Navigating the World-Wide-Web

Mark Levene and Richard Wheeldon

School of Computer Science and Information Systems
Birkbeck University of London
Malet Street, London, WC1E 7HX, U.K.
email: {mark,richard}@dcs.bbk.ac.uk

Summary. Navigation (colloquially known as "surfing") is the activity of following links and browsing web pages. This is a time intensive activity engaging all web users seeking information. We often get "lost in hyperspace" when we lose the context in which we are browsing, giving rise to the infamous *navigation problem*. So, in this age of information overload we need navigational assistance to help us find our way through the tangled web of pages and links. Search engines partially solve this problem by locating relevant documents and finding "good" starting points for navigation, but more navigational assistance is needed to guide users through and between web sites.

We present a model for navigation which has enabled us to develop several tools and algorithms for helping users with their navigation difficulties. Our point of view is that to help understand how users navigate the web topology we can attach probabilities to links giving rise to a probabilistic automaton, which can also be viewed as a Markov chain. These probabilities have two interpretations, namely, they can denote the proportion of times a user (or a group of users) followed a link, or alternatively they can denote the relevance (or expected utility) of following a link.

We present a new metric for measuring the navigational potential of a web page, called the *potential gain*. This metric is used to find "good" starting point for an algorithm we describe in detail, called the *Best Trail* algorithm, which semi-automates web navigation by deriving relevant trails given a user query. We also present techniques we have developed in the area of web usage mining, detailing our algorithms for analysing records of trails that emerge from either an individual user or a group of users through navigation within the web graph over a period a time.

We also give historical and current overviews of attempts to address the navigation problem, and review the various navigation tools available to the web "surfer". Finally, we give a brief introduction to navigating within the mobile web, and discuss new navigation techniques that have arisen from viewing the web as a social network.

1 Introduction

We are living in an era of information overload, where finding relevant content is becoming increasingly difficult. The World-Wide-Web (the web) collates a massive amount of online information of varying quality, some of which can found through search engines and some which can only be traced through intensive browsing of web pages coupled with navigation via link following. As an indication of the massive volume of the web, a recent estimate of its size given by Murray of Cyveillance during 2000 [49] was 2.1 billion pages, and more recently towards the end of 2001, Google reported that their index contains over 3 billion web documents; see www.google.com/press/pressrel/3billion.html. We expect that the actual number of web pages is much higher than 3 billion, as each search engine covers only a fraction of the totality of accessible web pages [36]. Moreover, this estimate does not include *deep web* data contained in databases which are not accessible to search engines [5].

A user seeking information on the web will normally iterate through the following steps:

(1) *Query* − the user submits a query to a search engine specifying his/her goal; normally a query consists of one or more input keywords.
(2) *Selection* − the user selects one the returned links from the ranked list of pages presented by the search engine, and browses that web page displayed as a result of clicking the link.
(3) *Navigation* (colloquially known as "surfing") − the user initiates a navigation session, which involves following links highlighted by link text and browsing the web pages displayed.
(4) *Query modification* − a navigation session may be interrupted for the purpose of query modification, when the user decides to reformulate the original query and resubmit it to the search engine. In this case the user *returns* to step (1).

In other cases the user may go directly to a home page of a web site or some other starting point, and start navigation by iterating through steps (2) and (3).

Behind each query to a global search engine there is an information need, which according to Broder [11] can be classified into three types:

(1) Informational − when the intent is to acquire some information presumed to be present in one or more web pages.
(2) Navigational − when the intent is to find a particular site from which the user can start "surfing".
(3) Transactional − when the intent is to perform some activity which is mediated by a web site, for example online shopping.

Depending on the specification of the query and the quality of the search engine, the user issuing an informational or transactional query may satisfy

his/her information need with minimal navigation and query modification. For example, if the user is interested in a particular paper by a given author, he/she may find the paper directly through an informational query such as "bush as we may think". As an example of a transactional query, the keywords "bargains online bookshop", would point the user towards online bookshops where further interaction with the user will take place within the online bookshop rather than with the global search engine. In this case the user will probably interact with a local search engine and may have to navigate within the web site to find the information needed. As an example of a navigational query, the user may wish find Tim Berners-Lee's keynote address in the WWW2002 conference by issuing the query, "www2002 conference", and then following the appropriate links to the desired information. In this case the effort the user needs to expend to find the keynote address depends on two factors: (1) on the navigational assistance provided within the WWW2002 web site and, (2) the navigational competence of the user in picking up the available navigation cues within the WWW2002 web site. Although Broder's taxonomy was devised with global search in mind, it is also relevant for search within medium to large web sites, with the user's starting point often being the home page of the site.

Here we will limit ourselves to step (3) of the information seeking process, i.e. the navigation step, which is not directly supported by search engines. Although search engines can provide a user with "good" web pages for starting a navigation session, once the user is "surfing" the search engine provides *no* additional support to assist the user in realising his/her goal. If the navigation process is unsuccessful the user may opt to modify the query through step (4) or go back to a previous page and choose a different link to follow.

It may be argued that it is not within the scope of search engines to provide navigational assistance and that firstly, the browser should supply some navigation tools, and secondly, web sites should aid users navigating within them. To some degree this is a valid argument (see Section 5) but we take the view that search engines can give help by providing contextual information in the form of trails, see Section 7. The notion of a *trail*, inspired by Bush's vision [14], is defined as a sequence of links which may be followed by the user at step (3) of the information seeking process. Thus navigation is the process enacted when following a trail of information, where the value of the trail as a whole is, in general, greater than the individual values of pages on the trail. We believe that trails should be first-class objects in the web, in addition to pages and links which are considered to be the basic building blocks of any hypertext [51].

Providing navigational support in the web, and, in general, in any hypertext, is important due to the infamous *navigation problem*, whereby users become "lost in hyperspace" [51], meaning that they become disoriented in terms of

- where they are relative to prominent pages such as the home page,
- what they should do next in order to achieve their goal, and
- how they can return to a previously browsed page.

In other words, web "surfers" or, more generally, hypertext readers may lose the context in which they are browsing and need assistance in finding their way. In this critical review we will concentrate on ways of tackling the navigation problem, mainly within the web which is the definitive, existing global hypertext.

Our starting point will be the presentation of a formal model of hypertext and the web, which will provide us with a useful reference point; see Section 2. We will then review the field from an historical perspective starting from Bush's seminal work on the *memex* [14], through Nelson's *Xanadu* project and his vision of a global hypertext [50], culminating in the current day web invented by Berners-Lee, the current director of the World-Wide-Web Consortium which is most influential in directing for the evolution of the web [6]. Following that we will review the navigation problem from a hypertext perspective, leading to recent web specific proposals for tackling the problem; see Section 4. We will not ignore recent developments relating to the *mobile web* [62], where access to the web is becoming pervasive through a wide variety of input modalities and computing devices such as voice via mobile phones and pens via handheld PCs; see Section 9. In Section 10 we will briefly review recent work, which shows that viewing the web as a social network leads to new techniques for tackling the navigation problem. Finally, in Section 11 we list some open problems that warrant further investigation.

