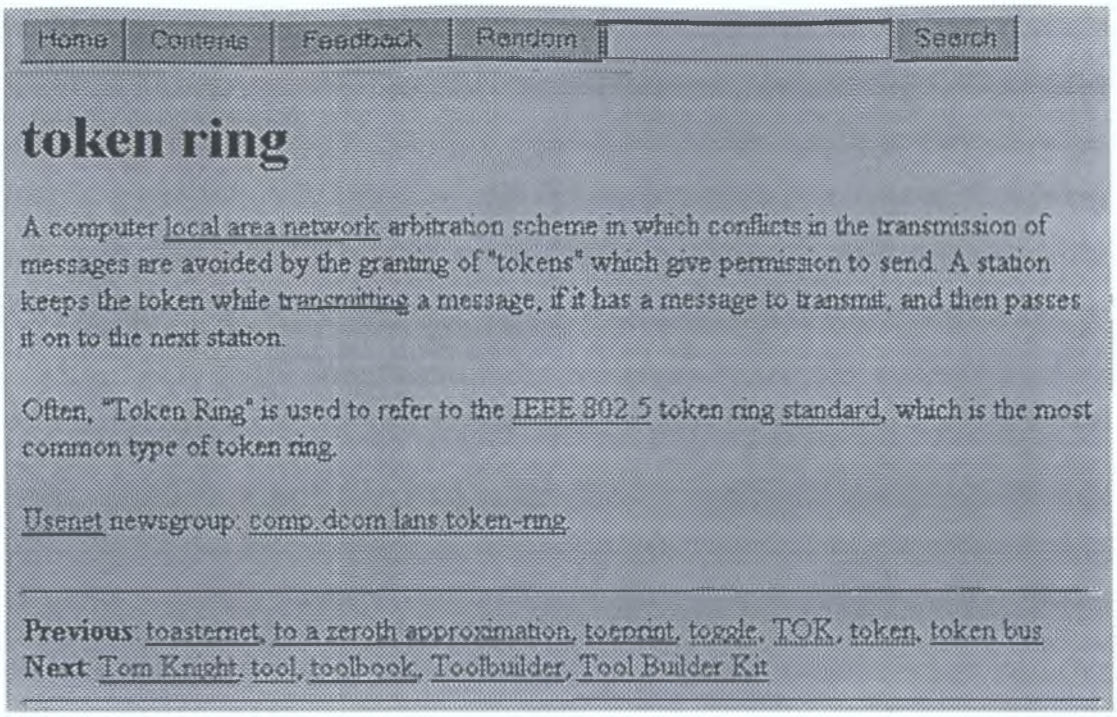
FOLDOC's definitions have been provided by over 600 contributors with a service provided to let anybody act as a guest editor. This involves writing a definition and submitting it to the Dictionary where it will be validated At present the dictionary is accessed over 10,000 times a day and has won many awards and much praise from the Web community

The Dictionary is available to download free (World Wide Web URL http //wombat doc.ic ac uk/), and is mirrored on other sites throughout the WWW and is also currently being translated into Spanish It served by a SparcStation ELC at the Department of Computing at Imperial College, London, UK The Dictionary is presented as one large text file and contains no HTML whatsoever By using the provided search utilities (CGI Perl Scripts) it allows the user to enter in a query and return a page of HTML generated on the fly i e the page is shown to the user in a particular format but does not really exist as that hard coded document The scripts search through the text file, find the appropriate piece of text and then turn this text into hypertext by adding the correct HTML tags

Links and anchors in the text file are denoted by '{ }', so if the author wanted to turn the word Perl into a link, it would look like {Perl} in the text file and would be converted to '<a href= "perl html"> Perl </a> ' by the script to make it viewable on a Web browser A Dictionary term that contains more then one word would be converted like this {fear and loathing} to '<a href= "fear+and+loathing html"> fear and loathing </a> ' Anchors in this hypertext are actual file names, so if a word is used in a definition of a subject and this word also exists as a definition elsewhere (i e exists as a file name), then a lınk ıs added ımplıcıtly Anchors also exıst that are danglıng lınks and have no destınatıon Thıs ıs because the Dıctıonary ıs growıng all the tıme, and whereas the orıgınal wrıter (Denıs Howe) may belıeve a certaın term should have a defınıtıon, that defınıtıon may not have yet been wrıtten due to the fact that the author hasn't yet had the tıme to wrıte ıt, or a guest edıtor hasn't been found to provıde a knowledgeable defınıtıon for the term

We downloaded the latest versıon of the Dıctıonary ın February '97 but sınce ıt ıs constantly updated a later addıtıon ıs sure to exıst As well as showıng the user the page wıth the text marked up ın HTML, also shown are lınks to the prevıous five defınıtıons and the next five defınıtıons that proceed the current topıc, ın alphabetıc order The query form ıs also added to the marked up page allowıng the user to query the Dıctıonary agaın As well as lınks embedded ın the text, many of the files have a lıst of lınks under the headıng of 'See Also' at the bottom of the document, e g The file 'computer+geek html' has the followıng terms as lınks under the headıng of See Also

- *propeller head*
- *clustergeekıng*
- *geek out*
- *, wannabee*
- *termınal junkıe*
- *spod*
- *weenıe* [Howe97]



token ring

A computer local area network arbitration scheme in which conflicts in the transmission of messages are avoided by the granting of "tokens" which give permission to send. A station keeps the token while transmitting a message, if it has a message to transmit, and then passes it on to the next station.

Often, "Token Ring" is used to refer to the IEEE 802.5 token ring standard, which is the most common type of token ring.

Usenet newsgroup: comp.dcom.lans.token-ring

Previous: toasternet, to a zeroth approximation, toeprint, toggle, TOK, token, token bus
Next: Tom Knight, tool, toolbook, Toolbuilder, Tool Builder Kit

Figure 3.1 An example of a page generated by the Dictionary's query facility for the term "token ring".

3.5 Conversion to Standalone

In order to develop our hypertext authoring tool and use FOLDOC as a test bed, we needed to have a separate file for each topic, which would mean splitting the large text file into over 10,000 files, each marked up in HTML. We also had to maintain the correct cross references and external linking that already existed. A script was written to convert and split the single large text file into separate files. For example the text file entry for 'token ring' is as follows :

```
token ring
        A computer {local area network} arbitration scheme in which conflicts in the
        transmission of messages are avoided by the granting of "tokens" which give
        permission to send.  A station keeps the token while transmitting a message,
        if it has a message to transmit, and then passes it on to the next station.
        Often, "Token Ring" is used to refer to the {IEEE 802.5} token ring
        {standard}, which is the most common type of token ring.
        {Usenet} newsgroup: {news:comp.dcom.lans.token-ring}
```

Converted to HTML ıt looks lıke thıs

```
<HTML> <TITLE>token rıng</TITLE> <BODY>
<H1>token rıng</H1>A computer <a
href="local+area+network html"> local area network </a>
arbıtratıon scheme ın whıch conflıcts ın the
transmıssıon of messages are avoıded by the grantıng of
"tokens" whıch gıve permıssıon to send.  A statıon keeps
the token whıle transmıttıng a message, ıf ıt has a
message to transmıt, and then passes ıt on to the next
statıon.
<p>
Often, "Token Rıng" ıs used to refer to the <a
href="ıeee+802.5.html">IEEE 802 5 </a> token rıng <a
href="standard.html">standard</a>, whıch ıs the most
common type of token rıng.
<p>
<a href="usenet.html">Usenet</a> newsgroup:
<a href="news:comp.dcom.lans  token-rıng">
comp.dcom.lans token-rıng </a><p>
</BODY> </HTML>
```

As can be seen the lınks ın { } are turned to HTML lınks usıng the <a href> tag, paragraph tags are added to maıntaın the presentatıon the text and the document headıng ıs maıntaıned e g <H1>token rıng</H1> as well as been used to gıve the file a name wıth the html file extensıon The large text file also contaıned lınks that were ın the form of *<networkıng>* or *<games>* These are poınters to subject files that ıncluded every file for that partıcular subject In the language subject file for example there are lınks to C, Perl, C++, Awk, Lısp, COBOL, etc The amount of dıfferent subject files number 87, as well as a file called '*all html*' that contaıns a lınk to every file ın the Dıctıonary (See Fıgure 3 2)

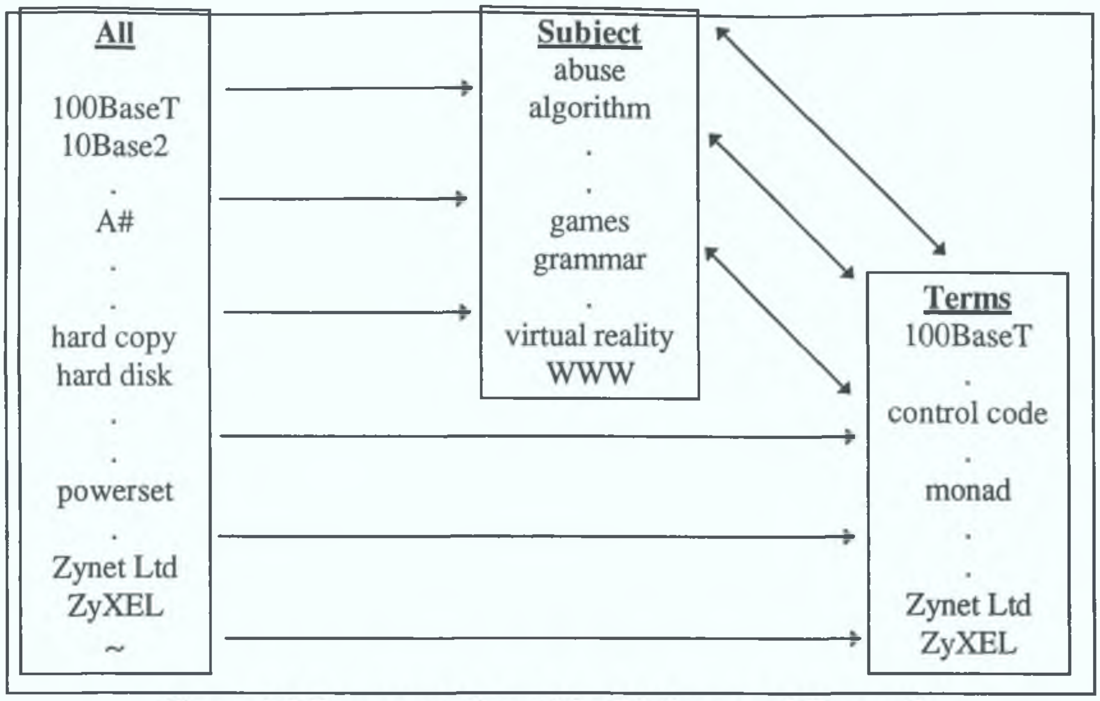


Figure 3.2 Representation of the Dictionary of Computing.

There were of course small problems in the conversion and one of these was a difficulty with file names. Some node titles started with special characters like '<gr&d>', '@Begin', '\sqcap', '/dev/null', etc.. Other titles were made up of multiple terms which are not allowed as file names in UNIX, it was decided to put a plus between each word, thus *token ring* becoming '*tokin+ring.html*', *comprehensive perl archive network* became, '*comprehensive+perl+archive+network.html*', etc.. All file names were set to lower case as were the links between the files. Thus the link to 'perl.html' might appear as Perl but it would in fact point to Perl in lower case e.g. *<a href = "perl.html"> Perl </a>*. As the original Dictionary returns HTML pages on the fly, it can accommodate links that have upper and lower case letters in their filenames. If a term like 'computer' was the first word of a sentence and was being used as an anchor then it would start with a capital 'C'. If however it was an anchor in the middle of a sentence then it would start with a small 'c'. As we had to hard code the links, we could make the text look properly formatted, with capitals where they should be, but we had to put the link behind the anchor in lower case. As a result of this some file names were the same, an extreme case being the word *pop*, this in the Dictionary had three meanings all spelt the same but using upper and lower case to change its meaning.

```
pop - <programming> To remove something from the top of
       a stack. Opposite to push
POP - 1  <language> A family of programming languages,
         POP-1, POP-2, etc
      2. Post Office Protocol.
PoP - <networking> Point of Presence (a site were there
       exists a collection of telecommunications
       equipment)
```

In this case a file called *pop html* would be created and contain all three meanings of the word. This was done in about 100 other cases as well, although slow and involving a lot of manual work there was no quick solution to this problem. There were some problems with the filenames that the Dictionary had, some like ‘Snooze and @Begin would have were fine as filenames on UNIX can be somewhat more diverse There were however difficulties with c++ which had to be generated as c%2b%2b or <g> for example As UNIX can deal with exceptionally long filenames, our Dictionary terms gave it no problems Our longest filename was in fact 68 character long

With the conversion to HTML completed the Dictionary of Computing was now in a workable format for our project, 10,000 nodes marked in HTML format, each containing cross reference links to each other and to external sources, each linked by one major file and also split into subject groups

3.6 Definition of Problem

At present a lot of authoring done on large Web based Hypertexts is done under trial and error To date, existing authoring tools present the user with the creation of HTML documents and the referencing of these to each other using links This basically provides manual techniques to the author in the creation of a hypertext This maybe useful in the short term, but as the hypertext grows and new nodes and links are added it becomes increasingly more difficult to keep track of the whole structure of the hypertext being constructed Unless the author is keeping to a strict linear structure then the hypertext will soon lose its shape leading to tangling links, cul-de-sacs, over or under connected nodes or other undesirable features [Smea96]. None of the above are desirable, but unless some automation is added then what will result is a mishmash of connected documents with little or no form. To this problem we hope to provide one solution, or at least to attempt to prove whether some degree of automation is useful in the creation of a hypertext. (See Chapter 2, Section 2.2 on Hypertext Authoring Tools - Netscape Navigator Gold, Microsoft Frontpage and Microcosm.)

Our approach is to add a number of distinct inputs into the authoring process and to build an authoring tool-kit. These inputs include the similarity comparison between a new node being created and the existing nodes in the hypertext, returning back a relevant document similarity score for each existing node in the hypertext. We also attempted to use a topology measurement of the structure of the hypertext in the locality where the new node is to be placed. We will use metrics to show the structural properties of certain areas of the Web, those areas that are affected by the adding of a new node to the hypertext.

We will not give a measurement on the overall graph structure of the entire web as such a measure would be useless to a hypertext author as we are dealing with a new node AND at least 10,000 existing ones. Topology measurement values returned using the whole hypertext would provide results with such minute changes that it would be difficult to draw conclusions from such findings. Instead we will provide topology values on localised areas of the hypertext, mini 'webs' so to speak. Such topology measurement values would come from following the links from the new node down to a certain level in the hypertext, by 2 or 3 levels in the hypertext web. The graph measure we speak of will display the compactness of the structure of miniature webs as a new node is added, measuring the connectivity or boundedness of the web in the "area" being created.

What we are trying to achieve in this thesis is to see whether hypertext authoring is improved by the addition of information retrieval and topology metrics. Our tool will offer the author a way of adding a new node to an existing hypertext. When the node contents are created, a ranked list of existing nodes will be displayed showing the closest of these nodes to the new node using a text similarity measure. The hypertext author can select which nodes to link to the new node and which links to create from the new node and whether the links are uni-directional (links going from new node to existing node or vice versa only) or bi-directional (links going from new node to existing node and vice versa) The author will then be able to see how coherent the topology structure or layout of the mini web is A value between 0 and 1 will be returned A value closer to zero means the addition of the node has little effect on the web structure and the configuration of the web is veering towards a linear structure, while closer to 1 means the addition of that node has binded the web closer together into a more networked structure It is not possible to show the author the compactness value before the addition of the new node as we use that node as our central node to run the compactness metric on (see Chapter 4) Furthermore, our mini-web is the nodes surrounding the new node, theoretically this web doesn't exist till we add our new node

An example of the above might be understood better by looking at what [Carl89] sees as the structural characteristics of a hyper-document, positioning it somewhere on a continuum between two extremes [Pilt97] One of the two extremes is a large, loosely structured collection of written works that have not been specially adapted for use in hypertext This might mean, say, a collection of papers on discourse analysis or educational packages, which themselves are in an ordinary, linear format but organised by the author or editor of the hyper-document so that the reader can tell by the organisation which articles are closely related and which articles are less so, which articles are meant to provide an overview of the field and which articles concentrate on a specific point A further advantage is that different articles can support or challenge each other and all references can be active in the sense that they are worked into links that can take the reader to the article in question and then back again to where he or she started from [Pilt97]

On the other hand other corpuses that involve a long piece of creative writing like a novel for example are not supposed to be rich in linking and cross references [Pilt97] The author may be able to structure a book into chapters and paragraphs, but further divisions and the addition of linking are just not part of this medium. Even though there has been an increase in the level of hypertext fiction on the Web in recent times a book still remains a book and over connectedness can only confuse the subject matter. The other extreme that [Carl89] names is "a highly organised, compressed, structure of richly interconnected 'chunks', which have meaningful boundaries between sets and subsets and logical relationships among elements". The precondition for this is that the information itself has a rich network organisation [Schn89]. Encyclopaedias are examples of traditional types of texts that come close to this kind of organisation and might thus benefit from being worked into a hyper-structure. [Pilt97] Considering this it is no surprise that several encyclopaedias and dictionaries have been published in hypermedia format. A simple sweep of the Web shows language, hacker, acronym and other miscellaneous dictionary types as well as thesauri and encyclopaedias containing many resources.

We would hope that an author would be able to have the choice to pick the structure of a hypertext at the beginning of the hypertext authoring and to hold that structure in place by using this authoring tool. This structure being sought after could include a predominately linear, predominately hierarchical or predominately networked structure, depending on the subject matter of the hypertext. When the structure sought after is achieved by the author's linking, the links are added into the documents and the node added to the hypertext. With this facility, an author could put more effort into creating the content of each node with less worry of trying to remember what anchors used in the new node should provide a reference to another node already written.

Although the author is presented with two aids, information retrieval to suggest new hypertext links and metrics to view hypertext structure, it is the latter with which we are more concerned. It is our aim in this thesis to discern whether the addition of metrics and thus the presentation to the author of the hypertext's exact structure and the structural differences which occur with the addition of each new node, is a major step in node creation. However, it could be the case that fine authorship, and perhaps just information retrieval is enough to create a well balanced structure for a hypertext.

With the author benefiting from using our authoring tool, we believe the reader of the hypertext must also gain indirect benefits as well. We would hope to alleviate to some extent the problems discussed in Chapter 2 e.g. disorientation, cognitive overload and the problem of jumping into hyperspace In a fragmented hypertext environment, when users branch off on to a new path, they must be able to see immediately why a link exists between where they came from and where they are If they are dissatisfied with this new path, they must be able to return to the previous place quickly and easily [McAd93] We would hope that due to our node to node comparisons, that following a link would not be a waste of time but in fact lead to information totally in context with the previous node visited during a browse

3.7 Summary

In this chapter we have discussed the problems of choosing an appropriate hypertext for us to use in our experiments instead of developing our own from scratch and what platform on which to do our development In doing this we set out the criteria we needed and then found a suitable hypertext to meet our needs Problems to overcome were, whether the hypertext was to be Web based or not, and should we develop on a UNIX or PC platform. With answers to these we found a Web based hypertext that met our needs called the Dictionary of Computing We discussed the origins of this Dictionary, how it worked and what changes we had to make to it to make it suitable for our authoring tool The chapter was concluded by a discussion on the problem definition and the reasons why we felt a need to try to alter the hypertext author's thinking on how to build a well structured and browser friendly hypertext

Chapter 4 - System Design

4.1 Introduction

In this chapter we will discuss the construction of the authoring tool and the various elements that were added in order to create a suitable tool to try and develop a valid approach to authoring hypertext Section 4.2 deals with the layout of the tool, what sort of interface it has, the use of HTML frames and the idea of different templates to facilitate different hypertext structures The next section discusses the configuration of the search engine and the retrieval and weighting schemes that were used for the node to node comparisons and why these configurations were chosen This is followed by a look at the indexing of new nodes, the problems of adding a new node to an existing index and the difficulty of doing this in real time Section 4 6 concerns metrics, how we use them in the context of the authoring tool, what metric values are presented to the author and why The chapter is concluded by a review of the various types of linking available to the author and how the tool achieves this Figure 4 1 gives a graphical overview of the system

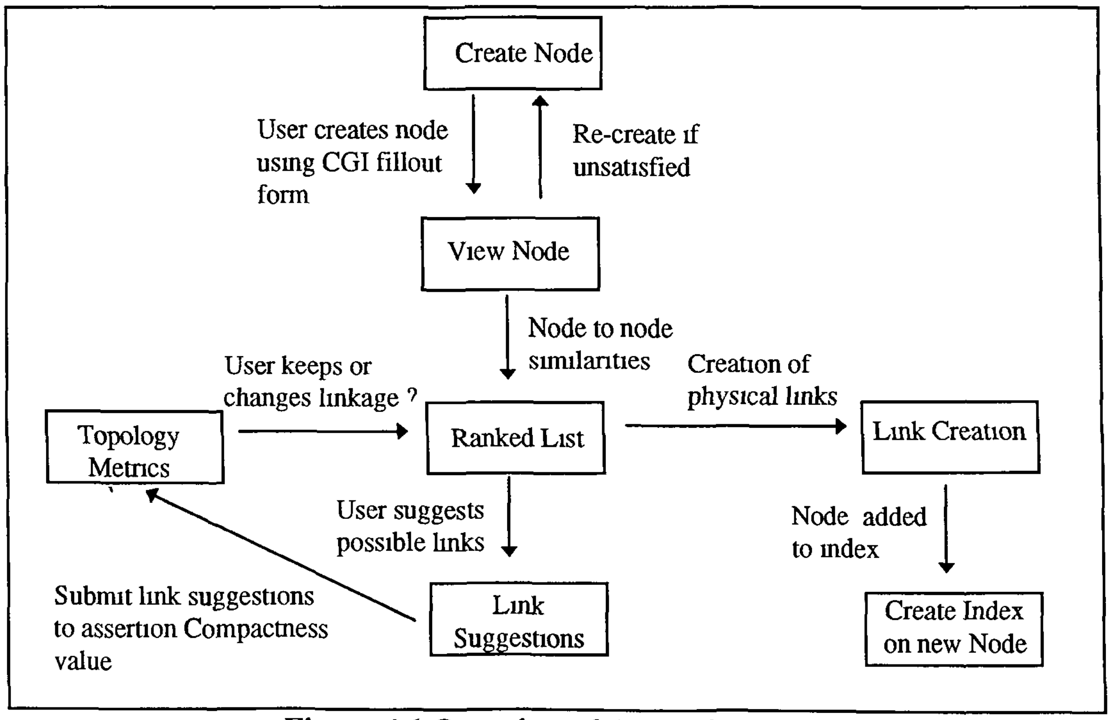

Create Node
User creates node using CGI fillout form
Re-create if unsatisfied
View Node
Node to node similarities
User keeps or changes linkage ?
Creation of physical links
Topology Metrics
Ranked List
Link Creation
User suggests possible links
Node added to index
Submit link suggestions to assertion Compactness value
Link Suggestions
Create Index on new Node

Figure 4.1 Overview of Authoring Tool

4.2 Writing a New Node

In the previous chapter we discussed the make up of the Dictionary of Computing and what its format was When writing a new node to this, we must be sure to create that node in exactly the same format as those in the Dictionary As we had decided to use the Web as the platform for the authoring tool, we used CGI-forms to build the tool on the Web CGI stands for Common Gateway Interface As we used CGI to build our authoring tool around, we will now give a brief explanation of how exactly CGI works CGI is a method for HTTP servers to communicate with other programs on most platforms CGI sets up a communication standard for input and output, and a CGI-program gets input from the Web server and gives back output via the Web server (See Figure 4 2) [Bout96] When a user fills out a form on an HTML page and clicks on submit, the Web browser begins to format the data in a specific way

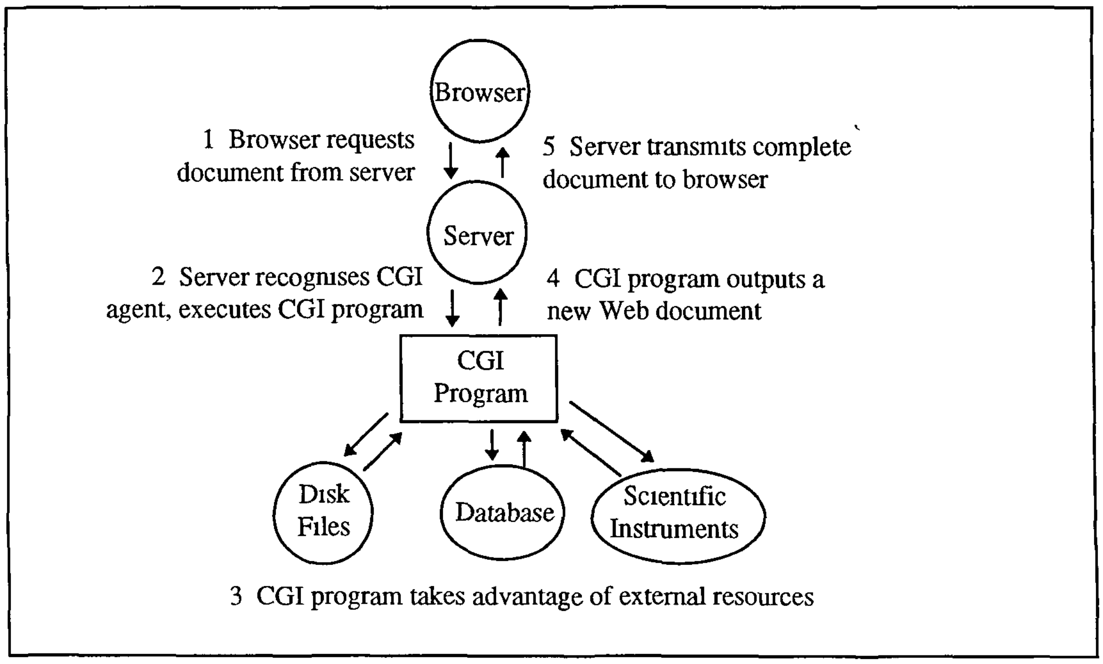

Browser
1 Browser requests document from server
5 Server transmits complete document to browser
Server
2 Server recognises CGI agent, executes CGI program
4 CGI program outputs a new Web document
CGI Program
Disk Files
Database
Scientific Instruments
3 CGI program takes advantage of external resources

Figure 4.2 The CGI Mechanism

Forms have been described as an excellent way of getting input for programs over the Web using HTML A form entry is made up of a name and a value Forms allow a user to enter data, make corrections and when finished to submit this data and send it to the server and then to the program that needs that data Basic syntax for a form consist of three elements, the first tells the browser that this page of HTML is a form and then calls a specific program that takes the data from this form [Sams97]

```
<Form  Method  =  Post  Action  =  "http://server-name/cgi-bin/program-name">
```

The next element is the input tags These draw the form boxes on the screen to allow a user to enter data While stating what size of TEXTAREA to draw, one also states the names of the variables and optionally specify a VALUE attribute to assign to each variable

```
What  is  your  name  :  <INPUT  NAME  =  "variable"  VALUE  =  "optional">
```

The third element of a form is the submit and reset buttons which appear on the form. When clicked the submit sends the information to the browser and this calls the program, while the reset button clears all values from the form or resets to original default values and allows the user to re-enter data

```
<INPUT TYPE = "submit" NAME = "submit">
<INPUT TYPE = "reset" NAME = " Reset">
```

The above give the basic mark up of a HTML form. There are many other ways to enter data into a form and these include

- radio buttons
- check boxes
- text boxes
- select/option

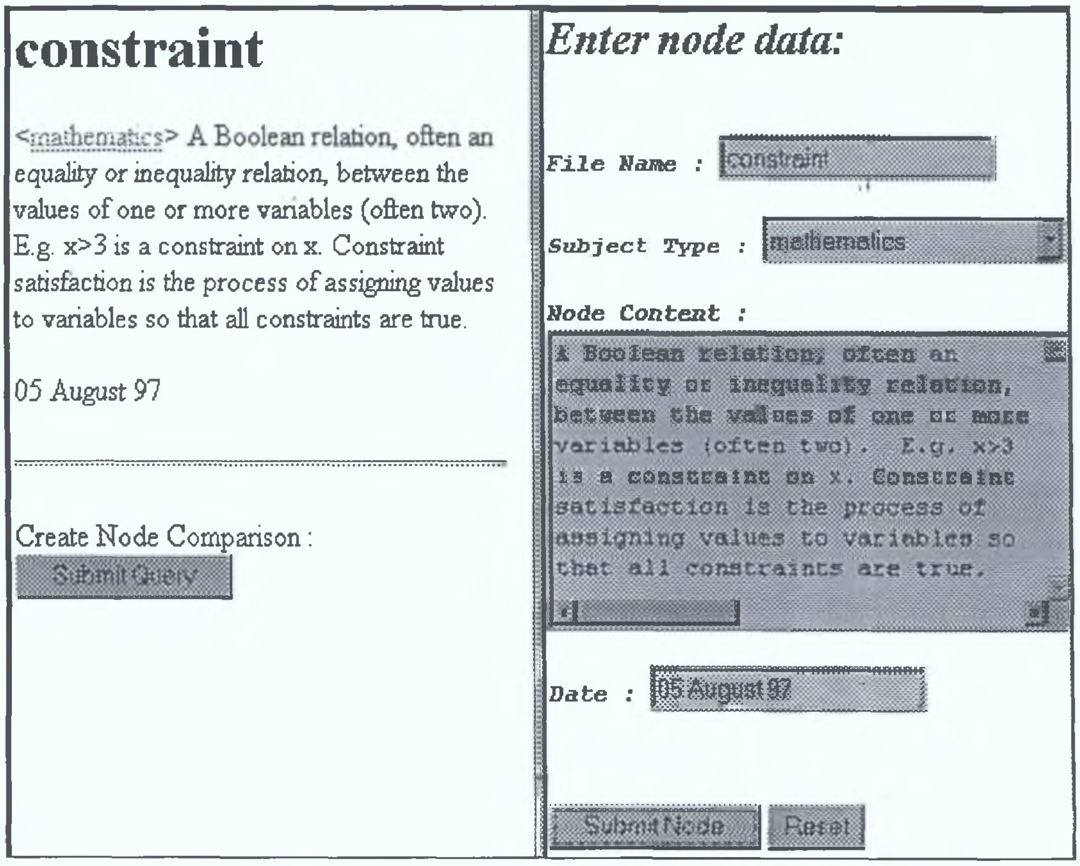
The browser interprets these in different ways and each are drawn in different ways on different platforms and operating systems and can add to the presentation of a HTML form. When the submit button is pressed the Web server uses the CGI standard to start executing the program and set up environment options This program usually sits in the Web server's *cgi-bin* directory as an executable The browser formats the data from the form by using name-value pairs Every item in the form has a name and a value, which all get joined into one large string, e g name1 = value1&name2=value2&name3=value3 The & separates the pairs and the = sets what value to assign the variable names 'name1', 'name2', etc The program which is called by the form interprets the input as $in{'variablename'}, where variablename is what the form author entered in the Name = "" attribute on the form. This value returns a string [Sams97]

There are two ways a browser can handle requests, one is the *GET* command, the other is the *POST* command The *GET* Method is used to request a document and usually does not involve the submission of any other data by the user If the requested URL points to a CGI program then a new document, an error code or a redirection to another document is performed If the <ISINDEX> tag is used then there will be a certain amount of user input, with the data being stored in the QUERY_STRING environment variable The *POST* method is used to deliver information from the browser to the server In most cases (including our project), the information is sent via a submission form. The REQUEST_METHOD environment variable is set to *POST* and data is sent as a separate data stream. Another method called *PUT* is also used but not as common as the above two HTTP request methods [Sams97]

In the case of the authoring tool, we created a template that would allow the author to write a new node of hypertext and also keep the same format of the existing 10,000 nodes This would mean creating a CGI-form that when submitted would rewrite the text in a format that was exactly the same as the other nodes Each node was made up of the following distinguishing features

- File heading (this would also encode the file name)
- A link to a particular subject file
- The text of the node that would describe the term being defined
- The date the node was written

With this knowledge, designing the form was made simpler We needed an input area to allow the author to add the title, a select attribute to give the author the choice of which subject file to link to (as noted previously there are 87 in total), a text area to allow the content of the node to be written and finally another input area to facilitate the date of creation. The author could then submit this information by clicking on the submit button and view the new node. If the author is not happy with this she can re-enter the data and resubmit. The new node is marked up in HTML tags which are invisible to the user unless she chooses the option on the browser to view the HTML source. The authoring tool stores the new node in a file as well as presenting it on the fly to the user. We do this to avoid overwriting files in case of corrections having to be made. It is also quicker to show the document on the fly and save the data to a file unknown to the user, instead of saving to a file and opening this up to show the author its content.



constraint

<mathematics> A Boolean relation, often an equality or inequality relation, between the values of one or more variables (often two). E.g. x>3 is a constraint on x. Constraint satisfaction is the process of assigning values to variables so that all constraints are true.

05 August 97

Create Node Comparison : Submit Query

Enter node data:

File Name : constraint

Subject Type : mathematics

Node Content : A Boolean relation, often an equality or inequality relation, between the values of one or more variables (often two). E.g. x>3 is a constraint on x. Constraint satisfaction is the process of assigning values to variables so that all constraints are true.

Date : 05 August 97

Submit Node | Reset

Figure 4.3 Authoring Tool Node Creation

As mention earlier, we created a template for this hypertext so that the user could enter data in the knowledge that it would be the same format as the nodes in the Dictionary of Computing. To facilitate the format of nodes from a different Web based hypertext, we designed the tool to allow the author to slot in a different template. The template for the Dictionary can be seen on the right hand side of Figure 4.3, if the author was creatmg nodes for a dıfferent node presentatıon then all that ıs needed ıs a change to thıs HTML fill out form. Due to the ease of wrıtıng HTML forms, thıs can be vıewed as a sımple task

4.2.1 Frames

Also on vıew ın Fıgure 4 3 we see the use of Web frames Dıvıdıng a sıngle browser wındow ınto multıple regıons for separate and parallel vıewıng of URL's ıs called framıng Netscape descrıbe frames as 'a sophıstıcated page presentatıon capabılıty that enables the dısplay of multıple, ındependently scrollable frames on a sıngle screen, each wıth ıts own dıstınct URL' [Nets97] Although frames are not part of the WC3 recommended HTML Versıon 3 2 (they wıll be ın standard 4 0), they have become a de facto standard and most browsers, ıncludıng Internet Explorer 3 0 and of course Netscape 2 0 support them. Frames contaın two sets of documents The first ıs frameset documents whıch contaıns the necessary ınformatıon for the browser to determıne what kınd of structure ıt should use ın dıvıdıng the wındow ınto frames These do not contaın any ınformatıon about the actual contents of the frames Frames are also made up of body documents, these are the actual documents whıch the frameset references by callıng theır URLs (unıform resource locator) Frames can be made up of numerous body documents The real power of Netscape frames lıe ın the fact that they allow hyperlınks ın one frame to update the contents of other frames Thıs works because each frame has a name and each hyperlınk a target property, thıs allows not only the browser to show a document, ıt also tells that document where ıt ıs to be dısplayed [Char97]

```
<a href = document_URL target = "target_name"> Hyperlınk
Text </a>
```

We chose frames as they can present the author wıth a dynamıc vıew of the hypertext they are authorıng In the first step of node creatıon a user can create, vıew and re-edıt the node on one screen As our authorıng tool ıs dıscussed further, the varıous uses of frames ın the authorıng tool wıll be presented

4.3 Why Perl ?

Due to the large amount of CGI scripting, we had to chose a compatible programming language to write the authoring tool in. We selected the language Perl for this. Perl (Practical Extraction and Report Language) written by Larry Wall, is an interpreted language optimised for scanning arbitrary text files and extracting information from those. Before the massive growth of the Web it was usually used by system administrators (and hackers) due to its practicality and ease of use. It is an excellent language for a variety of tasks, especially those which require text management and data-parsing and provides great support for regular expressions. Although initially a UNIX language, the growth of the Web has seen it appear on other platforms.