We do not attempt to review all recent work in this area as the number of publications in this area is well beyond a single review; we do note that many recent contributions in this area can be found on the web.

Our work in recent years has looked at the navigation problem from two perspectives:

(1) Given a user query, and no other information about the user, is it possible to semi-automate the navigation process? For a search engine the unit of information that is manipulated is a single web page. We investigate the possibility of a trail being the logical unit of information, through the development of a navigation engine where trails are manipulated as first-class objects.
(2) Given a (continuous) log of users navigation sessions, is it possible to provide these users with navigational assistance using web usage mining techniques? Such assistance should be personalised and have the ability to suggest users with relevant links to follow.

We review the following solutions we have been developing for the navigation problem:

(1) The *potential gain*, which is a query independent measure of how "good" a web page is as a starting point for navigation; see Section 6.
(2) The *Best Trail* algorithm [70, 71], which is an algorithm for finding relevant and compact trails given a user query; see Section 7.
(3) Data mining of user navigation patterns [8, 9, 10] based on a novel model of trail records which is suitable both for virtual and physical spaces; see Section 8

2 A Model for Web Navigation

We now introduce the main points of our model via an example. Consider the web topology shown in Figure 1, where each node is annotated with its URL (Unified Resource Locator), U_i, which is the unique address of the page P_i represented by the node. In addition to the URL U_i each node contains the score which is a measure of the relevance of the page P_i to the input query. (We assume that users query represents their information need.) Thus, in the hypertext tradition [51], the web is represented as a labelled directed graph [13], which we refer to as the *web graph*.

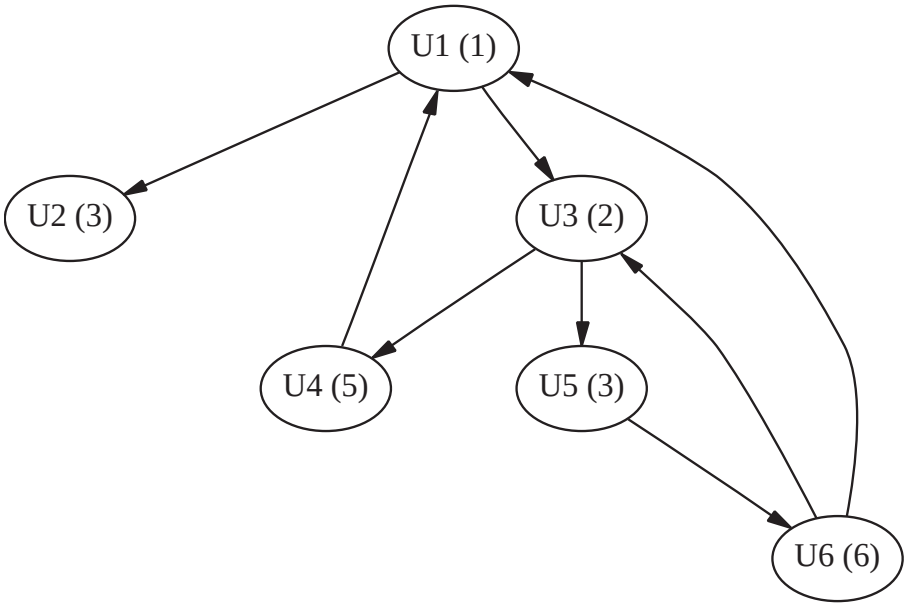

Fig. 1. An example web graph

A *trail* of information through the web graph consists of a sequence of pages visited by the user in a navigation session. For example, with respect to Figure 1 four possible user trails starting from P_1 are:

1) $P_1 \rightarrow P_2$,
2) $P_1 \rightarrow P_3 \rightarrow P_4 \rightarrow P_1 \rightarrow P_2$,
3) $P_1 \rightarrow P_3 \rightarrow P_5 \rightarrow P_6 \rightarrow P_1$ and
4) $P_1 \rightarrow P_3 \rightarrow P_5 \rightarrow P_6 \rightarrow P_3 \rightarrow P_4$.

In our formal model we view the web as a finite automaton called a *Hypertext Finite Automaton* (HFA), whose states are web pages and transitions are links [38]. In a HFA all states can be initial and final, since we allow navigation to start and finish at any page. The state transitions of the HFA occur according to the links of the web graph, namely the state transition from state s_i to state s_j, labelled by symbol (page) P_i, is given by

$$s_i \xrightarrow{P_i} s_j$$

and corresponds to a link from page P_i to page P_j. Our interpretation of this state transition is that a user browsing P_i decides to follow the link leading to page P_j. At the end of the navigation session, after some further state transitions, the user will be in state, say s_k, browsing page P_k.

A *word* that is accepted by a HFA, which we call a *trail* of the HFA, is a sequence of pages

$$P_1 \rightarrow P_2 \rightarrow \ldots \rightarrow P_n$$

which were browsed during a navigation session, starting at page P_1, then following links according to the state transitions of the HFA and ending at page P_n. The *language* accepted by a HFA is the set of trails of the HFA. In other words, the language accepted by a HFA is the set of all possible trails a user could follow, which are consistent with the topology of the web graph.

Let xy denote the concatenation of the words x and y. Then a word y is a *subword* of a word w if $w = xyz$ for some words x and z, and a word w is the *join* of words xy and yz if $w = xyz$ and y is not the empty word. In [39] we provide a characterisation of the set of languages accepted by a HFA, as the subset of regular languages closed under the operations of *subwords* and *join*. This result is intuitive in terms of web navigation since subwords correspond to *subtrails*, and the join of two words corresponds to the join of two navigation trails, where the second trail completes the first one.

We advocate the support of trails as first-class objects, and thus define a keyword-based query language, compatible with the usual input to search engines, where a *trail query* is of the form

$$k_1 \; k_2 \; \ldots \; k_m$$

having $m \geq 1$ keywords.