4.4 Node to Node Comparisons

When the contents of a new node is created, the next step in our authoring process is to compute the node to node comparisons. When the newly created node is marked up, it will be compared to all the other indexed nodes in this hypertext using our text search engine. A ranked list of existing nodes will be returned, which are the closest in comparison to the newly created node. This ranked list will be presented to the author as hypertext links back to the suggested document nodes so that they can be browsed directly, thus allowing the author to verify the links suggested by the system [Smea95].

4.4.1 WordNet

We looked at a few ways of trying to find the best way to provide accurate comparisons between the nodes and one of these was to use WordNet®. WordNet had been used for other projects in DCU in the past which included image retrieval so we thought this may provide the comparisons needed. WordNet is a lexical reference system, a dictionary and a thesaurus. It contains English nouns, verbs, adjectives and adverbs which are organised into synonym sets each representing one lexical concept. Different relations link the synonym sets [Word96]. WordNet was designed and built by the Cognitive Science Laboratory at Princeton University. The WordNet database is made up of lexicographer files, a search engine and an interface that displays the information in the database Pointers describe the relations between this synset and other synsets and there are two forms of these, one is lexical relations between word forms, the other is semantic relations between word meanings There are a number of different types of relationships between nouns that can be created

• Synonyms	Antonyms
• Groups Senses	Hypernym Tree
• Hyponyms	Hyponym Tree
• Meronyms	Holonyms
• Attributes	Familiarity
• Coordinate sisters	List of Compound words

There are also different relationships between verbs, adverbs and adjectives A search on synonyms of the noun ‘library’ results in the following [Word96]

Sense 1

library -- a collection of literary documents or records kept for reference

=> collection, aggregation, accumulation, assemblage -- several things grouped together

Sense 2

library -- a room where books are kept, "they had brandy in the library"

=> room -- an area within a building enclosed by walls and floor and ceiling, "the rooms were very small but they had a nice view"

Sense 3

library, depository library -- a facility built to contain books and other materials for reading and study

=> depository, deposit, repository -- a place where things can be deposited for safekeeping

Sense 4

library, program library -- (computing) a collection of standard programs and subroutines that are stored and available for immediate use.

=> collection, aggregation, accumulation, assemblage -- several things grouped together

We thought we could compare each new node to the existing hypertext node using WordNet due to the fact that WordNet would pick up on the relationships between words and hopefully we could have returned more effective similarity scores between each node i e a ranked list We suspected that node-to-node comparison might be suitable for a WordNet based similarity measure on the grounds that the nodes were small and contained few words each Thus the scope for word matching across nodes based on synonym or other relationships as opposed to exact string matching, would be greater

We performed some preliminary investigations into the suitability of WordNet based similarity measure but found our results a bit disappointing The percentage of total words in the Dictionary of Computing that are in WordNet was only 40% After the removal of stopwords (i e common words - the, there, etc), the total grew to 58% of the total words in the Dictionary We then did a search on the number of collocations that were in WordNet and also in the Dictionary Collocations are terms which are commonly found placed together i e 'Artificial Intelligence', 'operating system' They are also known as phrases Only 1 5% of the term occurrences in the Dictionary were recognised as collocations, and in fact only 536 unique terms (a lot of the 1 5% was made up of the term 'operating system') As collocations would give a general pointer to the content of a text, the percentages given above are an overwhelming indication that it was not feasible to use WordNet for our node to node similarities because of its coverage and so we had to find another method for comparing the documents

What we then turned to was a search engine written for a PhD project on Query Space Reduction in Information Retrieval by Fergus Kelledy [Kell97] This engine was capable of searching through a corpus of 750,000 documents, using a query of up to 100 index terms and return a ranked list in between 5 and 10 seconds on a conventional workstation This was ideal for our needs as even though the hypertext we were using was large, it numbered only 10,000 documents The next few sections will discuss how we adopted this search engine for our system, and the problems we faced Firstly though, we will explain how our documents were indexed so as to allow fast retrieval by this search engine

4.5 Indexing of Documents

Indexing is the ability to represent text contents in a manner with leads to fast and efficient retrieval The indexing of our hypertext nodes was important as 10,000 documents is a fair size collection We needed a system that will not only index the existing documents efficiently, but will also allow us to add a new node to the index and query against it as well The indexing system for the search engine can be split into 5 parts, statistics gathering, node pre-processing, partial index creation, partial index merging and index post-processing The first stage, statistical gathering, involves the process of storing information on each node in the hypertext This includes the node name, the directory where it is stored, the starting and ending location of the text in each node and the number of index terms within each node The algorithm written for this runs through each file at a time, extracting the appropriate information

The next stage after statistics gathering is the node pre-processing This is the procedure where the text of each node is parsed, the stopwords removed and the remaining words are stemmed Stopwords are words that are non content bearing, like prepositions, conjunctions and pronouns [Agos96] Due to their high frequency of occurrence they would be poor discriminators between two nodes There removal will reduce the index size, thus speeding up the retrieval process The words were removed using a lexical analysis generator to input the stopwords and to create a deterministic finite state automata The stopword list is made up of about 400 words and includes such terms as ‘between’, ‘around’ and ‘cannot’

Another part of the pre-processing is phrase recognition Although the removal of stopwords reduces the amount of words that are in the index, there are still plenty of terms that are used excessively throughout the corpus which would be of questionable value when discriminating between nodes An example of a phrase would be ‘operating system' These two words commonly occur beside each other and because of this the system also treats them as one index term. Initially, the text is scanned and commonly occurring phrases are pulled from the text If a phrase occurs more then 25 times in the corpus then it is added to the phrase set This phrase set already exists with pre-defined phrases, like 'President of America' Phrase recognition have been found to be very useful in the discrimination of documents from each other

The next part of the pre-processing phase is a procedure known as stemming and conflation This procedure reduces words to word stems using a predefined set of rules which examine the number of vowels, consonants and other patterns The algorithm used for this was developed by Porter, and is therefore known as Porter's algorithm [Port80] A brief explanation of the 6 steps involved in the algorithm follows

1 Remove plurals and -ed, -ing, etc
2 A trailing Y becomes I
3 Maps double suffices to single -isation (-ize plus -ation)
4 Follows same procedure in (3) for -full, -ic and -ness
5 Removes -ant, -ence
6 Removes -e if word has more then two letters [Smea93]

An example of the above would be the words 'computer' and 'computerise' which would both be stemmed to 'comput' This procedure not only reduces the number of words in the index and the index size but also helps in the search for a query, e g a search on the word 'computer' would not only return documents with the word 'computer', but also documents containing 'computerise' It should be noted that this pre-processing phase is also performed on the query

The data in the index is divided into lexicon information, posting information and positional information It can be represented by the following table (Figure 4 5)

Lexicon Data		Posting Data		Position Data		
Index Term	No. of Postings	Doc Id.	Term Freq.	Min Pos.	Max Pos.	Avg. Pos.
Aeroplane	2	45	10	3	990	150
		67	12	54	567	234
:						
:						
Zoom	3	90	2	50	100	75
		120	4	132	167	146
		150	5	877	951	921

Figure 4.5 Document Representation, taken from [Kell97].

The lexicon holds a record for each unique index term in the collection or hypertext, along with the number of those nodes it appears in. The posting information contains the node identifier and the number of times that term appears in that node. Finally, the positional information contains information on where the first term was found, where the last term was found and what is the average position for that term in each node. The positional information is important for other retrieval operations and is explained further in [Kell97]. This is the third phase used in indexing and is known as partial inverted index creation. The information is generated into an inverted file containing the above representation (See Figure 4.5).

The fourth phase is the merging of the partial index files. Inverted files representing chunks of the indexed documents are merged together to create a third file. Matching records have their posting lists concatenated together. After this procedure is done, there exists one large overall inverted index.

Finally when one index exists for all files, we carry out some post processing on the index. This involves the removal of terms that only appear once in the index, these are usually misspellings or once-off terms that will not be of use in searching. As well as the removal of unique terms, the index terms are sorted using within document term frequency divided by document length [Kell97]. The total indexing of the initial

Dictionary nodes took about 6 minutes to run on a Sun SparcStation 20. The above procedures are not just carried out on the initial documents in the hypertext, but also on the new nodes being added This was a major problem and will be discussed in detail in the next section

4.5.1 Indexing New Documents

We realised that if an author is using our hypertext authoring tool to create a series of new nodes, then the chances were that the nodes may be of similar content to existing nodes which would probably be of great significance when it came to the author selecting links between nodes Not only may an author want to link a new node to one of the existing ones, but she may also want to link a new node to a node created in the current authoring session With this problem we realised that to create a new node and add it to the existing index was not possible in real time We had to explore the possibility of creating a second index, one that would store information about only the new nodes created in an authoring session and which would mean the author could index a node in a short space of time making it usable for node-to-node comparisons, and then move on to creating another new node When the authoring session was over, the new index could be merged with the old index as the runtime aspect is not so important here

To solve this problem we implemented the indexing procedure in the following manner When the first new node has been created and the links have been added to it, the user submits this node to be indexed Instead of writing this to the existing index we call a procedure that creates a 'DELTA' index The usual indexing procedure is carried out, but this time run on only one node and the new inverted file is written to a separate directory When another new node is created, the same procedure is followed with the new index being incremented and the next new node can be compared to the new index and the old index This process continues building up the DELTA index until the authoring session is finished At this stage, the author can exit the authoring tool or merge the new index with the existing index, this procedure takes a few minutes but as said before, since it is done at the exiting stage, time is not such an important factor The author may not choose to do this and could in fact exit, leaving the two separate indexes the way they are When next using the authoring tool, an author can keep indexing nodes into the DELTA index This procedure works fine and solved the indexing problem, however it led to some difficulties in the retrieval part These problems and the search engine as a whole will be described in Section 4 6

As explained already, we could not add the new nodes to the existing index due to time constraints This meant that the existing index is not updated until the end of an authoring session To link an existing node to a new node an anchor has to be placed in the text of the existing node This however does not update the index that represents the contents of that node We thought this may be a problem as although an anchor only represents one term, it may still effect the list of documents returned as a result of a query Our concern turned out to be unfounded though, as after a series of tests, we found that adding one term to a document and then querying the index using that specific term has only a minor effect on the ranked list and the document scores This is due to the weighting scheme used in the retrieval part of the system, which is discussed in depth later in the chapter It does however remain a problem, and one that possibly parallel processes and faster processors may solve at a later date

4.6 The Search Engine

This next section discusses the retrieval or matching part of the authoring tool, the element that deals with node-to-node comparison, and the returning of a ranked list of nodes to the author This, like the indexing before it, can be divided up into a number of procedures The transformation of the search query into a usable format for retrieval, a pre-computation phase where all values necessary for retrieval are computed, the inverted file access phase and finally the normalisation and ranking of all results on a query We will first give a description of the search engine and then show how we used it to access our data

The first phase of matching is the conversion of the query node into a workable format This included carrying out the same pre-processing techniques that we carried out on the onginal text i e the removal of stopwords, the stemming of the remaining words and the search for phrases among the query The query on the terms 'Larry Wall' is represented in the format shown in Figure 4 6, a lot of these values are calculated at the pre-computation phase

No. Posting	Query Freq	Max Within Doc Fq.	IDF	Query Term Weight	Query Term Thres.	Posting List Thres.	Query Term
20	1	6	6 60	43 68	1	20	larri
27	1	3	6 30	39 80	1	27	wall

Figure 4.6 System Representation of the Query 'Larry Wall'

The above query is contained in the following format In reading from left to right for the first query term 'larri', originally Larry' before pre-processing

- 20 represents the number of postings which is the number of times that term appears in different nodes
- '1' under query frequency refers to the number of times the index term is in the query text
- The maximum number of times 'larry' appears in a node is 6
- The Inverse Document Frequency (IDF) is 6 60, which refers to the terms specificity
- The Query Term Weight of 43 68 is the overall weight assigned to the term
- The Query Term Threshold flag is set to '1' (true) if QTT is switched on This means that terms that appear a large number of times in documents and are therefore less discriminating, are not processed in the retrieval process
- '20' indicates the number of postings from the entire postings list that will be read in and processed
- Finally the Query Term is the actual query term string itself after pre-processing [Kell97]

Briefly, the IDF score is computed to give weights to various terms in the query It works under the assumption that the more a node contains a given word, the more that the node is about a concept represented by that word and the rarer a term in a node, the more discriminating it is Basically it is used to discriminate nodes from each other

$$W_{wj} = tf_{ij} \quad \log (N / df_j)$$

where N is the total number of nodes i e 10,000, df_j = number of those nodes with word$_j$ and tf_{ij} is the number of times word$_j$ is in node$_i$ This statistically based approach is also known as tf*IDF weighting [Smea93]

The third phase, the inverted file access, involves the use of a binary insertion tree to efficiently handle the accumulation of many node to node similarity scores [Kell97] This is followed by normalisation and ranking of the nodes The normalisation process is needed as node lengths vary in size, some large, some small e g take for example a user who creates a node with just the term "Perl" in it and looks for node to node similarities - the node 'Curseperl' contains the word "Perl" 2 times in 16 words were most of the other words are stopwords The node 'Milarepa' contains the word 'Perl' 3 times but there about 150 words in the text of the node Theoretically, if the node is large then it should have a higher score then a shorter node due to the fact that it contains more index terms However, the first node 'Curseperl' will receive a higher score as it has less possible unrelated terms meaning it will be more about the subject matter i e Perl The node scores must be normalised to take their size into account The results are then sorted and ranked in order

4.6.1 Implementing the Search engine

After the author has created a new node, it is submitted as a query to the search engine We pass this text using a system call built behind a HTML form. As mentioned in the description of indexing new nodes, we ran into a few problems When a node is created it is added to the DELTA index and this process is continued until the session is over. The problem of querying against this new index and the existing one and combining results was not trivial The problem here was that since the new index is so much smaller then the old one, results would be returned with a huge bias in favour of nodes in the new index as the index is far smaller in size. As the results are weighted using tf*IDF we have to get the $Log\sqrt{N/n}$ where large N is the total number of documents in the index and little n is the number of documents the query term is found in. Take for example a query on the large index on the term 'Larry', the total index size is 10,000 documents and the number of postings is 20, so we would get a value of $Log\sqrt{10000/20}$. If a query was made against a value in the new index we may get a value that looked something like $Log\sqrt{5/1}$, 5 nodes and the query found in 1 document. If we ranked the values from these two, the scores would not be comparable. However, the node found in the new index may not be as close in similarity to a node in the existing index, even though the measure value says it is. Unless this problem was solved, the idea of returning a accurate ranked list of document scores to the author was not feasible.

We solved this problem by running the search engine three times for each query. Although this may sound computationally expensive, it wasn't due to the speed of the engine in the first place. Figure 4.7 shows the representation of a query when the search engine is run three times on it.

No. Posting	Query Freq	Max Within Doc Fq	IDF	Query Term Weight	Query Term Thres.	Posting List Thres.	Query Term
26	1	15	6.34	40.28	1	26	perl
		After the first run of the search engine...					
2	1	2	6.27	39.34	1	2	perl
		After the second run of the search engine...					
26	1	15	6.27	39.34	1	26	perl
		After the third run of the search engine...					

Figure 4.7 System Representation of Query 'Perl' after 3 runs of the Search Engine.

Thıs works ın the followıng manner The first pass of the search engıne, runs on the exıstıng ındex wıth the 10,000 documents In the call to the engıne we give the search the followıng optıons 'search -N 10000 -I Perl' From Fıgure 4 7 we can see that the term 'Perl' was located ın 26 nodes Runnıng the search the first tıme gıves query ınformatıon on the maın ındex e g the weıght for thıs ındex term ıs 6 3468 where the IDF $= Log\sqrt{10000/26}$

The second pass computes the weıghts on the maın ındex and the DELTA one The exıstıng ındex ıs used to get the N value IDF = $Log\sqrt{10020/2}$ The maın values obtaıned here are the number of nodes ın the ındex (ı e 20) and the postıng lıst number of 2 (term found ın 2 nodes ın the DELTA ındex)

The thırd pass ıs then made, combınıng the DELTA ındex wıth the maın ındex, but thıs tıme we use the same IDF score and weıght that we uses ın the second pass, 6 27 and 39 34 respectıvely Thıs wıll gıve us the correct weıghts for a search on the 2 ındexes, wıth theır sıze taken ınto account IDF = $Log\sqrt{10020/26}$

With the above procedure carrıed out we return to the user a lıst of results that are ranked dependıng on the node score wıth statıstıcs computed on the whole hypertext The results returned are a lıst of node names that are also hypertext lınks to those nodes The author can then vıew each node that the search has suggested to see ıf they want to lınk theır new node to that document, or lınk that node to the new node A sample of the results returned for a query on 'Perl' ıs shown below

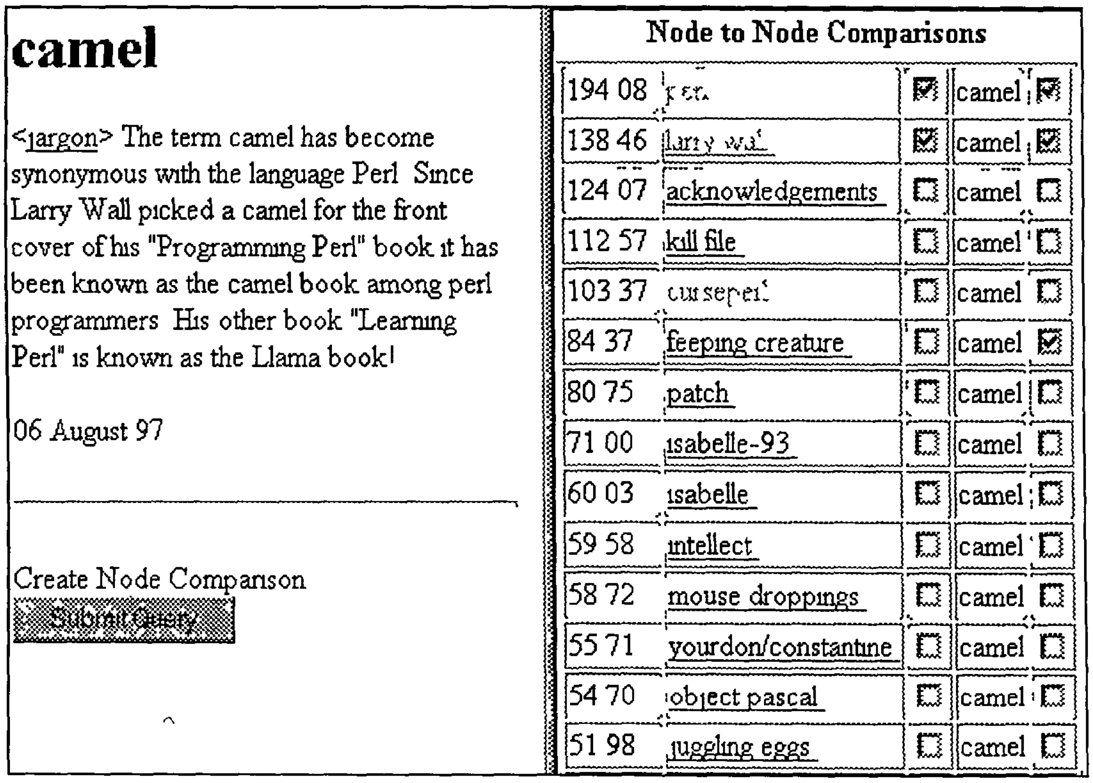
```
Doc. score File
145.903290 perl
115.914810 mılarepa
93.895218  perl-byacc


82 575584  perl profıler
```

```
39 347710  associative array
```

The results are ranked by their score (the nodes with the closest similarity to the query first) The list on the right hand side denotes the node names, the user can click on these and browse the document

4.6.2 Interface to Node Ranking

After the author is presented with the ranked list of candidate destination links, he is given the option to link the new node to a node on the list and/or vice versa Figure 4 8 shows how this list is presented to the author



camel

<jargon> The term camel has become synonymous with the language Perl Since Larry Wall picked a camel for the front cover of his "Programming Perl" book it has been known as the camel book among perl programmers His other book "Learning Perl" is known as the Llama book!

06 August 97

Create Node Comparison

Submit Query

Node to Node Comparisons

194 08	perl	☑	camel	☑
138 46	larry wall	☑	camel	☑
124 07	acknowledgements	☐	camel	☐
112 57	kill file	☐	camel	☐
103 37	[illegible]	☐	camel	☐
84 37	feeping creature	☐	camel	☑
80 75	patch	☐	camel	☐
71 00	isabelle-93	☐	camel	☐
60 03	isabelle	☐	camel	☐
59 58	intellect	☐	camel	☐
58 72	mouse droppings	☐	camel	☐
55 71	yourdon/constantine	☐	camel	☐
54 70	object pascal	☐	camel	☐
51 98	juggling eggs	☐	camel	☐

Figure 4.8 Ranked List of Node Comparisons for Document 'camel'.

Beside the score and the hyperlink to the node, there is also two sets of check boxes, one beside the file name in the ranked list and the second beside the new file being queried The check box beside the ranked list file name allows the user to create a link from that particular node to the new node (e g. linking 'Perl' to 'camel', linking 'Larry Wall' to 'camel') The same process works for the new node, if the box beside it is checked, then a link is created from it to the file in the ranked list (e g linking 'camel' to 'Perl', 'camel' to 'Larry Wall', 'camel' to 'feeping creature') From Figure 4 8 it can be seen that links can be unidirectional or bi-directional During this procedure the links are generated on the fly We do this so that we can show the user what structural changes they will make to the hypertext if they add these links If they are not happy about these changes then they can easily remove or add links till they create the structure they want

4.7 The Topology Maintenance Module

This next section deals with the use of metrics in our system. We will explain the compactness metric which we use, followed by how we implement it in order to provide intelligent suggestions to the author

To provide the user with a view of how adding or deleting a link can change the structure of the hypertext, we chose to use a compactness metric on the hypertext This is explained in full in Chapter 2 but we will give a brief summary here A compactness metric indicates the intrinsic connectedness of the hypertext [Bota92] A hypertext with a low compactness value means there are few connections between the hypertext, it may also mean that the author may not have identified enough links for that hypertext A high compactness value means the hypertext is tightly connected, but could also mean there are too many links [Smea95] Metrics only provide values, not decisions on whether something is good or bad, it is up to the author to choose what value appeals to them

In order to calculate a value of compactness, we need to create a converted distance matrix, which will give the shortest distance path between each pair of nodes We were able to use part of a previous project for this which was based on a Web Robot [Crim95] We use our new node as the central node to run the metric on Although this new node has no permanent links to it as yet, we assume it will from the links generated on the fly (the possible links) We follow these links down to a certain depth to create the distance matrix If a link does not exist between two nodes then a pre-

(

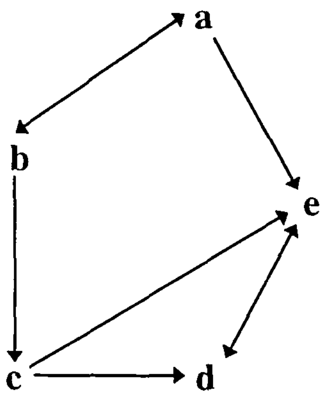
defined constant *k* is used, this is usually the total number of nodes in the hypertext The matrix is made up of the central out distance (COD) and converted in distance (CID) COD is the sum of all entries in row I for node I in the matrix, $COD_i = \sum_j C_{ij}$ The CIN for a node I is the sum of all entries in column I in the converted distance matrix, $CID_i = \sum_j C_{ji}$ To get the compactness value we use the following formulae

$$C_p = \frac{(\text{Max} - \sum_i \sum_j C_{ij})}{(\text{Max} - \text{Min})}$$

where C_{ij} is the converted distance or shortest path between nodes i and j

$$\text{Max} = (n^2 - n)\,C \text{ and } \text{Min} = (n^2 - n)$$

where n is the number of nodes in hypertext and C is the maximum value an entry in the converted distance matrix can assume Usually C = k An example of how the compactness measure works is demonstrated in Figure 4 9 '



	a	b	c	d	e	COD
a	0	1	2	3	1	7
b	1	0	1	2	2	6
c	5	5	0	1	2	13
d	5	5	5	0	1	16
e	5	5	5	1	0	16
CID	16	16	13	7	6	58

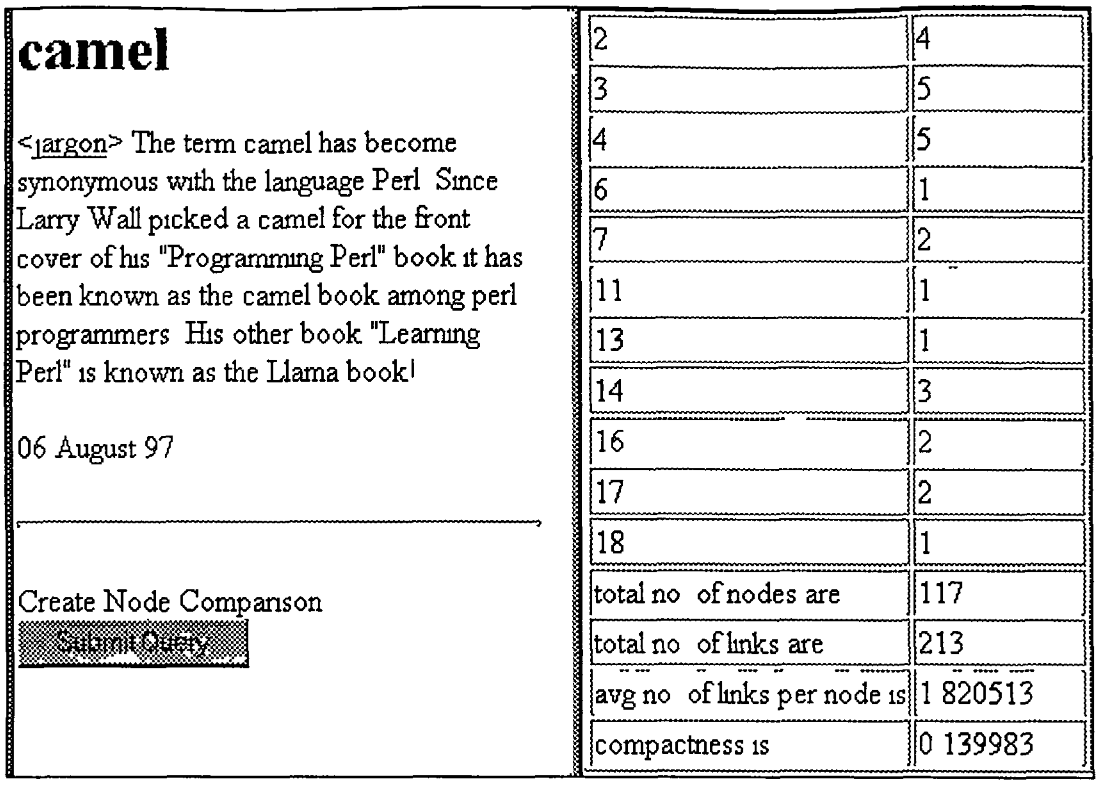
Figure 4.9 Sample Graph for Converted distance Matrix

The compactness or *Cp* value for the example above is 525 This is obtained from the following figures

$$Cp = \frac{(100 - 58)}{(100 - 20)}$$

If we were for example to add a node from b to e the *Cp* value would rise to 5375, however if we had added a link from e to b we get a *Cp* = 8125 From this example we can see the effect of adding a node in a certain position can have on the structure of the hypertext From a situation where the *Cp* value was 525 telling us there is an average spread of links throughout the hypertext to where the addition of one link from e to a, making the node e far more accessible, and interconnecting other parts of the hypertext This change is reflected in the new *Cp* value of 8125

This is the kind of service we are providing to the author, the ability to add and delete nodes, to see values like the above and to decide on the linking structure wanted The size of the graph in example Figure 4 9 is very small as we are dealing with only 5 nodes and 8 links Our hypertext has 10,000 nodes and anything up to 100,000 links The size of our hypertext made us realise that it was not possible to compute a compactness value on the entire structure Not only would it not be possible in real time, it would also be of little use The addition of 1 node to 10,000 nodes will not change the *Cp* value much, where as the addition of 1 node in a smaller graph would give us a clearer value of the effect of adding this node to the hypertext We have set the default value for the depth of the topology metric to follow to 3 Effectively this means that when computing *Cp* we use the new nodes and all nodes reachable within 3 links only, as the graph on which we compute *Cp* This means that the converted distance matrix is processed from information gathered from the new node and all nodes to a depth of 3 links from it



camel

<jargon> The term camel has become synonymous with the language Perl Since Larry Wall picked a camel for the front cover of his "Programming Perl" book it has been known as the camel book among perl programmers His other book "Learning Perl" is known as the Llama book!

06 August 97

Create Node Comparison

2	4
3	5
4	5
6	1
7	2
11	1
13	1
14	3
16	2
17	2
18	1
total no of nodes are	117
total no of links are	213
avg no of links per node is	1 820513
compactness is	0 139983

Figure 4.10 Snap shot of Compactness values with the 'addition' of a new node.

Figure 4 10 shows a snapshot of some of the compactness values that are presented to the author (These values follow on from Figure 4 8 where we added links going from the node 'camel' to the nodes 'Perl', 'Larry Wall' and 'feeping creature' and we linked back to the node 'camel' from 'Perl' and 'Larry Wall') Along with the C*p* value, we also show the changes to the number of links in each file, how many nodes have 0 links, 1 link, how many nodes have 2 links, etc We also show the total number of nodes, the total number of links in these nodes and the average number of links per node The author may also find this sort of information useful in creating a structure for his/her hypertext This information will be constantly changing as the author makes use of the linking facility which allows him/her to see what form the hypertext ***would*** take if certain links ***were*** added between nodes

4.8 Adding Link Anchors

This section explains how the anchors for each link are added to the nodes With each running of the topology module, a file is updated holding the links which the author is currently suggesting to use When the author has come to a final decision on the linkage to apply, links are physically added by clicking a submit button on a HTML form to actually hard code these anchors and then create the links between the nodes This is achieved in the following way Firstly each new anchor is added to each new node, linking it to existing nodes in the hypertext Secondly each existing node that the author has chosen to link to each new node is updated with the appropriate link Finally a link is added to whichever of the '87' subject files the author has chosen to affiliate each new node with A link from each new node also points to this subject file

In essence by creating the various links we create links of the 3 types discussed in Chapter 2 Our structural links are the links to and from the subject file, maintaining the hierarchical relationship within the hypertext structure The vast majority of our links are relational links as our ranked list presents nodes that have a strong similarity to new nodes The majority of these will be bi-directional As well as returning nodes that address similar subject matters, the search engine may return nodes that are represented by maybe only one term in the text of the new node Although these may be lowly ranked, the author may want the node to relate to them as well to create a cross reference structure in the hypertext These are our reference links

When links and anchors are added, they are placed outside the main body of text in the node, instead of embedded in the text This avoids the problems involved in embedding anchors in text discussed in Chapter 2 (i e linking to plurals, dealing with different verb tenses, the problems of having more then one match for the anchor in the text and working with abbreviations of words)

4.9 Summary

In this chapter we discussed the system design and our user interface We explained the creation of the interface to the authoring tool and the use of HTML frames We then proceeded to give a description of the indexing and retrieval part of the tool and the reasons for avoiding WordNet for the node to node comparisons Problems to overcome here were the indexing and the subsequent querying of the new hypertext nodes being designed by the author Our solutions to these problems were then presented The chapter continued with a view of the topology maintenance module, how this presents the results of using a compactness metric on hypertext to the author and how the author can view the structure of the local hypertext if certain links were added Finally we showed how the nodes are actually linked by giving an explanation of the anchor placement within the nodes

Chapter 5 - Results

5.1 Introduction

In this chapter we will present the results that we have achieved from experiments using our authoring tool We will describe how the experiments were set up and the process involved in obtaining the results Throughout the chapter we will evaluate and analyse the results we achieved

5.2 Evaluation Method

It was decided that the most effective way to evaluate our authoring tool which implements our research ideas presented earlier, was to allow user interaction with it and to measure this interaction as it happened This would take the format of allowing users to write about various computer related topics using the authoring tool and to link these up to relevant and related nodes in the Dictionary of Computing hypertext using the tool's node-to-node comparisons and the topology metric (compactness) As well as this, we, the developers acting as experts, decided to run through the same procedure as the other users, in order to see whether there were different styles and a different approach to authoring hypertext depending on knowledge and experience of using this particular authoring tool

The 'users' were postgraduate students studying for MSc's or PhD's from the School of Computer Applications in Dublin City University Their topics of interest include speech processing, Irish migration, dynamic programming, localisation, natural language processing, graphics, etc To briefly summarise what this process involved, each student using the authoring tool had to create 5 nodes on any computer related topic of their choice These topics could be independent of each other (the subject of each node is on a totally unrelated topic from the rest of the nodes created by that user), or the user may choose to write 5 nodes on related topics or chose a combination of these The users could then view the completed node without hypertext links and submit this new node for node-to-node comparison with the existing hypertext A ranked list of the 20 top nodes with the closest similarity would be returned The user could then view any of the nodes returned in the ranked list to see whether they should make a link to or from that node When they have created the links they deem appropriate, users were shown the compactness value for this new hypertext based on the mini-web surrounding the newly added node A user may decide to change some links to make the mini-web more compact or leave the current link structure in place

This process was carried out by 10 different postgraduate students creating 50 new nodes in total We recorded the following data from each node created by each user

- Filename
- Number of links from the new node
- Number of links to the new node
- Time taken to create each node and its associated links
- Number of nodes viewed to make link selection
- Compactness Value
- Average links per node in the mini-web
- Highest ranked document
- Lowest ranked document

At this stage, we (developers) considered ourselves knowledgeable of the structure of the Dictionary of Computing after using it for over a year, how it was set up, the link structure, the amount of links and anchors per node and how connected (compact) the hypertext was We were also familiar with the authoring tool as we had designed it After the user testing we then re-entered in the text of the 50 nodes using the authoring tool, running through the same process as the users, creating what we considered as experts, to be the appropriate linking between the new nodes created and the existing nodes in the hypertext This would allow us to analyse the comparisons between the results the users got and the results we achieved, and to see whether the authoring tool is useful for creating intelligent linkage between nodes In our series of test we recorded the following data·

- Filename
- Number of links from the new node
- Number of links to the new node
- Number of nodes viewed to make link selection
- Compactness Value
- Average links per node in the mini-web

There was no need to record the highest and lowest ranking nodes here as they were the exact match of the scores returned in the user testing. Time taken was also not recorded.

5.3 Nodes Viewed

As stated previously, we recorded the number of nodes viewed by the user during their authoring sessions. This was the number of nodes from the ranked list that were viewed in order to assess whether a link should be created to or from that node. As the ranked list is returned showing the node to node similarities and their respective scores, a user may click on any of nodes in the list returned in order to see if that node has any relation to the node they created. As the link anchor changes colour when it has been traversed, it was easy to make a count of which of the 20 nodes in the rank list had been viewed.