A trail, T, which is accepted by a HFA *satisfies* a trail query if for all k_i in the query there is a page P_j in T such that k_i is a keyword of P_j. We note that in the special case when the trail has a single web page then all the keywords must be present in this page, complying with the usual semantics of search engine query answering. (We will discuss trail scoring mechanisms in Section 7.) In [38] we show that checking whether a HFA accepts a trail satisfying a trail query is NP-complete. The proof of this result utilises a duality between *propositional linear temporal logic* and a subclass of finite automata. In temporal logic terminology the condition that k_i is a keyword of page P_j is the assertion that "sometimes" k_i, viewed as a condition on P_j, is true. Therein we also defined a more general class of trail queries which supports the additional temporal operators "nexttime" and "finaltime", and more general Boolean conditions. In the context of the web the natural interpretation of "time" is "position" within a given trail. So, "sometimes" refers to a page at some position in the trail, "nexttime" refers to the page at the next position in the trail, and "finaltime" refers to the page at the last position in the trail. In [38] we have shown that only for restricted subclasses of trail queries is the problem of checking, whether a HFA accepts a trail satisfying a trail query, polynomial-time solvable. Such a subclass essentially prescribes a one-step at a time navigation session using the "nexttime" operator. Current navigation practice where links are followed one at a time conforms to this subclass.

These time-complexity results have led us to enrich the semantics of HFA by attaching probabilities (or equivalently weights) to state transitions resulting in *Hypertext Probabilistic Automata* (HPA) [39]. The transition probabilities in our model can have two interpretations. Firstly they can denote the proportion of times a user (or a group of users) followed a link, and secondly they can denote the relevance (or expected utility) of following a link. The first interpretation will be developed in Section 8 while the second in Section 7.

We further develop the notion of HPA by viewing them as finite *ergodic Markov chains* [30]. In order to realise this view we may consider the user's home page as an artificial starting point for all navigation sessions and assume that there is a positive probability (however small) of jumping to any other relevant web page. We can thus modify Figure 1 by adding to the graph the user's home page and links from it to all other pages. The probabilities of these links are the initial probabilities of the Markov chain. Moreover, we assume that the user is following links according to the transition probabilities and when completing a navigation session returns to his/her home page. Thus we would further modify Figure 1 by adding links from existing pages to the artificial home page. A probability attached to such a link denotes the probability of concluding the navigation session after visiting a particular web page. The resulting HPA can be seen to be an ergodic Markov chain [40]. The probability of a trail T is thus defined as the product of the initial probability of the first page of the trail together with the transition probabilities of the links in the trail.

As a further example, consider the web graph shown in Figure 2, which shows a fragment of the School of Computer Science and Information Systems (SCSIS) web site; see www.dcs.bbk.ac.uk. Note that the logical organisation of the web site gives rise to many meaningful trails such as

$$\text{SCSIS} \rightarrow \text{Research} \rightarrow \text{Students} \rightarrow \text{Kevin}$$

which is intuitive and easy to understand.

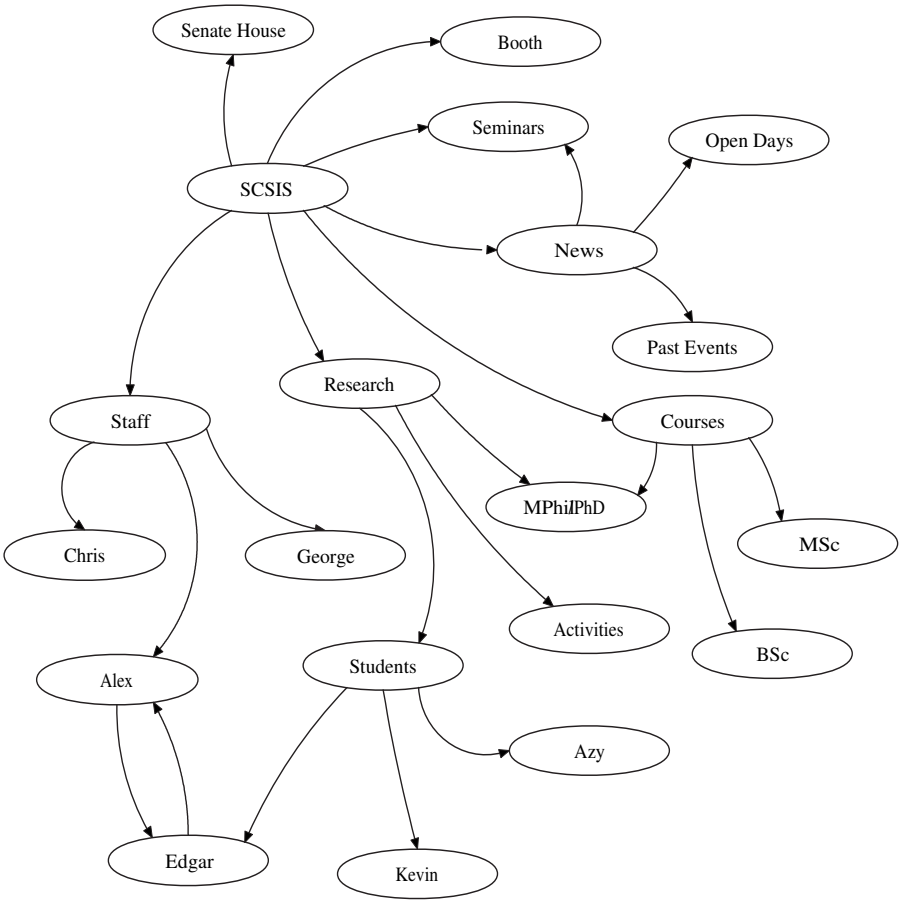

Fig. 2. SCSIS web graph

In our model we can distinguish between the following different types of trails according to their mode of creation:

(1) *authored trails* – trails that have been explicitly defined for a given purpose; for example a guided-tour [68, 63] through the research areas of a Computer Science Department.
(2) *derived trails* – trails that are derived according to specific criteria; for example, as a result of query (see Section 7) or following the hierarchy (or more generally, a suitable ontology) of a web-directory such as Yahoo or the Open Directory.
(3) *emergent trails* – trails that are created via repeated navigation and browsing through a web space (see Section 8). These may be
   a) *personal trails* – trails that arise from an individual's activity, or
   b) *collaborative trails* – trails that arise from a group activity through the space.

3 An Historical Perspective of Navigation in Hypertext

The inspiration for hypertext comes from the *memex* machine proposed by Bush [14] (see [53] for a collection of essays on Bush and his memex). The memex is a "sort of mechanized private file and library" which supports "associative indexing" and allows navigation whereby "any item may be caused at will to select immediately and automatically another". Bush emphasises that "the process of tying two items together is an important thing". By repeating this process of creating links we can form a *trail* which can be traversed by the user, in Bush's words "when numerous items have been thus joined together to form a trail they can be reviewed in turn". The motivation for the memex's support of trails as first-class objects was that the human mind "operates by association" and "in accordance to some intricate web of trails carried out by the cells of the brain".