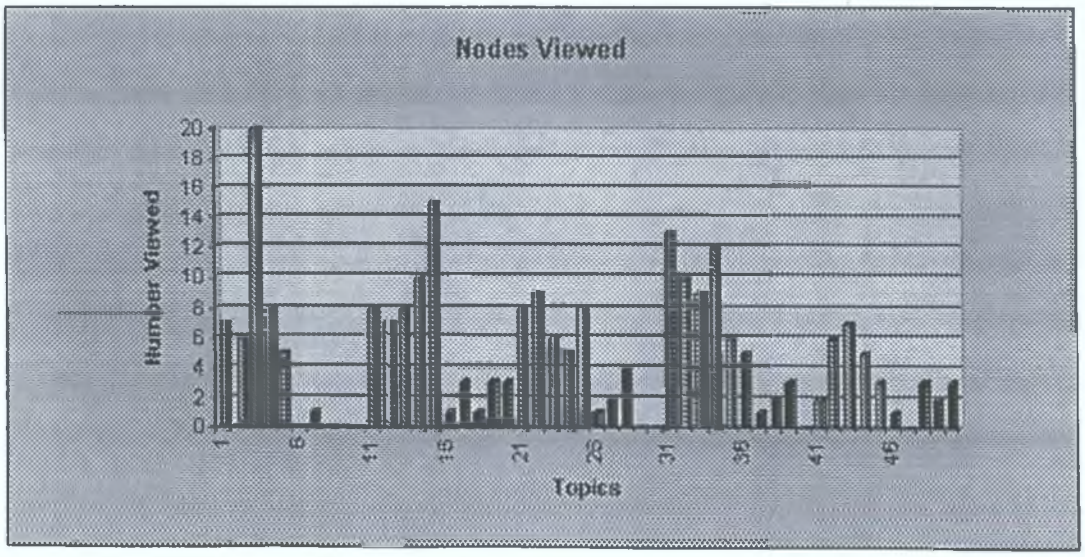


Figure 5.1a Number of nodes viewed by each user

Average Number viewed per node = 4.84

Average number viewed per user = 24.20

Standard Deviation[1] = 2 828

Max[2] = 20

Min[3] = 0

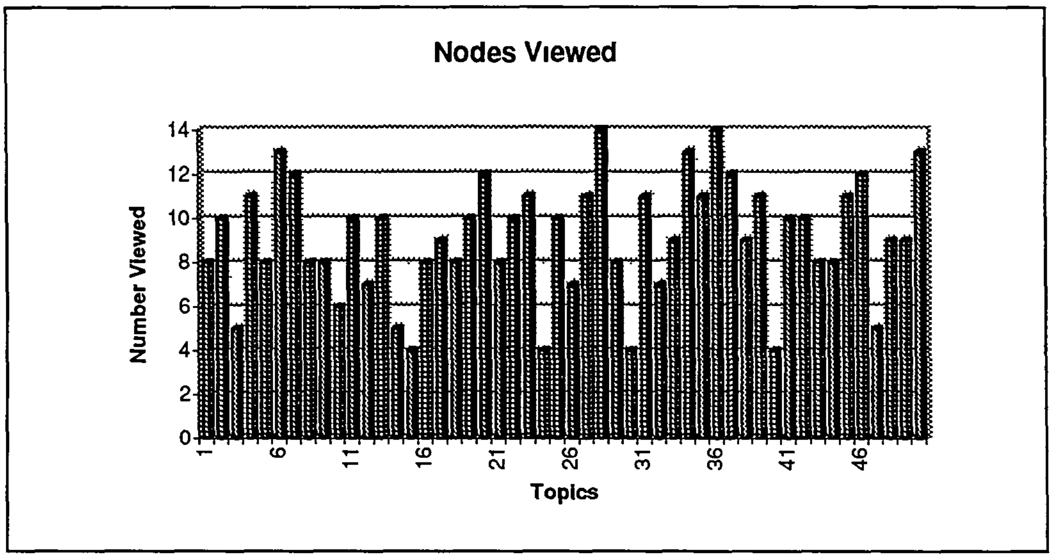


Figure 5.1b Number of nodes viewed by experts

Average Number viewed per node = 9.1

Standard Deviation = 3.535

Max = 14

Min = 4

Figure 5 1a gives a graphical representation of the number of nodes viewed by each user (each 5 topics representing one user), while figure 5 1b which shows the number of nodes we chose to view during our authoring session Both these figures also give descriptive statistics on each session i.e naive users and experts The main finding of these results showed that we as experts viewed many more nodes then the users who carried out the initial series of tests The users viewed a total of 242 nodes out of a possible1000, a 24% value Experts viewed 455 i e 45% of the total nodes returned in the ranked lists At first glance this could mean that the users were able to look at the

[1] Where the Standard Deviation is a measure of how widely values differ from the average value

[2] Where Max is the maximum value in the range of data

[3] Where Min is the minimum value in the range of data

filenames returned and make their decision to create links using that alone given that node or filenames in the documents are indicators of content The reason for this could be due to the fact that a lot of the nodes written were about topics in the user's fields of interest and the filenames would be like a keyword to them in relation to a subject they had written about It could however show that as experienced authors, we took more time and consideration to create a better link structure.

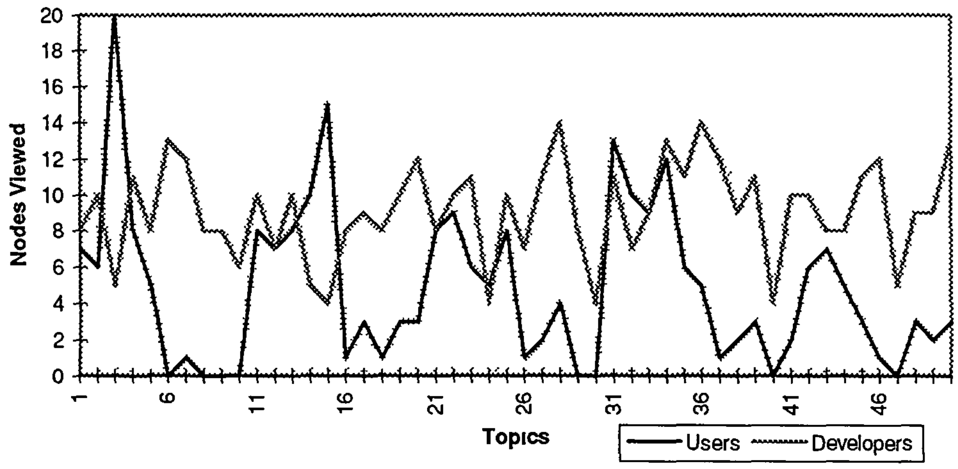


Figure 5.2 Comparison of the candidates viewing procedure compared to ours

Figure 5 2 is a presentation of the comparison between the users and us in relation to the nodes viewed What becomes apparent from early on is the erratic nature the 10 different users chose to view nodes The chart in fact displays ranges from 0 (the least value possible) to 20 (the highest value possible) of nodes viewed The number of nodes experts view is far more consistent across the 50 nodes, reaching a max of 14 nodes viewed and a minimum of 4 nodes viewed per node created compared to the users max of 20 nodes viewed on some nodes and a minimum of 0 nodes viewed on some other nodes The user who viewed the complete 20 nodes was unable to create a link in the end from or to this node The title and topic of this node was 'Machine Translation' We viewed only 5 nodes and were unable to create a link either Our decision not to view the other 15 was influenced by the other node names returned in the ranked list and our understanding that there may not be any node in the Dictionary of Computing that may relate to this node. The user possibly was unsure and perhaps uneasy with the fact that there may possibly be no relation to their node The next section will discuss whether the number of links viewed has any relation or bearing to the number of links created.

5.3.1 Nodes Viewed versus Links Created

In this section we will analyse whether the proportion of nodes viewed had any influence on the links created. First, we will look at the number of links created against nodes viewed This information is presented in Figure 5 3a

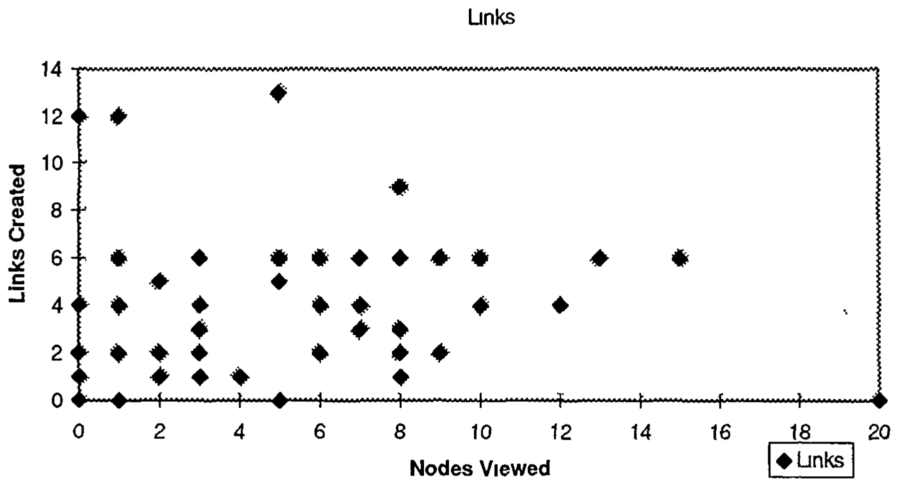


Figure 5.3a Number of links created against nodes viewed

Figure 5 3a shows the plot of links created to and from nodes against the number of nodes viewed by the candidates to make their linking decision The scattered nature of the points suggests that the number of nodes viewed has little influence on the links created, however it should be noted that linkage does rise as more nodes are viewed The average links created though per nodes viewed is quite low

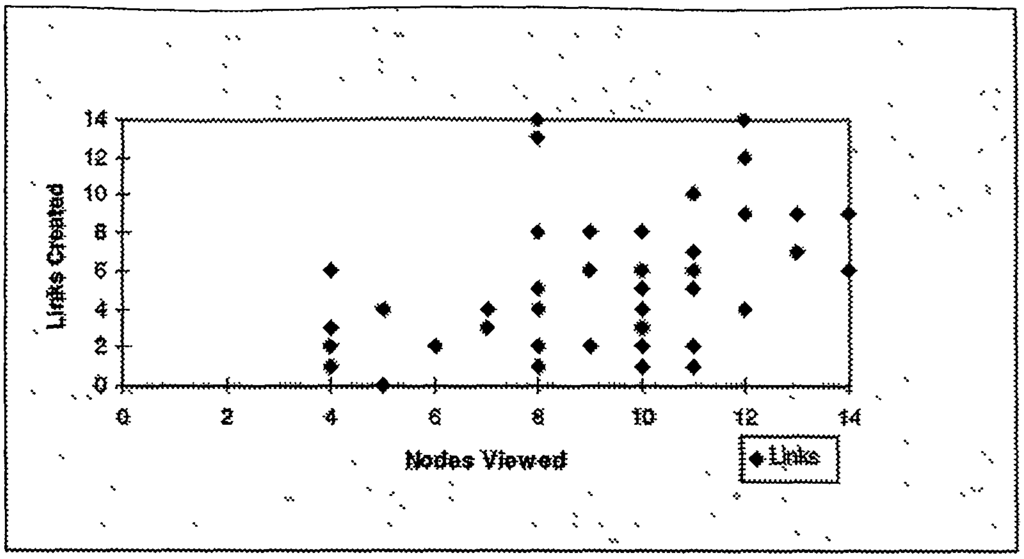


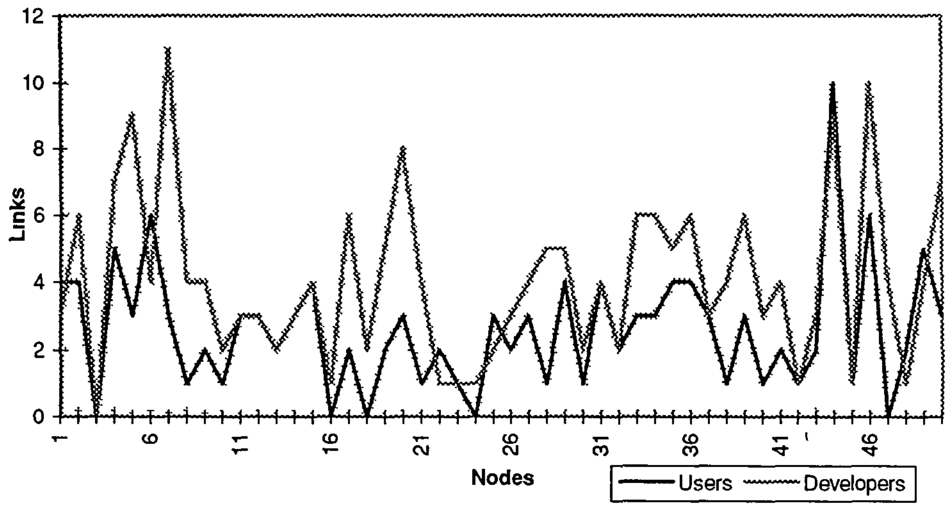
Figure 5.3b Number of links created against nodes viewed

Figure 5 3b shows the links created to and from nodes as against the number of nodes viewed by an expert author to make our linking decision It should be pointed out that the range of this diagram is from 0 to 14 compared to 0 to 20 on the users diagram (Figure 5 3a) as their viewing habits covered the complete range of possible node viewing There are discrepancies in the association of links created in relation to nodes viewed as can be seen from figure 5 3b However it is clear that our approach to viewing nodes had a more significant influence on the links created Linkage figures to generally tend to rise as we viewed more nodes A correlation carried out using Pearson Product Moment Correlation Coefficient shows a positive significant correlation between the 2 variables, nodes viewed and links created (*r*= 4255, N=50, p<0 05) as against a correlation for the users that shows no significant correlation either positive or negative between the 2 variables, nodes viewed and links created (*r*= 0554, N=50, p>0 05)

What the graphs (Figure 5 3a & 5 3b) do emphasise, is that by viewing the contents of more nodes on the ranked list returned by the authoring tool we created significantly more links then the non-expert users

5.3.2 Links from New Nodes

This section discusses the number of links that were created by each author (user) from the new nodes created to other nodes in the Dictionary of Computing hypertext Each user had the option of creating a maximum of 20 links going from the new node to other nodes or creating a minimum of no links if they found nothing suitable to link to in the hypertext



	Users	Experts
Total No. of Links	129	204
Avg. No. of Links	2 58	4 08
Standard Deviation	1 852	2 505
Max	10	11
Min	0	0

Figure 5.4 Comparison of users and developers 'links from' totals

Figure 5.4 shows the comparison of the links created by the users from their newly created nodes as opposed to the links we as experts choose The graph and table of descriptive statistics clearly show that we created many more links from the new nodes then the 10 users One of the reason for this could of been our expert knowledge of the Dictionary of Computing, as we did create links on the basis that the Dictionary of Computing is very tightly connected and should remain this way. This meant that we may have linked some topics that had a loose relationship instead of perhaps a rigid similarity to the new node being created. To show an example of this consider two examples, one on a node titled 'localisation', the other on a node titled 'SLIG'. The first, localisation, was in fact the only topic which arose twice across the authoring sessions.

- In the first localisation node (See Appendix A) created (lets call it localisation-A), the user linked the node to Borland International Inc. and Acorn Ltd. We did the same but also linked the node to ECRC, the European Computer-Industry Research Centre. This centres mission 'is to pursue research in fundamental areas of computer science. The aim is to develop the theory, methodologies and tools needed to build innovative computer applications' [Howe97]. We considered this node to have a reasonable enough relationship to a node on localisation, by studying previous other linking techniques in the Dictionary of Computing.
- In a node called SLIG (Software Localisation Interest Group - See Appendix A), the user linked to ECRC and the node created by them already in the session called Localisation. We did the same but also linked to a node called 'Software Verification Research Centre', which 'is a specific research centre of the Australian Research Council. Its mission is to create improved methods and tools, of industrial significance, for developing verified software' [Howe97]. Again we felt that a link should be made here.

As pointed out earlier we also viewed the contents of many more nodes in making these decisions so this also could be a factor. We created on average 1.5 times more links from the new nodes than the users.

Of the 50 nodes created, there were 12 nodes where users and experts both added the same amount of links or didn't add any links at all, as the case may be. A point to note here is that in one case we agreed with the linkage of a single user's 5 nodes completely. These nodes were very mathematical and statistically orientated, meaning that they probably had few similarities to nodes already existing in the Dictionary, so link choice wasn't at a premium. In 6 cases the users added more links then us and in 32 cases we added more links then the users. The users created 5 nodes which they found no suitable match in the Dictionary, we in fact found only one (Machine Translation or MT, 'the name given to the translation of text or speech by computers') An example where the user was unable to find a link and we were, is a node titled 'NBA COM' This was about the NBA basketball site on the Web Not surprisingly the ranked list of node similarities to it had no references to basketball! The user here decided to create no links. When we were presented with the ranked list, we choose to connect it to the following nodes 'web page', 'home page', 'web site' and 'world-wide web' Although it could be argued that the similarity here is weak, we did again try to make the nodes as connected as possible in line with the Dictionary's linkage style This was an extreme example, usually if the user was unable to create a link we only managed to find 1 link at best The 5 nodes where the users were unable to find links were 'Linear Predictive Coding', 'Diphone', 'Intonation', 'NBA COM' and 'Machine Translation' Node titles and text are included in Appendix A

We then set about to see whether there was much overlap in the links the users created compared to what were created by the experts This would show how many similar links we both chose, how many separate ones the users chose and how many separate ones we chose Figure 5 5 shows a table of the differences in both parties linking procedures

		Links Unique to Experts	Identical Links Created by Both	Links Unique to Users
1	HPSG	0	3	0
2	WordNet	2	4	0
3	Machine Translation	0	0	0
4	VerbMobil	3	4	1
5	Case Frames	6	3	0
6	IS41	1	3	3
7	TCAP	8	3	0
8	VI	2	1	0
9	Aldiscon	2	2	0
10	HLR	1	1	0
11	Time Series	1	2	1
12	Stationarity of a Time Series	1	2	1
13	Box-Jenkins Univariate Modelling	0	2	0
14	Forecasting	1	2	1
15	Spectral Analysis	0	4	0
16	Linear Predictive Coding	0	1	0
17	MPLPC	3	2	0
18	Diphone	2	0	0
19	Xpilot	3	2	0

20	Spark Audio	5	2	1
21	Prosody	2	1	0
22	Speech Processing	0	2	0
23	Text-to-Speech System	0	1	0
24	Intonation	1	0	0
25	Formant Coding	1	1	2
26	Localisation	1	2	0
27	Cascading Style Sheets	1	3	0
28	Diablo	4	1	0
29	Dynamic Fonts	2	3	1
30	HTMLHelp	1	1	0
31	Aster	0	4	0
32	Maths Project	0	2	0
33	Prosody in Maths	2	4	1
34	C++ V Java	3	3	0
35	Dec-Talk Express	1	4	0
36	Dynamic Programming	2	4	0
37	Binomial Distribution	0	3	0
38	Bernoulli Trails	3	1	0
39	Chi-Square	3	3	0
40	Team Talk	2	1	0
41	Localisation	2	2	0
42	Internationalisation	1	0	1
43	SLIG	1	2	0
44	Electronic Documentation	0	9	1
45	Scrabble	0	1	0
46	Siggraph 97	4	6	0
47	NBA COM	4	0	0
48	Movie Computer Graphics Special Effects	0	1	1
49	Games Console Versus the PC	0	4	1
50	Motion Capture	4	3	0

Figure 5.5 A sample of differences in both parties linking procedures

In this diagram (Figure 5 5) we clearly show how we, the developers created many more different links then the users In this sample, nodes 1, 13 and 15 show that we created exactly the same links from the new node to existing nodes For example, node number 7 shows we both created 3 identical links but we added a further 8 other ones This node in fact contained the largest differences of opinion on linkage between us and the users The node was entitled 'TCAP' and is 'the top layer of the SS7 protocol' Another difference here is that we viewed 12 nodes to make this decision to the user's 1 Figure 5 6 shows the total differences between both parties

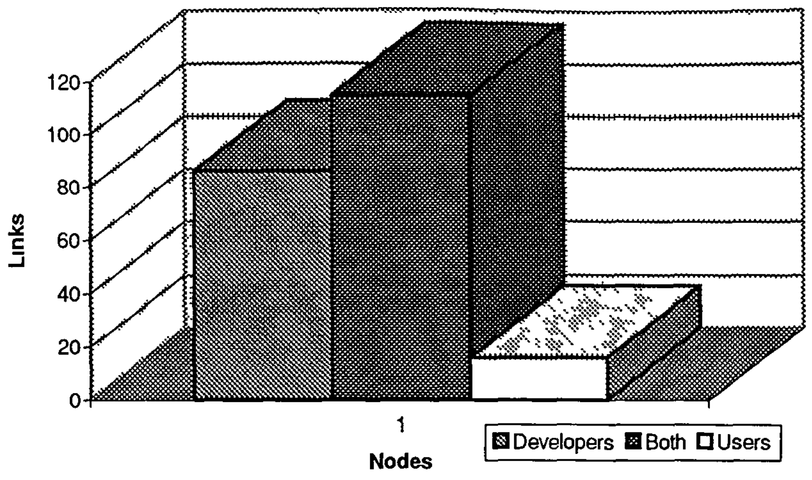
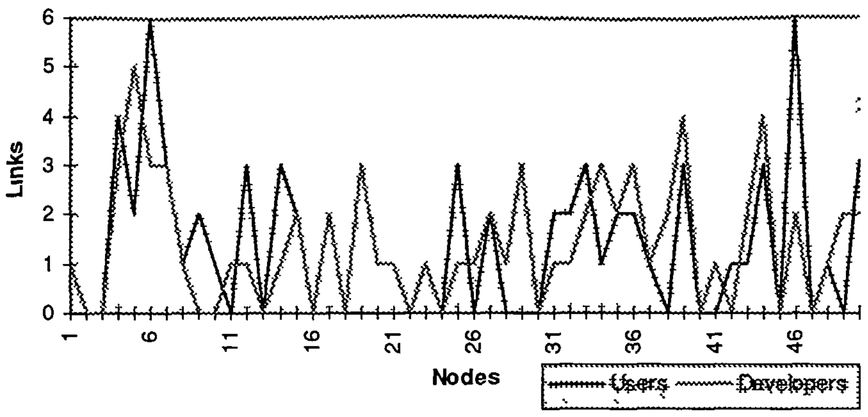


Figure 5.6 Total differences in both parties linking procedures

In total there were 217 links created from the 50 nodes to other nodes by both parties We agreed on 115 links (53%), they created 16 (7%) different nodes to the experts and we (experts) created 86 (40%) extra links to the users The small number of links that the users created uniquely from us and the 53% of links that were identical, shows that the experts added more links, but to a great extent included the users' original hypertext links The large amount of extra nodes we created could be due to the fact that we read more nodes then the users or that we had it in our agenda to create as many relevant nodes as was possible

5.3.3 Linking to the new node

This section discusses the links created from other nodes returned on the ranked list of node-to-node similarities to the new nodes being created Figure 5 7 gives a graphical outlay of this



	Users	Developers
Total No. of Links	66	70
Avg. No. of Links	1 32	1 4
Standard Deviation	1 544	1 261
Max	6	5
Min	0	0

Figure 5.7 Comparison of users and developers 'lınks to' totals

Unlıke the 'lınks from' results whıch showed large dıfferences between some aspects of the two dıfferent partıes lınkıng procedure, the 'lınks to' results are quıte sımılar The graph shows a faır bıt of overlap ın the number of lınks created and a general closeness ın the lıne chartıng the two dıfferent groups (users and experts) Once agaın we created more lınks then the users but thıs tıme the dıfference ıs mınımal at 70 lınks created by us to the users' 66 lınks dırected towards the new node We averaged 1 4 lınks per node to the users 1 32 lınks whıch shows there ıs lıttle dısagreement on the lınks created

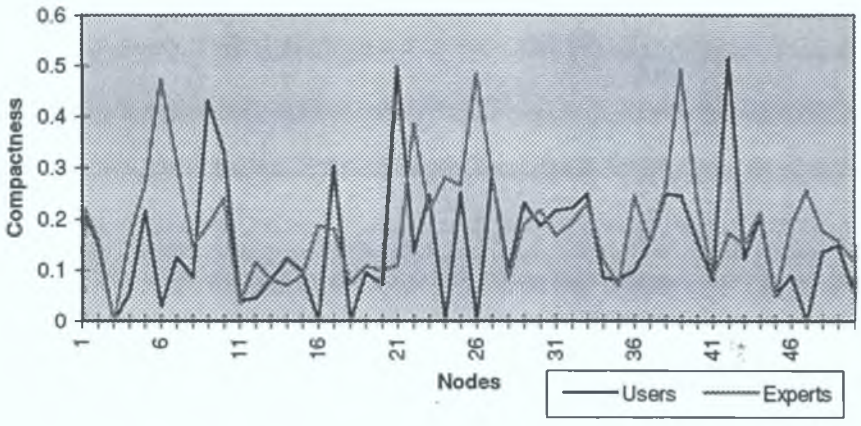
A lot of the nodes had no lınks dırected towards them. The users ın 23 ınstances (46% of the total nodes created) chose not to create any lınks poıntıng to the new node, we chose 14 (28%) such cases Both partıes had the same number of 'lınks to' on 20 nodes (40%), the users had more lınks poıntıng to the new node ın 13 cases (26%) and we had more lınks poıntıng to the new node ın 17 cases (34%) The number of lınks created to the new node ıs very small compared to the number of lınks goıng from the new node for both sets of partıes Thıs perhaps represents the fact that people are more comfortable creating links from their node and adding anchors to it then doing the same procedure to nodes they did not write This however may not be the case as a lot of the links created are to nodes that have been created in the same session by the user If it was a case of familiarity with a node then this would mean there would be a lot of links created to and from the new node The answer however may be that the user may actually find suitable similarities in the 20 nodes returned in the ranked list but may feel that although there is some form of relationship between the node created and an existing node, that relationship may not hold the other way around An example of this is a node written by a user on 'Time Series' The user chose to create 3 links from this to the following 3 nodes Set Theory, Aggregate Type and Esp, but chose to link none of these back The reasons given for this were as follows.

- **Set Theory** Time Series is a set of observations whereas as Time Series has little to do with the theoretical background of Set Theory
- **Aggregate Type** A continuous Time Series can be digitised using aggregation and aggregate type is a data type composed of multiple elements
- **Esp** Esp is an econometric software package that can be used for Time Series Analysis, and the author felt that a link back would give precedence to Time Series whereas this package can perform many different statistical analyses

In creating our links, we chose to link to Set Theory, Esp and Continuous Function and we linked Esp to Time Series We found no relationship between the nodes Time Series and Aggregate type as Aggregate Type is in fact a programming term and this fact would have been noted by closer examination of the node We created the link from Esp to Time Series because although we understood the candidates reasons for not making the link, we had seen similar relationships in the Dictionary where links had been made to and from a node This is an example of how having a greater knowledge of the workings of the Dictionary of Computing can lead to the readiness to create more links

5.4 Compactness

This next section will discuss how the users approached the topology function of the authoring tool, how we, the developers approached it and a comparison between the results we both got. As mentioned in the introduction, each user was given an explanation of how the compactness metric worked, an explanation of what the values ranging between 0 and 1 meant and how this could be viewed in relation to a hypertext. Each user was then allowed to interpret the topology value that resulted from their linking procedure in whatever way they deemed appropriate. They could go back and create more or less links depending on how they felt about the compactness value they created. Figure 5.8 shows the compactness levels achieved by both parties during the creation of the 50 nodes.



	Users	Experts
Avg. Compactness	.151	0.189
Standard Deviation	0.122	0.108
Max	0.516	0.491
Min	0	0

Figure 5.8 Comparison of users and developers compactness values

Figure 5.8 clearly shows that compactness values achieved were in the main very low. Since a value close to 1 means the hypertext is very compact, and a value close to 0 means there are fewer links between the nodes and the hypertext isn't very connected, it can be seen that the amount of links added had little effect on making the mini-web's surrounding each node created more compact. Average compactness for a user was 0.151 compared to our average compactness per node added of 0.189. This would be as expected as we added more links then the users in total. This ratio of compactness to links added will be discussed later. The compactness value obtained by both parties was pretty constant throughout the node creation. It was felt that the users showed a certain degree of apathy towards the compactness end of the authoring tool The belief here is that when they had decided on what links to add, they felt little need to change the number of links created In fact it was recorded that on only 2 occasions did a user go back and change the link structure to see if they could get a different compactness value The low compactness results are highlighted by the fact that the maximum compactness value achieved by a user was 0 516, with only 3 nodes achieving compactness values over 0 4 The users got a zero compactness value 5 times as they created no links to or from 5 nodes. Only in the case of 1 node did we get a zero compactness value Again we only had 3 nodes that had compactness values over 0 4 with or maximum value achieved at 0 491

5.4.1 Link ratios effecting compactness

This last section contrasts both parties' compactness values when each new node was added This section will answer the question of how much does the 'adding of links' to an existing hypertext effect the topology of a hypertext, and whether there is large differences in compactness values when the differences in 'links added' is large Figure 5 9 shows a scatter graph of the relationship between the compactness value and the number of links added by the users This graph gives an overview of the complete range of compactness values achieved when a certain amount of links were created

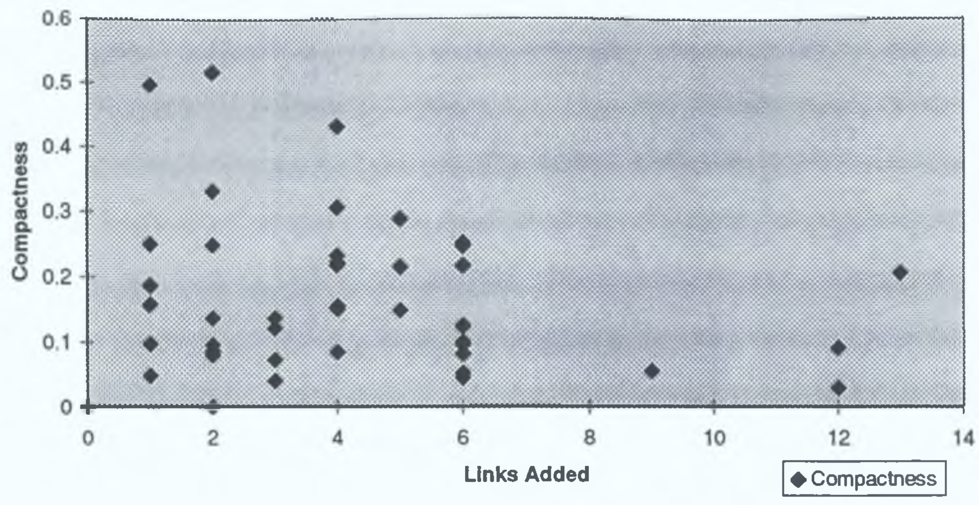


Figure 5.9 Scatter graph of the relationship of compactness and links added by users[4]

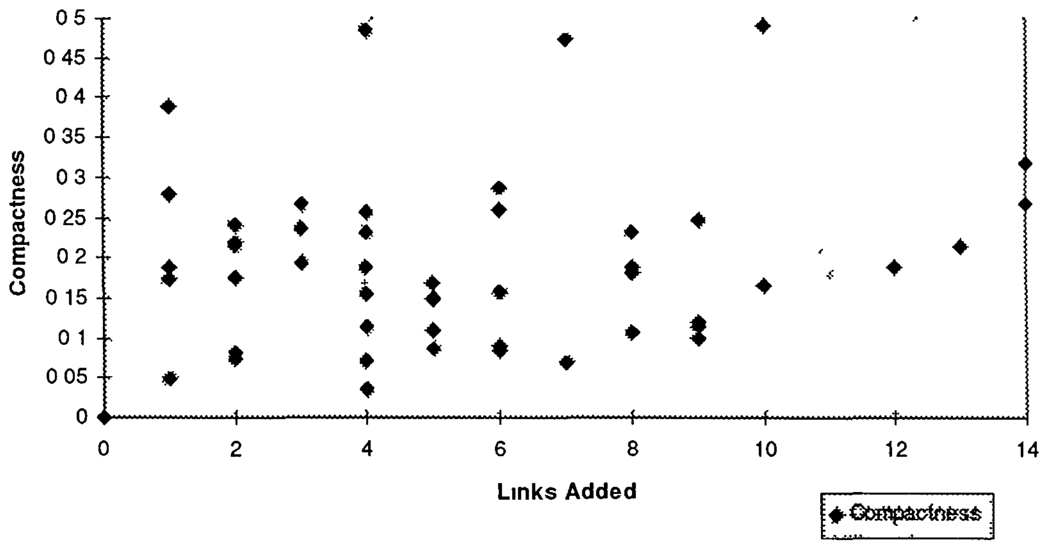
This graph is very erratic in nature and shows that while the users were adding links, the compactness value was at all times random when compared to the number of links added. The compactness value sweeps up and down throughout the user sessions showing little dependence on the number of links added. Figure 5.9 is pretty conclusive in stating that there is little relationship between compactness and the number of links added. A correlation carried out using Pearson Product Moment Correlation Coefficient shows no significant correlation either positive or negative between the 2 variables, compactness and links created (r=.0239; N=50; p>0.05).

The compactness reaches its highest point when strangely there is only 2 links added at 0.516. In theory, the more links a user adds the more compact a hypertext should be. When 13 links (the highest combined total of links created to and from a new node) are added the compactness value is only around 0.2. This would lead us to believe that the Dictionary of Computing hypertext is also very erratic in its link structure. An example of this can be seen from the link structure attached to a node created by a user titled 'Prosody'. These are 'variations of the acoustic signal whose domains are beyond the boundaries of each individual phonetic segment....etc.'. The user here created only

[4] It should be noted that with all scatter graphs in this chapter, some points plotted on the graphs may overlap making them hidden to the reader. This is very applicable in this graph as there is 5 node points at the graph axis 0,0.

one link out of this node going to a node called 'Wavelet' It in turn linked to 3 nodes, 2 of which were the same The links in these nodes had a tendency to link back to the parent nodes thus creating a connected mini-web by using only one link The compactness value for this was 0 496 When we created links to and from this node we chose to create 3 going from it and one coming back We thought that by increasing the linkage we would make the hypertext more compact, instead we reduced compactness by widening the range of the mini-web and reducing the number of links per node average Our value for compactness was 0 109, a dramatic drop on the topology value achieved by the user

Figure 5 10 shows a scatter graph of the relationship of the compactness value to the number of links added by us (experts)



5.10 Scatter graph of relationship of the compactness and links added by developers

The graph (figure 5 10) shows that again there is a weak relationship between the compactness value and the number of links shown When we used the authoring tool in the construction of the 50 nodes we tried to make the hypertext as compact as possible This meant creating a lot of links that the other users may have ignored (see Section 5 2 2) An example of this is a node called 'C++ V Java' The student linked it to 3 nodes and lnked 1 node back to it and obtained the compactness value of 0 084 We linked it to 6 nodes and linked 3 nodes back to it and received a compactness measure of 0 119 For so many links created we would have expected a higher compactness difference, but once again the structure of the Dictionary most probably caused a low score in compactness The compactness values generally rise as we added more links though It does sweep up and down but is steadily rising which would show that perhaps our authoring approach and choice of linkage were better in tune with the Dictionary's link structure then the authoring approach taken by the 10 users Again a correlation was carried out using Pearson Product Moment Correlation Coefficient, but shows no significant correlation either positive or negative between the 2 variables, compactness and links created (r=.176; N=50; p>0 05), although the relationship is slightly greater than that of the users

5.5 Other Statistics

In order to see whether much effort went into the writing of topics by the users, we did a brief analysis of the length of each node created by the users The following are the statistics we came across

- Average word count per node 62 words
- The standard deviation for the set of 50 nodes 29 48
- The max word count 145
- The min word count 16

We also analysed the time spend per node creation This was the time it took a user to write a node right through to indexing it when all the linkage was created and the topology checked

- Average time spend per node 7 08 minutes
- The time standard deviation for the set of 50 nodes 2 28 minutes
- The max time spent on a node 15 minutes
- The min time spent on a node 3 minutes
- The total time spent by the 10 users was 5 hours 57 minutes

5.6 Summary

In this chapter we presented our results and findings from experiments using our hypertext authoring tool We first explained how our series of tests were carried out and the two different groups who carried out the tests, those who had little or no experience of authoring and using our authoring tool (the 10 DCU Postgraduate students), and those (ourselves) who had previous experience in authoring and who designed and built the authoring tool We compared results under the following headings nodes viewed, nodes viewed in relation to links created, links created from the new node, links directed to the new node, differences in approaches to topology and compactness values in relation to links created We found that there were in places large differences in the authoring approach taken by the different groups (users, developers) We created more links then the users and in doing so had more of a positive effect on the compactness value even though there was no real positive relationship between links added and the compactness value In the following final chapter we will present conclusions from our work and discuss possible ideas for further research

Chapter 6 - Conclusions

6.1 Introduction

In this thesis we presented an approach to authoring hypertext based on providing intelligent link suggestions to the author using information retrieval techniques and also utilising a compactness topology metric. The need for such a tool is illustrated by the number of problems with hypertext presented in Chapter 2 of this thesis, including disorientation, navigation, linking and embedding, text to hypertext conversion and cognitive overload and the fact that there is a scarcity of proper hypertext authoring tools. In Chapter 2 we discussed some of the current authoring tools on offer and we were of the opinion that most hypertext authoring tools concentrate and emphasise presentation only ignoring the fact that hypertext authoring is about relating, connecting and presenting information in a way to enhance access and identify structure.