Bush also envisaged the "new profession of trailblazers" who create new trails for other memex users, thus enabling sharing and exchange of knowledge. The memex was designed as a personal desktop machine, where information is stored locally on the machine. Trigg [68] emphasises that Bush views the activities of creating a new trail and following a trail as being connected. Trails can be authored by trailblazers based on their experience (these are authored trails) and can also be created by memex which records all user navigation sessions (these are emergent trails). In his later writings on the memex, published in [53], Bush revisited and extended the memex concept. In particular, he envisaged that memex could "learn form its own experience" and "refine its trails". By this Bush means that memex collects statistics on the trails that the user follows and "notices" the ones which are most frequently followed. Oren [54] calls this extended version *adaptive memex*, stressing that adaptation means that trails can be constructed dynamically and given semantic justification; for example, by giving these new trails meaningful names.

Engelbart's NLS system [23] was the first working hypertext system., where documents could be linked to other documents and thus groups of people could work collaboratively. The term "hypertext" was coined by Ted Nelson in 1965 (see [50]), who considers "a literature" (such as the scientific literature) to be a *system of interconnected writings*. The process of referring to other connected writings, when reading an article or a document, is that of *following links*. Nelson's vision is that of creating a repository of all the documents that have ever been written thus achieving a universal hypertext. Nelson views his hypertext system, which he calls *Xanadu*, as a network of distributed documents that should be allowed to grow without any size limit and such that users, each corresponding to a node in the network, may link their documents to any other documents in the network. Xanadu can be viewed as a generalised memex system, which is both for private and public use. As with memex, Xanadu remained a vision which was not fully implemented. Nelson's pioneering work in hypertext is materialised to a large degree in the web, since he also views his system as a means of publishing material by making it universally available to a wide network of interconnected users. An interesting feature of Xanadu is its copyright mechanism where reuse of material is done through linking to the portion of material being republished. Berners Lee turned the vision of hypertext into reality by creating the World-Wide-Web as we know it today [6] through the invention of the URL, HTTP and HTML, and more recently through the semantic web and XML.

4 Tackling the Navigation Problem

We have already introduced the navigation problem in Section 1 whereby users "get lost in hyperspace" while they are "surfing", as a result of losing the context in which they are browsing, and are then unsure how to proceed in terms of satisfying their information need. We will now briefly survey some recent suggestions for tackling this problem; we will differ discussion of our proposal via the Best Trail algorithm to Section 7.

Search engine results are not always up-to-date, since they only access information stored in a static index. The idea of *dynamic* search is to fetch web pages online during the search process thus guaranteeing valid and precise information. The downside of such a dynamic approach is that is does not scale. An early dynamic search algorithm called *fish search* [20] uses the metaphor of a school of fish (search agents) foraging (searching) for food (relevant documents). When food is found, the fish reproduce and continue looking for food. In the absence of food or when the water is polluted (poor bandwidth), they die. An improved algorithm is *shark search* [26], which uses the vector-space model [59] to detect relevant documents, and a decay factor between zero and one to reduce the influence of pages which are further away from the starting point. The decay factor can also be viewed as taking into account the cost

to the user of following an additional link [44]. A further improvement in the shark search algorithm is to give priority to anchor text attached to links and its broader textual context, in determining the relevance of documents that may be followed by clicking on a link.

The spread of activation algorithm [56] simulates users' "surfing" patterns when they are foraging for relevant information at some web locality, i.e. a collection of related web pages, in an attempt to understand how "surfers" allocate their time in pursuit of information. The activation network is represented as a graph whose nodes are web pages and where each link has a strength of association attached to it. The strength of association of a link may indicate textual similarity between pages or, alternatively, usage statistics, i.e. the number of users following the link. The activation level of pages is represented by a vector which evolves over time and decays at each time step to reduce the influence of pages according to their distance from an initial set of activated pages. In [56] the algorithm is used to find textually similar pages within a locality and the most frequently browsed pages in a web site of home page visitors.

Related to the above work is that of information foraging by navigation, i.e. "surfing" along links, and picking up proximal cues, such as snippets of text, to assess distant content which is only revealed after following one or more links. The *scent of information* is the perception of the value and cost obtained from these proximal cues representing distant content. In [16] various techniques have been developed to predict information scent based on usage mining analysis, content and link topology. In particular a technique called *web user flow by information scent* has been developed, that simulates agents navigating through a web site, to better understand how users navigate and to predict when their information need is met. In this technique the agents have information needs described by a simple textual description such as "research groups in the computer science department" and, as in the foraging model, the scent at a given page is evaluated by comparing the user's need with the information scent associated with linked pages. The navigation decisions based on the information scent are stochastic, so more agents follow higher scent links. Agents stop "surfing" when they either find a target page or they have expended a given amount of effort.

Web-Watcher [29] is an automated tour guide for the web. Web-Watcher accompanies users as they are browsing and suggests to them relevant links to follow based on its knowledge of users' information needs. At the beginning of the tour each user types in some keywords to describe their information need. Whenever a user follows a link the description of this link, which is initially just its anchor text, is augmented by the user's keywords. Thus links accumulate keywords and Web-Watcher can recommend to a user the the link description that best matches his/her keywords describing the information need, where similarity is measured using the vector-space model. A complimentary learning method used by Web-Watcher is based on reinforcement learning with the

objective of finding trails through the web site that maximise the amount of relevant information encountered when traversing the path. More specifically, the technique is based on Q-learning [65], which chooses a state such that the discounted sum of future rewards is maximised; in this application the states are web pages and the reward is the score returned for the web page the user is browsing, with respect to keywords the user initially specified; again the score is computed using the vector-space model.

An adaptive agents approach to satisfying a users' information need, called Info-Spiders search, was proposed in [47]. Info-Spiders search is an online dynamic approach, as is the shark search algorithm, the motivation being that traditional search is limited by static indexes which are incomplete and often out of date. In this approach a population of agents navigate across web pages and make autonomous decisions about which links to follow next. Initially an information need is specified as a set of keywords along with a set of starting pages. Each agent is then positioned at one of the starting points and given an amount of energy that can be consumed. Each agent situated at a page makes a decision which link to follow based on the current document content, which is used to estimate the relevance of neighbouring pages that can be reached by link traversal. An agent performs its decision using a local feed-forward neural network, which has an input node for each initial keyword and one output node for the relevance. When an agent moves to the next page by following a link its energy level is updated and a reward, in the form of additional energy, is given to the agent if the page reached turns out to be relevant; the agent incurs a cost each time it accesses a document thereby reducing its energy level. As in Web-Watcher, Info-Spiders adapts its behaviour using Q-learning. After each reinforcement step an agent either replicates or dies according to it energy level. Menczer and Belew [47] conducted several experiments to test Info-Spiders and conclude that their algorithm performs better than traditional search algorithms such as breadth-first-search and best-first-search. Info-Spiders is seen to add value to search engines: the search engine can provide "good" starting points and Info-Spiders can reach fresh pages in the neighbourhood of the starting points, which may not have been indexed by the search engine.