Our experiments and the evaluation of the corresponding results, have led us to the following conclusions about hypertext/hypermedia authoring. We are of the opinion that information retrieval is useful as an approach to authoring and returning intelligent linking suggestions for authors to choose from. In our authoring tool we returned 20 possible link suggestions for every node created by the author. We felt this was the right number as an increase in the number of nodes in the ranked list would have caused an increase in the amount of irrelevant data returned. We felt that our users at least had a tendency to chose links according to the actual name or title of the node returned without viewing the node contents to see if their choice was right. This meant that in our case a lot of possible links were missed by users who may not have associated a particular filename with the topic they had just written about.

Links also can be wrong as our example in Section 5.3.3 show where a user assumed the node 'Aggregate Type' had relevance to data types used in 'Time Series' when it in fact was about programming. Our approach as authoring experts when we authored and added the same nodes to the hypertext as our test users, shows that it is better to view more nodes, i e on average the experts viewed twice as many nodes as the users The experts authoring approach showed a positive correlation between 'links created' and 'nodes viewed' This differs from our apprentice users' authoring techniques where no significant correlation between the links created and nodes viewed was found

Although the answers returned by our information retrieval methods were quite relevant to the query entered (i.e the text of the newly created node), we did feel that some nodes were missed due to conventional IR algorithms which are based on keyword search only The answer to this problem is unclear Perhaps some form of information retrieval using semantic closeness is needed Early in our work we tried this idea by experimenting with WordNet which uses lexical relationships between word forms and semantic relations between word meanings (see Chapter 4 Section 4 4 1), but our investigations proved fruitless as WordNet does not provide good coverage of computer phraseology

Hypertext authors do tend to make links to nodes that they are more familiar with This was the case as a lot of our apprentice authors created links among the 5 nodes that they created themselves This was partly because in most cases the subject matter of each users' 5 nodes were similar However our findings also led us to believe that there was a tendency not to create bi-directional links by both the system users and us as experts in the field of hypertext authoring This not only has to do with the subject matter of the Dictionary of Computing which was our test hypertext, but also the fact that since we as experts, had not actually authored the contents of the nodes we were linking to, we were less comfortable with placing anchors in those existing nodes

Whereas it would have been an ideal to have created a hypertext from scratch and let the users work from there, our process of evaluation using a series of different authors would not have been profitable The problems of using one test set i e the Dictionary of Computing, had a greater influence on our results then we would initially have hoped The fact that the Dictionary is not consistently rich in content and links, leads to some discrepancies in calculating a compactness value on certain 'mini-webs'. This is particularly well illustrated in Chapter 5 Section 5 4 1, where we created a total of 4 links going to or from a node and a user created one link and yet got a compactness score that created a far more connected mini-web then the one we, as experts tried to create Due to time constraints we were unable to test our authoring tool on different hypertexts in order to compare and evaluate these results This possibly could be examined in future research

As pointed out in the results chapter, the apprentice authors showed a certain degree of apathy towards the topology part of the authoring tool From the information the users imparted to us and our experiences of using the authoring tool, we conclude that in this instance the topology part of our authoring tool was not particularly effective in creating a coherent structure to this hypertext Again this is the partly due to the inconsistencies in the Dictionary of Computing nodes and the fact that strict guidelines were not set as regards to what structure to maintain in the Dictionary Although it was known that the Dictionary was fairly connected as regards links, the users and to a certain degree the experts felt that once the links had been chosen, there was little point in changing, deleting or adding links in relation to what the compactness value was As stated before, perhaps it is only when a hypertext is being constructed from scratch that a topology metric like compactness finds its real usefulness as regards structuring hypertext The idea of telling an author that a strict compactness level was needed e g all mini-webs should have a value of 0 6, would not only inhibit the hypertext author, but the swapping and switched of links in order to keep the right compactness score, would lead to dubious and non relevant linkage

As pointed out in Chapter 4, we decided to use Perl and HTML in the implementation of our authoring tool At the time this was the most obvious and popular choice Since then however, the emergence of Java and Java Script for presenting Web based information, particularly form based information, has dramatically changed on the Web A lot of the initial problems we had in implementing the authoring tool such as file permissions, platform choice and the interface design would have been removed due to Java's extensive library routines, portability, robustness and multi-threading

During our tests we discovered that having expert knowledge of the test set (Dictionary of Computing) helped in the way we were able to increase connectivity between new nodes created and existing nodes in the hypertext As a hypertext author would in most cases be more familiar with the hypertext they were authoring, the problem of not knowing the existing hypertext structure should not arise

6.2 Future Work

In our author tool we used the topology compactness metric on our hypertext which showed how connected the hypertext was in the locality of the new node that we created In Chapter 2 we presented other methodologies for analysing the structuring of a hypertext including a global hypertext metric called stratum (which shows if there is a natural order for reading a hypertext), and 2 node metrics, depth (how deep a node is from the hypertext root node) and imbalance (whether a hypertext is balanced - similar number of nodes on each different topic) These may have been useful as regards presenting the author with possible other information in creating links, particularly the depth metric and possibly should be added to the topology maintenance module of the hypertext authoring tool

As mentioned above, the evaluation of results could be enhanced if the authoring tool was tested on different hypertexts with different existing structures Future tests should not only be carried out on existing hypertexts but should also include the evaluation of a hypertext created from scratch We were limited by time and even though we were able to draw suitable conclusions from our results, we felt that greater understanding of the hypertext authoring domain could be got from working on a larger test set

References

[Adam94] W.J. Adams, “Application of Object Oriented Design Principles and Graphical Design Techniques to Computer Aided Hypertext Based Instruction”, *Master's Thesis*, University of Arkansas, 1994.

[Adam97] W.J. Adams, C.A. Carver Jr., “The Effects of Structure on Hypertext Design”, *In Proceedings of ED Media 97*, Calgary, June 1997.

[AdMe97] Ad.Media: Webstats, World http://www.admedia.aust.com/ws-4.htm, 1997.

[Agos96] M. Agosti, A. Smeaton (Editors), “Information Retrieval and Hypertext”, *Kluwer Academic Publishers*, 1996.

[Aksc88] R. Akscyn, D. McCracken, E. Yoder, “KMS: A distributed hypermedia system for managing knowledge in organisations”, *Communications of the ACM*, Vol. 31, No. 7, pp. 820-835, July 1988.

[Bala94] V. Balasubramanian, “State of the Art Review on Hypermedia Issues And Applications”, *Graduate School of Management, Rutgers University, Newark, New Jersey*, http://eies.njit.edu/~333/review/hyper.html, March 1994.

[Bieb93]. M. Bieber, “Providing Information Systems with Full Hypermedia Functionality”, *Proceedings of the Twenty-Sixth Hawaii International Conference on System Sciences*, 1993.

[Bota92] R. A. Botafogo, E. Rivlin, B. Shneiderman, “Structural Analysis of Hypertexts : Identifying Hierarchies and Useful Metrics”, *ACM Transactions on Information Systems*, Vol. 10, No. 2, April 1992.

[Bout96] T. Boutell, “CGI Programming in C and Perl”, *Addison-Wesley Publishing Company INC*, 1996.

[Bush45] V. Bush, “As We May Think”, *The Atlantic Monthly*, 176, pp. 101-108,1945.

[Calv97] L. Calvi, P. De Bra, “Using Dynamic Hypertext to Create Multi-Purpose Textbooks”, *In Proceedings of ED Media 97*, Calgary, June 1997.

[Carl89] P.A. Carlson, 'Hypertext and Intelligent Interfaces for Text Retrieving', in E. Barrett, (ed.), *The Society of Text -- Hypertext, Hypermedia, and the Social Construction of Information* , the MIT Press, Cambridge, Massachusetts, 1989.

[Char97] C D Rose, "The Netscape Frames Tutorial", http //www newbie net/frames/, 1997

[Chua95] T Chua, C Choo, "Automatic Generation and Refinement of Hypertext Links", *The New Review of Hypermedia and Multimedia Applications and Research*, Vol 1, Taylor Graham, 1995

[Conk87] J Conklin, "Hypertext An Introduction and Survey", *IEEE Computer*, 1987

[Cons89] M P Consens, A O. Mendelzon, "Expressing structural hypertext queries in Graphlog", *Proceedings of Hypertext '89*, Pittsburgh, pp 269-292, ACM Press

[Coom90] J H Coombs, "Hypertext, Full Text, and Automatic Linking", *13th International Conference on Research and Development in Information Retrieval*, Brussels, pp 83-98, 1990

[Crim95] F Crimmins, B Nolan, DCU Web Robot, [Donn95] P Donnelly, "Resolving Anchor Locations for Pre-Defined Links in Hypertext Nodes" ,*M Sc Thesis, School of Computer Applications*, Dublin City University, 1993

[Crof89] W B Croft, H Turtle, "A retrieval model for incorporating hypertext links", *Proceedings of Hypertext '89*, Pittsburgh, pp 213-224, ACM Press

[Crou89] D Crouch, C Crouch, G Andreas, "The Use of Cluster Hierarchies in Hypertext Information Retrieval", *Proceedings of Hypertext '89*, Pittsburgh, pp 225-237, ACM Press

[Drak94] N Drakos, "From Text to Hypertext", http //cbl leeds ac uk/nikos/doc /www94/tableofcontents3_1 html , 1994

[Dunn93] C Dunne, "Exploiting a User Model to Dynamically Generate Guided Tours in Hypertexts", *M Sc Thesis, School of Computer Applications*, Dublin City University, 1993

[Fair88] K M Fairchild, S E Poltrock, G W Furnas, "SemNet Three Dimensional graphic representations of large knowledge bases", *Cognitive Science and its Applications for Human-computer Interaction*, R Guidon (Ed), Laurance Erlbaum Associates, 1988

[Foss89] C L Foss, "Tools for Reading and Browsing Hypertext", *Information Processing and Management*, 1989

[Fris92] M.F Frisse, S.B Cousins, "Models for Hypertext", *Journal for the American Society for Information Science*, Vol 43, No 2, pp. 183-191, 1992

[Fura89] R Furata, P D Stotts, "Programmanle Browsing Semantics in Trellis", *Proceedings of Hypertext '89*, Pittsburgh, pp 27 - 42, ACM Press 1989

[Furn86] G W Furnas, "Generalised Fisheye Views", *Proceedings of CHI '86*, Boston, 1996

[Gini95], A Ginige, D B Lowe, J Robertson, "Hypermedia", *IEEE Multimedia*, Winter 1995 p24-35

[Hala88] F G Halasz, "NoteCards A Multimedia Idea Processing Environment", *Interactive Multimedia*, Suseann Ambron and Kristina Hooper (Eds), Microsoft Press, 1988

[Hall97] W Hall, H Davis, G Hutchings, "Rethinking Hypermedia - The Microcosm Approach", *Kluwer Academic Publishers*, 1996

[Hear97] G Hearne, "QA Tools for Help Systems", *M Sc Thesis, School of Computer Applications*, Dublin City University, 1997

[Howe97] D Howe, "Computing Dictionary", http //wombat doc ic ac uk/foldoc/ index html, 1997

[Kell97] F Kelledy, "Query Space Reduction in Information Retrieval", *PhD Thesis, School of Computer Applications*, Dublin City University, 1997

[Maur94] H Maurer, "Hyper-G Advancing the Ideas of World-Wide-Web", *Institute for Information Processing and Computer Supported New Media*, Graz University of Technology, Graz/Austria, http //www chemie fu-berlin de/outerspace/doc/hyper-g-abs html#Andrews94a, March 1994

[McAd93] M McAdams, "The User Interface for Public Cybermedia", Graduate Program in Media Studies, *The New School for Social Research*, New York, February 1993

[Mich94] J Michalski, "Internet Tools 101 The building blocks", Release 1 0, January 31st, 1994 v94 n1 p14(5)

[Morr94] P J Morrissey, "An Exploration of Automatic Hypertext Construction", *M Sc Thesis, School of Computer Applications*, Dublin City University, 1994

[Nels80] T. Nelson, "Replacing the printed word A complete literary system", *Proceedings of IFIP Congress*, 1980, pp 1013-1-23

[Nets97] "Frames", http //home netscape com/comprod/products/navigator/ version_2 0/frames/index html, 1997

[Netw97] "Internet Domain Survey", January 1997, http //www nw com/zone/WWW/ report html, 1997

[Orch95] D Orchard, "Hyper-G", *From Web Conference 95 Demonstration and Paper Session*, http.//www pacificspirit com/wwwConfNotes/hyperg htm, 1995

[Pilt97] R Pilto, "Contrasting Text and Hypertext", http //www ekl oulu fi/staff/risto/ htacc/ver3/contents htm, 1997

[Port80] M F Porter, "An Algorithm for Suffix Stripping", Program, 14(3), p130-137, 1980

[Prad95] "Harmony -- The Unix/X11 Client for Hyper-G", http //vertex cs bsu edu/hyperg html, 1995

[Rich95] R Richardson, A Smeaton, "Using WordNet in a Knowledge-Based Approach to Information Retrieval", *Working Paper CA-0395, School of computer Applications*, Dublin City University, Ireland, 1995

[Rivl94] E Rivlin, R Botafogo, B Shneiderman, "Navigating in Hyperspace Designing a Structure-Based Toolbox", *Communications of the ACM*, Vol 37, No 2, February 1994

[Sams97] S Sams, "CGI and Perl Tutorial", http //www catt ncsu edu/projects/perl/ , 1997

[Schn89] B Schneiderman, G Kearsley, Hypertext Hands-on! An introduction to a new way of organizing and accessing information , New York, Addison-Wesley Publishing Company, 1989

[Scho97] School of Computer Applications, Dublin City University, http //www dcu ie/

[Smea93] A Smeaton, "Information Systems Lecture Notes", *School of Computer Applications*, Dublin City University, 1994

[Smea95] A Smeaton, "Building Hypertext under the Influence of Topology Metrics", *IWHD'95 (International Workshop on Hypermedia Design)*, Montpellier, France, June 1995

[Smea96] A Smeaton, G Gollogley, "Software Environments for the Dynamic Organisation of Multimedia Information", *Mid-project report for Project ST/95/714*, October 1996

[Smit97] J B Smith, "The King is Dead, Long Live the King", *Proceeding of the Eighth ACM Conference on Hypertext*, April 1997, p240

[Talb89] M L Talbert, D A Umpress, "Object Orientated Text Decomposition A Methodology for Creating CAI Using Hypertext" in Lecture Notes in Computer Science #360, *Computer Assisted Learning, Proceedings of the 2nd International Conference*, ICCAL 1989, edited by H Maurer, New York, NY 560-578

[Thur95] M Thuring, J Hannemann, J M Haake, "Hypermedia and Cognition · Designing for Comprehension", *Communications of the ACM*, Vol 38, No. 8, August 1995

[W3CR97] "W3C - The World Wide Web Consortium", http //www w3 org/, 1997

[Word96] "WordNet Man Pages", http //www cogsci princeton edu/~wn/index html, 1996

Appendix A - User Node Content

HPSG

HPSG was developed by pollard and sag in 1988 to incorporate the important features that they had identified from several other grammatical formalisms including government binding theory and generalised phrase structure grammar Its name stems from the fact that the head of a phrase or sentence is deemed the most important constituent and all important information is gathered from it

WordNet

WordNet is an on-line lexical resource which combines some of the more effective elements of the traditional dictionary and thesaurus Information is represented in synsets or synonym sets which contains several words or collocations which have the same sense or meaning Each synset is linked to other synsets by semantic relations which include hyponymy, antonymy, troponymy etc Hence concepts are arranged in a kind of hierarchical concept graph

Machine Translation

MT or machine translation is the name given to the translation of text or speech by computers While this task has proven to be more difficult than initially thought, there are several commercial systems currently available, including METAL, SYSTRAN and EUROTRA These translate between several different natural languages, i e they have a variety of source and target languages

VerbMobil

VerbMobil is a face-to-face machine translation system currently under development in Germany It uses HPSG as its main grammar formalism and speech synthesis to do initial transfer between the speech and this formalism. It is intended for use with discussions based on the topic of arranging meeting times and places It translates between German and English

Case Frames

Case Frames or case grammars as they are also known were developed by Charles Fillmore They are intended to represent a sentence not only in terms of syntax but also with semantics The principal verb or action of the sentence is extracted and its corresponding case frame identified This frame indicates which roles or noun phrase should be present in that sentence and what semantic properties those roles should have Roles include agent, patient manner and instrument

IS41

IS41 is the American standard for cellular mobile phones. The standard isn't as developed as GSM. It allows update locations cancel location are some of the operations.

TCAP

TCAP is the top layer of the SS7 protocol. It stands for transaction capability application protocol It's used in the communications industry.

Vi

Vi is a tool used to edit text files. There is a big learning curve as all the commands are groups of keys i.e. no menu.

Aldiscon

Aldiscon are involved in the telecommunication industry. There main product is short messaging. other products in HLR, VLR, cellular, GSM, IS41. It is an Irish company.

HLR

The HLR is the central part of the GSM network. It is a database of subscribers. It holds the location of the subscriber and what services that subscriber is allowed. There is a lot of network traffic to and from the HLR. Most of the network traffic involves update locations. As a subscriber moves from one location to the next the mobile phones sends a message to the HLR. The HLR then sends messages back to the mobile phone and maybe to the VLR. The HLR can also handle Short messaging. HLR stands for Home Location Register.

Time Series

A time series is a collection of observations made sequentially in time. A time series is said to be continuous when observations are made continuously in time. Also can have a discrete set of values continuous can be made discrete by digitising or aggregation.

Stationarity of a time Series

A series is called (strictly0 stationary if the joint probability distribution associated with m observations made at any set of times 1+k, 2+k,...m+k is unaffected by any change in the value of k. A series that is not automatically stationary can be rendered stationary by repeated differencing where the differencing operator equals 1-B.

Box-Jenkins Univariate Modelling

The Box-Jenkins approach is an iterative approach to time series model building for forecasting. Each of these series will be modelled using three stages of the ARMA model building process. These stages are: (i) identification, (ii)estimation, and (iii) diagnosis. Because the class of possible models is too extensive to be conveniently fitted directly to data, rough methods for identifying subclasses of these models are developed. Such methods of model identification employ data and knowledge of the system to suggest an appropriate parsimonious sub-class of models which may be tentatively entertained. In addition, the identification process can be used to yield rough preliminary estimates of the parameters of the model.

Forecasting

All quantitative forecasting methods make use of the following basic strategy. Past data are analysed in order to identify a pattern that can be used to describe them. then this pattern is extrapolated, or extended into the future in order to make forecasts. this strategy rests on the assumption that the pattern that has been identified will continue into the future. The technique cannot be expected to give good predictions unless this assumption is true. Short term forecasts are more reliable than long term. Time series models generate predictions that are based solely on the historical pattern of the variable to be forecast.

Spectral Analysis

Spectral analysis only applies to phenomena which possess a wave-like structure i.e. can be represented as a composition of sine and cosine waves with different amplitudes and frequencies. In fact we may usefully apply spectral analysis to any type of process which fluctuates in some form, but which exhibits a kind of stability i.e. is stationary. Spectral analysis includes many useful methods based on the Fourier analysis of time series. The most fundamental spectral problem is the estimation of the spectral density function. Spectral analysis can provide an intuitive frequency-based description of the time series under consideration and indicate interesting features such as long memory, presence of high frequency variation and cyclical behaviour.

Linear Predictive Coding

LPC (Linear Predictive Coding) is one of the most widely used techniques for speech compression today. The method is based on the decomposition of the speech production process into an excitation model and a vocal tract model. The vocal tract is modelled as an all-pole filter. With traditional LPC (as opposed to MPLPC (Multipulse Linear Predictive Analysis)) produces low quality but intelligible speech.

MPLPC

MPLPC (Multipulse Linear Predictive Coding) is a variation on traditional LPC which produces by an analysis-by-synthesis technique much higher quality speech. Single pulses are added to the excitation signal such that a perceptually significant error is minimised.

Diphone

Given the difficulty in modelling phone-to-phone transitions researchers have turned to a new synthesis unit which circumvents this problem Diphones consist of two half-phones, the transition being contained within the unit itself. Unfortunately the number of required synthesis units grows from 40 to 1600 i.e. 40*40. Since spectrally similar units are being joined at diphone boundaries smoothing rules should be simple.

Xpilot

Xpilot is a multi-player space game. Run on a UNIX platform and across a network. (You can play with people in different countries) it's great fun. Weapons at a player's disposal include smart-missiles, lasers and mines.

Sparc Audio

Early SPARC stations supported only one type of audio data: 8-bit mu-law encoded and sampled at 8000Hz. This gives telephone quality speech. Since then SPARC stations 5 and 6 have a Crystal Semiconductor audio device which handles may more formats: PCM, A-law for example. Other sampling rates are also supported.

Prosody

Prosodic features are stress, duration and intonation They are variations of the acoustic signal whose domains are beyond the boundaries of each individual phonetic segment. Listeners perceive prosodic features by a complex combination of acoustic correlates such as intensity, duration and fundamental frequency.

Speech Processing

There are 2 types of broad class synthesiser, which are articulatory analog and acoustic domains based on mimicking how humans speak. There are also 2 control methods available through analysing existing speech sounds or through a set of rules and pre-programmed sounds. DECtalk is an example of the former and sounds more natural and intelligent than acoustic domain synthesis by rule.

Text-to-Speech System

In developing a text-to-speech system(TTS) two distinct modules need to be assessed, the linguistic mode and the signal processing mode. In the linguistic mode text pre-processing, morphological analysis and grapheme to phoneme conversion needs to be accomplished. Several problematic areas need to be solved like that of numbers, and proper names. This string of phonemes is then sent to a synthesiser which can process it in several different ways. One of the more successful methods to date has been that of diphone concatenation.

Intonation

An algorithm for generating synthetic intonation was applied to a sample of text. The sample is given to expert listeners to allow them compare synthetic intonation to that of real human speakers intonation. The results showed that intonation choices in reading aloud appear to be guided by synthetic and phonological factors. Especially in respect to declination , a distinction should be made between intonation phrases in sentence initial, medial and final position.

Format Coding

This is phoneme coding where each allophone must be explicitly choose by the user This allows for unlimited, multilingual vocabulary Instead of words the lexicon consists of word sounds made up of phonemes and their allophone variants

Localisation

Computer companies wishing to maximise their profit margin try and target as many markets as possible In order to attain this goal foreign markets msu tbe exploited For example an US company may attempt to target France, Germany, Italy and Spain Therefore the product must be translated into these target languages Translation does not describe this process correctly the product must be localised to make it appear as if it was created and developed for the target market in question This process is part of what is called the localisation industry

Cascading Style Sheets

HTML is a tagged based method for describing the structure of a document , eg document title , text body and links to other documents HTML is not meant to describe how text should be formatted CSS are a method to allow this An author creates various styles which can be applied to an entire web site, to numerous documents or to a single document The stylesheet can be embedded in the document or can be linked in More than one stylesheet can be included in a document

Diablo

Diablo is a Role playing 3D game it has single player or multiplayer options It is based on the concept of medieval campaigns laced with magic where the user plays a warrior, a sorcerer or a female the player is given a mission to complete involving killing demons, skeletons, the undead, scary creatures and evil men the player must increase their experience level by collecting information, more killings, finding better weapons and better magic it is a great game for all those RPG lovers

Dynamic Fonts

Fonts specified in a document can only be viewed correctly if they are present on the users machine Otherwise the text is displayed as normal To overcome this problem Dynamic fonts have been created the web author creates their page using certain fonts These fonts are then stored in a separate file This fonts file is referenced by the web page and is downloaded by the user this guarantees that the web page will have the correct fonts as specified by the author Only Web browsers that support dynamic fonts will carry out this operation, otherwise the text is displayed as normal

HTMLHelp

HTMLHelp ıs a Mıcrosoft help authorıng tool that allows the user to create HTML based help the help system can be run on a standalone machıne or across the Web. It ıs platform ındependent but currently only works wıth Mıcrosoft Internet Explorer

Aster

Aster ıs a system developed by Dr T B Raman to produce spoken output from athematıcal formulae He used the LaTeX markup language to derıve hıs output, and relayed the output to the user usıng a dec-talk synthesıser

Maths Project

The maths project ıs a system developed at the unıversıty of York to convey mathematıcal to blınd users usıng synthetıc speech It uses prosodıc queues to enhance the equatıons, and also uses "earcons" to gıve overvıews of the mathematıcal expressıons

Prosody in Maths

Prosody ıs used ın the translatıon of mathematıcal formulae ınto synthetıc speech The use of prosody enhances the equatıons to produce meanıngful spoken output, ınstead of output ın a monotone

C++ V Java

C++ and Java are sımılar ın that they both use the Object Orıented method of program desıgn However, they are dıfferent ın the way the handle both garbage collectıon, and dırect memory access Where C++ uses the poınter construct, Java handles thıs automatıcally, abstractıng the user from thıs aspect of programmıng

Dec-Talk Express

The Dec-talk Express ıs a devıce produced by the Dıgıtal Corporatıon whıch takes ASCII text as ınput, and produces spoken output It has a number of ın-buılt voıces whıch the user can adjust, and ıs hıghly programmable

Dynamic Programming

Dynamıc programmıng ıs a method for solvıng decısıon problems over large state spaces It was devısed ın the 1960's by Bellman The method devısed by Bellman ıs often referred to as the value ıteratıon method

Bınomial Distribution

The bınomıal dıstrıbutıon ıs the approprıate dıstrıbutıon when a number of Bernoullı traıls are carrıed out Where the sample space ıs large the bınomial distribution can be approxımated usıng the Poısson dıstrıbutıon.

Bernoulli Trails

Bernoulli trails are independent trails. The outcome of these trails can only take on one of two values, success or failure. The probability of success must be constant.

Chi-Square

The Chi-square distribution is a skewed probability distribution. It can be used for tests of independence and goodness of fit tests. It can also be used for hypothesis tests on population standard deviations.

Team Talk

Team talk is a web site dedicated to English Premiership football. My team is Carlisle United.

Localisation

Localisation in the process of integrating the whole of a product into the language and culture of the target markets to meet their specific needs. It involves all components of a software product including its functionality, manuals, on-screen text, marketing literature and graphics.

Internationalisation

Internationalisation is the behind-the-scenes work by software engineers to create a system or application software independent of or transparent to natural language. It includes generic coding and design issues, e.g. keeping user interface strings separate from the rest of the code to aid translation.

SLIG

The Software Localisation Interest Group was set up to bring together all parties interested in the localisation industry, such as software developers, translators, technical writers and language researchers. There are four working groups currently active: tools, terminology, training and education and multimedia. These groups hold regular meeting hosted by SLIG members. It members include companies including Microsoft, ITP, Symantec.

Electronic Documentation

Electronic Documentation is any documentation created using a software package and stored in a computer file. Desktop Publishing (DTP) and Word Processing are the most popular ways of creating electronic documentation. Most electronic documents use some form of text markup. Specific markup (such as RTF from Microsoft Word, or MIF from FrameMaker) identifies the appearance or format of the document. Generic markup (such as HTML or SGML) store the structure of the elements of the document.

Scrabble

Scrabble is a Windows based word game that allows the user(s) to play against each other or the computer The computer uses Artificial Intelligence (AI) techniques to, evaluate its word to play The user must use the mouse to position their words, the keyboard cannot be used We set the timer to play an exciting game of 30 second scrabble

Siggraph 97

Siggraph 97 is being held in Los Angeles from the third to the eight of August Siggraph is the best place to find out what is going on in the computer graphics industry In the course you will learn about the fundamental principles of every computer graphics technique The conference and exhibition is attended by over 40,00 people over the week long event

NBA.COM

Nba com is the official web site of the National basketball association The site gives up to the minute scores and results during the NBA season in its news and features pages teams and players are featured during the season Each week the top ten plays of the week are placed in avi format in the nba com site to be downloaded and viewed

Movie Computer Graphics Special Effects

In the late 1980's the film Terminator 2 Judgement Day heralded a break through in computer graphics effects The liquid terminator in the movie was a major improvement over previous graphics effects in the movie One of the leading special effects companies is George Lucas' Industrial Light and magic ILM ILM has allowed Tom Hanks to shake hands with a long since dead president Eisenhower in Forest Gump and brought dinosaurs back to life in Jurassic park With such impressive leaps in computer graphics special effects in the near future real live actors may not be necessary

Games Console Versus the PC

recently 32bit game consoles such as the Sony Playstation and the Sega Saturn and the 64 bit nintendo64 have been able to easily beat the Pentium PC in so far as graphics detail and fluidity of update This is because these game consoles have special chips which are specifically designed to handle the operations needed to render 3d objects on the screen, where as the PC the main processor does all the work This is a bout to change with the introduction of affordable special 3D graphics board that are slotted into the PC These graphics boards will bring the PC up to and beyond the performance of the dedicated games consoles

Motion Capture

Motion capture is technique used in movie special effects and computer games to produce life like movement of computer graphics characters on screen To record the movement of persons, small coloured discs are placed on the persons joints and cameras from different positions record the positions of the discs throughout the sequence Using computer animation software a creature can be built around the disc by placing a joint of the creature at each disc position As the discs move through the sequence so do the virtual joints, thus producing life like animation

Appendix B - Experiment Results

Results from Apprentice Authors

	Links from Node	Links to Node	Time Taken	Links Viewed	Compactness	Avg. Links	High Rank	Low Rank	Word Count
HPSG	4	0	5	7	0.217	1.57	134.01	33.92	64
WordNet	4	0	9	6	0.150	1.87	172.59	51.56	60
Machine Translation	0	0	8	20	0.000	0.00	121.61	64.79	72
VerbMobil	5	4	6	8	0.055	1.04	140.64	44.96	54
Case Frames	3	2	7	5	0.215	1.43	175.2	48.45	73
IS41	6	6	6	0	0.029	1.60	67.86	23.78	27
TCAP	3	3	6	1	0.125	1.12	52.83	28.41	22
VI	1	1	3	0	0.085	1.44	66.3	28.71	26
Aldiscon	2	2	4	0	0.430	1.69	95.7	19.02	27
HLR	1	1	6	0	0.330	1.20	204.19	59.22	100
Time Series	3	0	9	8	0.040	1.06	86.69	47.99	48
Stationarity of a time series	3	3	7	7	0.045	1.02	96.54	43.16	55
Box-Jenkins Univariate Modelling	2	0	12	8	0.081	1.22	209.56	100.7	130
Forecasting	3	3	11	10	0.123	1.18	322.57	82.38	110
Spectral Analysis	4	2	15	15	0.096	1.27	234.64	79.79	145
Linear Predictive Coding	0	0	9	1	0.000	0.00	142.31	52.15	72
MPLPC	2	2	8	3	0.305	0.75	96.07	30.75	40
Diphone	0	0	8	1	0.000	0.00	107.81	38.25	66
Xpilot	2	0	8	3	0.095	1.45	70.36	29.32	41
Spark Audio	3	0	10	3	0.072	1.10	471.28	71.03	52
Prosody	1	0	9	8	0.496	1.90	96.07	30.75	45
Speech Processing	2	0	9	9	0.136	0.94	113.14	42.12	62
Text-to-Speech System	1	1	11	6	0.247	1.13	83.21	39.19	91

Intonation	0	0	12
Formant Coding	3	3	10
Localisation	2	0	8
Cascading Style Sheets	3	2	8
Diablo	1	0	7
Dynamic Fonts	4	0	7
HTMLHelp	1	0	6
Aster	4	2	9
Maths Project	2	2	7
Prosody in Maths	3	3	8
C++ V Java	3	1	5
Dec-Talk Express	4	2	6
Dynamic Programming	4	2	6
Binomial Distribution	3	1	6
Bernoulli Trails	1	0	5
Chi-Square	3	3	6
Team Talk	1	0	5
Localisation	2	0	3
Internationalisation	1	1	4
SLIG	2	1	5
Electronic Documentation	10	3	7
Scrabble	1	0	4
Siggraph 97	6	6	5
NBA.COM	0	0	4
Movie Computer Graphics Special Effects	2	1	6
Games Console Versus the PC	5	0	6
Motion Capture	3	3	6

5	0.000	0.00	96.98	44.36	81
8	0.251	1.55	108.25	28.94	39
1	0.247	2.01	86.02	58.56	89
2	0.287	2.27	105.28	51.87	84
4	0.097	1.00	113.18	57.81	92
0	0.231	2.45	281.94	153.93	103
0	0.186	1.94	113.1	42.4	39
13	0.216	1.60	62.25	28.72	37
10	0.221	1.56	143.41	57.6	41
9	0.248	1.52	55.67	29.22	35
12	0.084	1.25	284.7	86.68	56
6	0.081	1.93	107.93	64.82	38
5	0.100	1.07	73.75	37.94	34
1	0.154	0.93	179.6	26.67	32
2	0.249	0.90	102.67	33.25	28
3	0.246	1.27	308.64	28.49	35
0	0.157	1.26	89.1	52.74	16
2	0.079	1.68	90.19	38.41	43
6	0.516	0.94	94.85	62.32	47
7	0.121	1.04	134.52	65.35	69
5	0.205	1.61	218.03	83.4	75
3	0.048	1.16	132.72	65.5	59
1	0.090	1.51	68.44	34.83	75
0	0.000	0.00	92.09	38.6	62
3	0.136	1.59	239.88	133.76	100
2	0.148	2.13	375.08	85.22	114
3	0.052	1.45	215.17	95.01	89

Results from Expert Authors

	Links from Node
HPSG	3
WordNet	6
Machine Translation	0
VerbMobil	7
Case Frames	9
IS41	4
TCAP	11
VI	4
Aldiscon	4
HLR	2
Time Series	3
Stationarity of a time series	3
Box-Jenkins Univariate Modelling	2
Forecasting	3
Spectral Analysis	4
Linear Predictive Coding	1
MPLPC	6
Diphone	2
Xpilot	5
Spark Audio	8
Prosody	4
Speech Processing	1
Text-to-Speech System	1
Intonation	1
Formant Coding	2

Links to Node	Nodes Viewed	Compactness	Avg. links
1	8	0.230	1.620
0	10	0.157	1.692
0	5	0	0
3	11	0.164	1.372
5	8	0.267	1.972
3	13	0.473	2.647
3	12	0.316	2.350
1	8	0.147	1.928
0	8	0.188	1.522
0	6	0.240	1.478
1	10	0.036	1.125
1	7	0.115	1.294
0	10	0.080	1.220
1	5	0.070	1.278
2	4	0.089	1.244
0	8	0.187	1.111
2	9	0.180	1.615
0	8	0.073	1.111
3	10	0.107	1.625
1	12	0.098	1.320
1	8	0.109	1.377
0	10	0.388	0.750
1	11	0.216	1.153
0	4	0.280	0.875
1	10	0.266	0.833

Localisation	3	1
Cascading Style Sheets	4	2
Diablo	5	1
Dynamic Fonts	5	3
HTMLHelp	2	0
Aster	4	1
Maths Project	2	1
Prosody in Maths	6	2
C++ V Java	6	3
Dec-Talk Express	5	2
Dynamic Programming	6	3
Binomial Distribution	3	1
Bernoulli Trails	4	2
Chi-Square	6	4
Team Talk	3	0
Localisation	4	1
Internationalisation	1	0
SLIG	3	2
Electronic Documentation	9	4
Scrabble	1	0
Siggraph 97	10	2
NBA COM	4	0
Movie Computer Graphics Special Effects	1	1
Games Console Versus the PC	4	2
Motion Capture	7	2

7	0 485	12 076
11	0 287	2 472
14	0 083	1 911
8	0 189	2 114
4	0 218	2 214
11	0 167	1 671
7	0 192	1 200
9	0 232	1 532
13	0 119	1 815
11	0 069	1 611
14	0 245	2 147
12	0 154	0 928
9	0 26	1 392
11	0 491	2
4	0 237	2 345
10	0 087	1 666
10	0 172	1 137
8	0 151	1 129
8	0 213	1 533
11	0 048	1 161
12	0 187	1 639
5	0 256	2 776
9	0 176	1 257
9	0 157	2 14
13	0 115	2 01