In [24] issues of navigation in web topologies are explored in terms of a *viewing graph* which is a small subgraph of the hypertext structure in which the user is currently navigating. Navigability is the property of being able to find the shortest path to a target node from the node currently being browsed by making decisions based solely on local information visible at the current node. This implies that at each node in the viewing graph sufficient information must be available to guide the user to the correct target node via the shortest route. Moreover, the information available at each node must be compact. Under this definition of navigability, navigation on the web is, in general, not effective, due to the fact that local information at nodes is limited. Ways of improving navigation on the web include: organisation of

information into classification hierarchies and the ability to make local decisions through similarity-based measures between nodes of close proximity. Examples of classification hierarchies are Yahoo and the Open Directory and an example of a similarity-based measure, mentioned above, is the similarity of link text to a user query.

5 Navigation Tools

Here we concentrate on navigation aids that help "surfers" orient themselves within the graph topology and find their way.

The *browser* is the component of a hypertext system that helps users search for and inspect the information they are interested in by graphically displaying the relevant parts of the topology and by providing contextual and spatial cues with the use of *orientation tools*. We have taken a wider view of the browser than currently implemented web browsers. In an interview (Wired News, 14 February 2003) Marc Andreessen, one of the founders of Netscape, said:

> "If I had to do it over again, I'd probably show some sort of graphical representation of a tree, so you could see what path you're travelling on and could backtrack. I'd also include thumbnail renderings on the tree to show where you'd been."

A simple orientation tool is the *link marker* which acts as a signpost to tell the user what links can be immediately followed and what links have recently been traversed. In the context of HTML, link text is highlighted and should give accurate information about the page at the other side of the link; so link text such as *click here* is meaningless as it does not convey any information to the user. Another useful orientation tool is the *bookmark*, allowing readers to mark a page to which they can return to on demand when feeling lost [7]. All web browsers provide a bookmark facility, which allows users to view the titles of the web pages on the list, normally through a pull-down menu, and load any one of these pages into the browser. Although bookmarks are useful, it has been found that the rate of page addition is much higher than the rate of page deletion, implying that users have problems in managing their bookmarks [17]. Readers may, in principle, also mark pages which were already visited in order to avoid repetition; such marks are called *bread crumbs* [7]. In the context of web browsers there is a primitive notion of bread crumbs when the colour of a link that has been clicked on is changed.

Maps (or *overview diagrams*) give readers a more global context by displaying to them links which are at a further distance than just one link from the current position. For instance, current web sites often provide a *site map* to give visitors an overview of the contents of the site. Maps can be displayed using a *fisheye-view* that selects information according to its *degree-of-interest*,

which decreases as the page under consideration is further away from the currently browsed page [67]. An ambitious kind of site map, based on the fisheye concept, is provided by the hyperbolic browser [35], which allows the user to dynamically focus on different parts of the web site by using a novel visualisation technique based on hyperbolic geometry; see www.inxight.com.

A set of tools that aid the construction of maps by performing a structural analysis of the graph topology is described in [57]. One such tool is an hierarchical structure that can be imposed on the web graph, where its root is chosen to be a central node whose distance to other nodes is relatively small. Another tool creates semantic clusters within the web graph by identifying strongly connected components of the graph [13]. *Landmark nodes* are prominent nodes within a web site, or more generally, nodes within the subspace the user is browsing through. A simple formulae for discovering landmark nodes in the web graph based on the number of pages that can be reached from a page or that can reach the page when following at most two links was proposed by [48]. Once the landmark nodes are known, the context of the page that the user is currently browsing can be shown by its relationship to nearby landmark nodes. A survey covering additional metrics based on web page content and link analysis can be found in [21].

In [32] the activity of user navigation within a hypertext is compared to the activity of *wayfinding* through a physical space, where wayfinding is defined as the process used to orient and navigate oneself within a space, the overall goal of wayfinding being to transport oneself from one place to another within the space. Both activities, in a virtual space and a physical one, include user tasks such as being aware of one's current location, planning a route to follow and executing the plan. Research into wayfinding in physical spaces is based upon the assumption of the existence of cognitive maps encoding the user's knowledge about the space he/she is navigating through. Such spatial knowledge can be classified into the representations of: place, route and survey knowledge, which concerns the spatial layout of the salient places. Various hypertext tools which aim to help solve the disorientation problem have been developed which are inspired by the spatial metaphor. These include: differentiation of regions, maps, guided-tours, landmark nodes, fisheye-views, history lists, history trees and summary boxes.

An orientation tool that has been developed within the hypertext community is the *guided-tour*, which actively guides users through the space being navigated by suggesting interesting trails that users can follow [68]. One such system, called Walden's paths [63], allows teachers to create annotated trails of web pages for their students to follow and browse. There remains the question of whether trail creation can be automated, at least partially, as is further investigated in Section 7. A dynamically created trail which highlights the path the user has followed within a web site is the *navigation bar* advocated by Nielsen [52]. For example,

SCSIS → Research → Activities → Database and Web Technologies Group

would indicate to the visitor the trail he/she has followed within the School's web site from the home page to the Database and Web Technologies group (see Figure 2). By highlighting the navigation history within the web site the visitor can easily *backtrack* his/her steps to a previously browsed page. This idea can be refined by using a side-bar to highlight links which provide navigation options to the visitor from the current page they are browsing. One can take this a step further and provide dynamic personalised navigation cues; see [43].

Two standard navigation tools provided by web browsers are the *back button* and the *history list*. Another simple navigation aid is the *home* button which allows users to jump to their home page at any time. The back button is a stack-based mechanism allowing the user to retrace their trail one page at a time, and the complimentary forward button returns to the page browsed before the last invocation of the back button. The history list contains the sequence of web pages that were browsed by the user and can be accessed sequentially according to the time browsed or some other criteria such as the most visited page. Current browsers also provide a search facility over the history list. The history list is displayed linearly, although in practice web pages act as branching points, for example users often start navigating from a home page of a site and take different routes according to their information need.

Two studies carried out in 1996 [66] and in 1999 [17] investigated how web "surfers" use the navigation tools provided by Netscape Navigator. Tauscher and Greenberg [66] found that, by far, the most used navigation tool was the back button. Other mechanisms such as the forward button and history list were used infrequently as a percentage of the total number of user actions. They calculated the recurrence rate as

$$\frac{\text{total number of URLs visited} - \text{different number of URLs visited}}{\text{total number of URLs visited}}$$

and discovered that this rate was close to 60%. It was also found that there is about 40% chance that the next page visited is within the last six pages visited. Overall the back button is the dominant mechanism used to revisit pages. It is simple and intuitive to use but inefficient in retrieving distant pages. The authors also found that the history and bookmarks mechanisms are used much less that the back button. In a followup study Cockburn and McKenzie [17] found the recurrence rate to be much higher at about 80% and that almost all users had one or two pages that they revisited far more often than others; this could be for example, their home page or their favourite search engine.

The stack-based behaviour of the back button is incompatible with the navigation history of the user as the following example shows. Suppose the user navigates as follows (see Figure 2):

SCSIS → Staff → *Back to* SCSIS → Research → *Back to* SCSIS

where *Back to SCSIS* indicates the user of the back button. The stack-based semantics means that Staff is *inaccessible* to the user through the back button, since when the user clicked on the back button the first time all pages above it in the stack were removed. Despite the semantics of the back button being misunderstood by many users it is still heavily used. Cockburn et al. [18] conducted a further study evaluating a history-based behaviour of the back button as opposed to the standard stack-based behaviour; so in the previous example, with history-based behaviour, a further click on the back button would bring the user to the Staff page. They concluded that from the users point of view there was no significant difference between the stack and history behaviours. Moreover, the history-based behaviour is inefficient in returning to parent pages from a deeply nested page but for distant navigation tasks it was highly efficient.

An interesting navigation tool is the toolbar recently introduced by Google, which can be installed within Microsoft's Internet Explorer; see http://toolbar.google.com. It provides the user with various options, allowing the user to access the search engine directly from the browser, either searching the web as a whole or just searching within the web site the user is currently at. As long as privacy and security issues are dealt with, we believe in the potential of adding navigation tools to the browser.

Web directories such as Yahoo and the Open Directory organise information according to a categorisation, so for instance *web usability* can be found by navigating the following path in the Open Directory,

Computers → Internet → Web Design and Development → Web Usability

where we find several sub-categories such as *accessibility*, several related categories such as *human-computer interaction* and many web pages which were manually categorised. (In principle, it is possible to automate, or at least semi-automate, the categorisation process but this problem is not within the scope of this chapter; see [61].) Hearst [25] argues that search engines should incorporate category information into their search results to aid navigation within web sites. Moreover, to help cut down the number of possible trails a visitor can follow, the user interface as well as the site's structure should reflect the tasks that the visitor is attempting to accomplish. Hearst points to the potential use of category meta-data to help achieve a better integration between the user's information need and the site's structure by dynamically determining the appropriate categories for a given query.

6 The Navigation Potential of a Web Page

Although, as far as we know, web search engines weight home pages higher than other pages, they do *not* have a general mechanism to take into consideration the navigation potential of web pages. Our aim in this section is to

propose a metric for finding "good" starting points for web navigation, which is independent of the user query. Once we have available such a metric we can weight this information into the user query so that the starting points will be both relevant and have navigational utility. From now on we will refer to the navigability measure of a web page as its *potential gain*. We note that the application that initially led us to investigate the potential gain was to provide starting points for the search and navigation algorithm described in the next section, but we believe that this notion has wider applicability within the general context of navigation tools.

In the following let the G represent the web graph, G having a set of pages (or URLs identifying the pages) N and a set of links E. Starting URLs should satisfy the following criteria:

(1) They should be *relevant*, i.e. their score with respect to the user's goal should be high.
(2) They should be *central* [13] in the sense that their distance to other pages is minimal.
(3) They should be *connected* in the sense that they are able to reach a maximum number other pages. (If G is strongly connected, then this clause is redundant.)

Now, let $links(U_i)$ be a function that returns the collection of links outgoing from U_i. Algorithm 1, given below, computes the potential gain of all the nodes in the node set, N, of a web graph G. It has two additional parameters: (i) MAX defining the maximal depth in the breadth-first traversal of the web graph, and (ii) $\delta(d)$ which is a monotonically decreasing function of the depth d. Two reasonable such functions are the reciprocal function, $1/d$, and the discounting function γ^{d-1}, where $0 < \gamma < 1$. The justification for these functions is that the utility of a page diminishes with the distance of the page from the starting point. This assumption is consistent with experiments carried out on web data sets [28, 37].

The algorithm also involves a constant C between 0 and 1, which is the lower bound potential gain of any pages in N; we will conveniently take C to be 1. The algorithms outputs an array, PG, where PG$[U_i]$ is the potential gain of the URL U_i in N computed to depth $MAX \geq 1$.

Algorithm 1 (Potential_Gain(G, MAX, δ))
1. **begin**
2. **for each** URL U_i *in* N
3. PG$[U_i] \leftarrow C$;
4. prev_count$[U_i] \leftarrow 1$;
5. **end for**
6. **for** $d = 1$ *to* MAX
7. **for each** URL U_i *in* N
8. cur_count$[U_i] = \sum_{U_k in\ links(U_i)}$ prev_count$[U_k]$;
9. PG$[U_i] \leftarrow$ PG$[U_i] + \delta(d) \cdot$ cur_count$[U_i]$;
10. **end for**
11. **for each** URL U_i *in* N
12. prev_count$[U_i] \leftarrow$ cur_count$[U_i]$;
13. **end for**
14. **end for**
15. **return** PG;
16. **end.**

We note that in practice we will need to normalise the potential gain values as, in principle, they may increase without bound. For the discounting function we can guarantee convergence as MAX tend to infinity, if the product of γ and the maximum out-degree of a web page is less than one. On the other hand, for the reciprocal function no such convergence is guaranteed. The potential gain of a starting web page can be seen to depend on the number of trails out-going from the web page, where the value of the trails diminish with distance.We further note that Algorithm 1 can be described much more concisely using matrix-vector notation, as:

$$\text{cur\_count} = G \cdot \text{cur\_count}$$
$$\text{PG} = \text{PG} + \big(\delta(d) \cdot \text{cur\_count}\big)$$

where the above equations are iterated MAX times.

A complementary metric to the potential gain is the *gain rank*, which measures the likelihood of navigating to a given web page. Its computation can be obtained by replacing in Algorithm 1, $links(U_i)$ by $inlinks(U_i)$, where $inlinks(U_i)$ is a function that returns the collection of links going into U_i. It would be interesting to compare the gain rank with Google's PageRank metric [55].

In Table 1 we show the unnormalised potential gain and gain rank values computed to depth $MAX = 100$, of the example web graph shown in Figure 2, using the reciprocal function in the calculation. It can be seen that the home page of the web graph, i.e. SCSIS, has the highest potential gain within the web graph, followed by the pages: Research, Students and Staff. On the other hand, the gain rank of the pages Alex and Edgar have the highest gain rank

within the web graph indicating that they are reachable through more trails than other pages.

URL	Title	Out-degree	In-degree	Potential Gain	Gain Rank
1	SCSIS	7	0	22.0414	1.0000
2	Booth	0	1	1.0000	2.0000
3	Senate House	0	1	1.0000	2.0000
4	Seminars	0	2	1.0000	3.5000
5	News	3	1	4.0000	2.0000
6	Open Days	0	1	1.0000	2.5000
7	Past Events	0	1	1.0000	2.5000
8	Courses	3	1	4.0000	2.0000
9	MSc	0	1	1.0000	2.5000
10	BSc	0	1	1.0000	2.5000
11	MPhil/PhD	0	2	1.0000	4.0000
12	Research	3	1	9.1874	2.0000
13	Activities	0	1	1.0000	2.5000
14	Students	3	1	8.1874	2.5000
15	Azy	0	1	1.0000	2.8333
16	Kevin	0	1	1.0000	2.8333
17	Edgar	1	2	6.1874	17.4999
18	Staff	3	1	8.1874	2.0000
19	Chris	0	1	1.0000	2.5000
20	George	0	1	1.0000	2.5000
21	Alex	1	2	6.1874	17.3117

Table 1. Potential gain and gain rank values for example

7 The Best Trail Algorithm

We present an algorithm for deriving relevant trails from the web graph given a user query. The algorithm, called the *Best Trail*, semi-automates navigation by probabilistically constructing a tree whose most relevant trails are presented to the user. The algorithm is adaptive in the sense that it dynamically searches for the preferred trails by mimicking a user navigation session and scoring the trails as they are expanded according to the topology of the web site.

Prior to presenting the algorithm we discuss the issue of scoring the relevance of trails as first-class objects. As before we let the G represent the web graph, G having a set of pages (or URLs identifying the pages) N and a set of links E. Every link connects two pages: its starting page is called the *anchor* and its finishing page is called the *destination*.

We interpret the score, $\mu(m)$ of a web page m in N, as a measure (or utility) of how relevant m is to the user. In this sense we cater for personalisation, and we would expect μ to be different for distinct users. Another interpretation of μ is that it is query specific and returns the relevance of a page with respect to a given query, where the query is viewed as representing the goal of the navigation session. In this interpretation μ can be viewed as the scoring function of a search engine with respect to a given query. In both interpretations of μ the user initiating a navigation session would like to maximise the relevance (or suitability) of the trail to the query. The relevance of a trail

$$T = U_1 \rightarrow U_2 \rightarrow \ldots \rightarrow U_n$$

is realised by its *score*, which is a function of the scores of the individual web pages of the trail; we denote the scoring function of a trail by ρ. Five reasonable scoring functions for a trail are:

1) The average score of the URLs in the trail, i.e.

$$\rho(T) = avg(T) = \frac{\mu(U_1) + \mu(U_2) + \ldots + \mu(U_n)}{n}.$$

2) The average score of the distinct URLs in the trail, i.e. for the purpose of obtaining the scoring of the trail, each URL in the trail is counted only once. In this case ρ is denoted by avg_dist.
3) The sum of the scores of the distinct URLs in the trail divided by the number of pages in the trail; this scoring function penalises the trail when a URL is visited more than once. In this case ρ is denoted by sum_dist.
4) The sum of the discounted scores of the URLs in the trail, where the discounted score of U_i, the URL in the ith position in the trail, is the score of U_i with respect to the query multiplied by γ^{i-1}, where $0 < \gamma < 1$ is the discount factor. The discounted score of T is given by

$$\rho(T) = discount(T) = \sum_{i=1}^{n} \mu(U_i)\ \gamma^{i-1}.$$

5) The maximal score of all of the URLs in the trail. In this case ρ is denoted my max.

We can also combine scoring functions (3) and (4) by discounting in (3) each URL according to its previous number of occurrences within the trail (This combination of scoring functions, in addition to sum_dist, are the ones we have used in the navigation system we are developing, which is discussed towards the end of this section.) We observe that all the trail scoring functions we have defined are bounded due to the definition of μ and the fact that N is finite, and as a result an important property of the above trail scoring functions is that they define convergent sequences.

We now describe the algorithm assuming one URL as its starting point although, in general, the algorithm will take as input several starting points and compute the best trail for each one of them; see the pseudo-code of Algorithm 2 below. Starting from the initial URL the algorithm follows links from anchor to destination according to the topology of the web, that is, when an out-link exists from the web page identified by its URL then it may be traversed by the algorithm.

The algorithm builds a *navigation tree* whose root node is labelled by the URL of the starting point. Each time a destination URL is chosen a new node is added to the navigation tree and is labelled by the destination URL. Nodes that may be added to the navigation tree as a result of traversing a link that has not yet been followed from an existing node are called *tip nodes* (or simply tips). We also consider the special case when a link has been traversed to a destination URL and the page associated with this URL has no out-links. Nodes in the navigation tree which are labelled by such URLs are called *sinks*, and are also considered to be tip nodes.

At any given stage of the running of the algorithm each tip node in the current state of the navigation tree is considered to be a destination of an anchor of a link to be followed; in the case when the tip node is a sink we can consider the destination to be the sink itself. The algorithm uses a random device to choose a tip node to be added to the navigation tree; in the special case when the tip node is a sink the navigation tree remains unchanged. The weight that is attached to a tip node for the purpose of the probabilistic choice is proportional to the score of the trail induced by the tip node, which is the unique sequence of URLs labelling the nodes in the navigation tree forming a path from the root node of the tree to the tip node under consideration. (The exact formula for calculating the probability of a tip node is given below.) We call the process of adding a tip node to the navigation tree *node extension*. The Best Trail algorithm terminates after a prescribed number of node extensions, each such extension being a single iteration within the algorithm.

The algorithm has two separate stages the first being the *exploration stage* and the second being the *convergence stage*. Each stage comprises a preset number of iterations. During the exploration stage a tip node is chosen with probability purely proportional to the score of the trail that it induces. During the convergence stage we apply a "cooling schedule", where tip nodes which induce trails having higher scores are given exponentially higher weights at each iteration according to the rank of their trails, as determined by their trail scores, and the number of iterations completed so far in the convergence stage. A parameter called the *discrimination factor*, which is a real number strictly between zero and one, determines the convergence rate. When the algorithm terminates the *best trail* is returned, which is the highest ranking trail induced by the tip nodes of the navigation tree. The convergence of the algorithm to the absolute best trail is guaranteed provided the number of iterations in both stages of the algorithm is large enough and the discrimination factor is not

too low. The Best Trail algorithm can be modified so that the discrimination factor decreases dynamically during the convergence stage.

We now define the terminology used for the Best Trail algorithm, given a starting URL, say U, of the current navigation session.

1) The *unfolding* of G is a possibly infinite tree having root U, resulting from traversing G starting from U in such a way that each time a URL is revisited during the traversal of G a duplicate of this URL is added to the node set of the unfolding of G. Thus duplicates of a URL resulting from multiple visits result in distinct nodes in the unfolding of G although their individual scores are identical. (Note that the unfolding of G is finite if and only if G is acyclic.)
2) A *navigation tree* having root U is a finite subtree of the unfolding of G.
3) A *frontier* node of a navigation tree is either
   a) a leaf node in the tree, or
   b) a node, say m, in the tree associated, say with URL U_i, such that the set of URLs associated with the successors of m in the navigation tree is a proper subset of $links(U_i)$, i.e. there is a link in E with anchor U_i which has not yet been traversed during the current navigation session.
4) A *tip* node in a navigation tree is either
   a) a node which is associated with a sink in G, i.e. a node whose associated URL has no successors in G; we call such a tip node a *sink* node (we note that in this case the tip node is also a frontier node which is a leaf), or
   b) a node, say m, associated with a successor in G of one of the URLs associated with a frontier node, say m', in the navigation tree, such that m is the destination of a link that has not yet been traversed from the URL associated with m'.
5) The *score* of a trail induced by a tip node, say t, is the score of the trail which is the unique sequence of URLs labelling the nodes in the navigation tree forming a path from the root node of the tree to t; we overload ρ and denote this score by $\rho(t)$.
6) The *extension* of a navigation tree with a one of its tip nodes, which we call *node extension*, is done according to the following two cases:
   a) if the tip node is a sink node then the navigation tree remains unchanged, otherwise
   b) add a new node and edge to the navigation tree such that the anchor node of this edge is the parent frontier node of this tip node and the destination node is the tip node itself. The new node becomes a frontier node of the extended navigation tree.

Assuming that U_1 is the starting URL of the Best Trail algorithm for the web topology shown in Figure 1, a possible navigation tree after seven node extensions is given in Figure 3. Each node in the navigation tree is annotated

with a unique number and with its URL; the tip nodes of the navigation tree have a shaded boundary. The root of the navigation tree is node 0, which is labelled by the starting URL U_1, and the nodes that were added to the navigation tree as a result of the seven node extensions are numbered from 1 to 7.

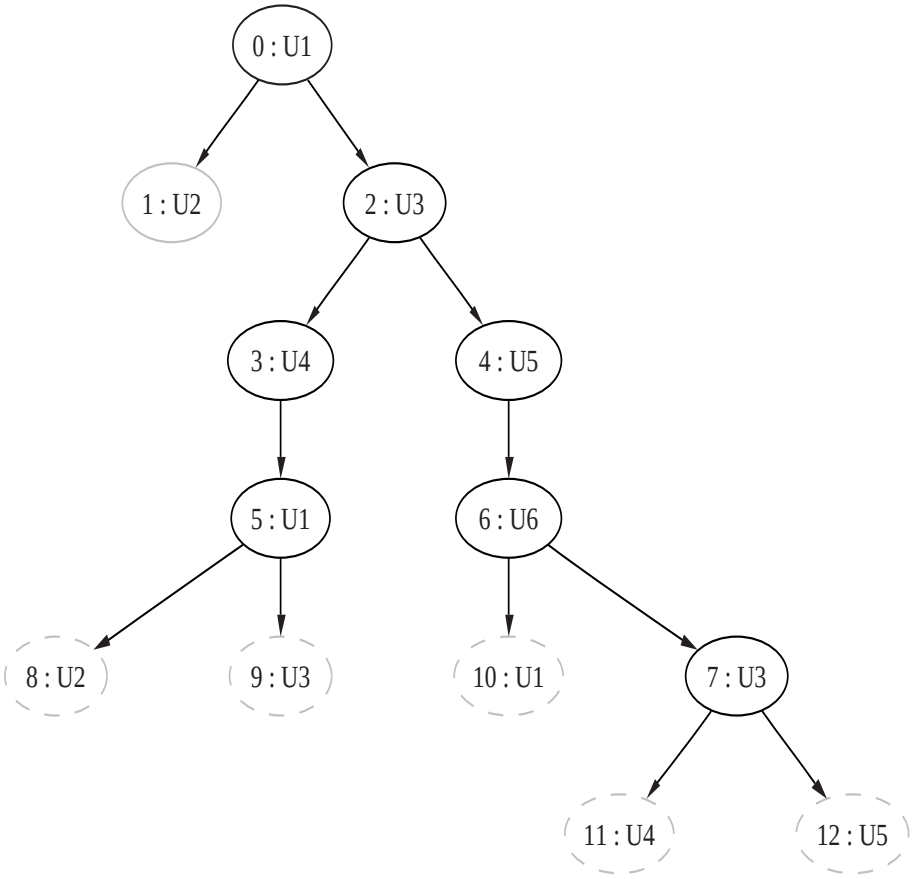

Fig. 3. An example navigation tree

The frontier nodes of the navigation tree are 1, 5, 6 and 7. Node 1 is also a tip node of the navigation tree since it is a sink. Node 5 is the parent of two tip nodes, numbered 8 and 9. Node 6 is the parent of a single tip node, numbered 10. Similarly, node 7 is the parent of two tip nodes, numbered 11 and 12. Table 2 shows the tips, their induced trails and the score of these trails according to the five trail scoring functions we have defined in Section 2. As

can be seen in this example, the trail to tip 11 is the highest scoring trail irrespective of the scoring function used.

TIP	INDUCED TRAIL	*avg*	*avg_dist*	*sum_dist*	*discount* ($\gamma = 0.75$)	*max*
1	U_1, U_2	2.00	2.00	2.00	3.25	3.00
8	U_1, U_3, U_4, U_1, U_2	2.40	2.75	2.20	6.68	5.00
9	U_1, U_3, U_4, U_1, U_3	2.20	2.66	1.60	6.37	5.00
10	U_1, U_3, U_5, U_6, U_1	2.60	3.00	2.40	7.03	6.00
11	$U_1, U_3, U_5, U_6, U_3, U_4$	3.17	3.40	2.83	8.53	6.00
12	$U_1, U_3, U_5, U_6, U_3, U_5$	2.83	3.00	2.00	8.06	6.00

Table 2. The trails induced by the tips and their scores

We now define several auxiliary functions and parameters used in the Best Trail algorithm, given a navigation tree D_i and a tip node t of D_i.

1) The *discrimination factor*, denoted by *df*, is a parameter such that $0 < df < 1$.
   Intuitively, *df* allows us to discriminate between "good" trails and "bad" trails by reducing the influence of trails which perform badly. Thus during the convergence stage "better" trails get assigned exponentially higher probability, taking into account the fact that the longer the history the more influence "better" trails have. This weighting guarantees that, asymptotically, the 'best" trail will eventually have probability one.
2) $I_{explore} \geq 0$ is the number of iterations during the exploration stage of the algorithm.
3) $I_{converge} \geq 1$ is the number of iterations during the convergence stage of the algorithm.
4) The *rank* of the trail induced by a tip node t_k of D_i, denoted by $\tau(t_k)$, is the position of $\rho(t_k)$ within the set of scores of the trails induced by the tip nodes $t_1, t_2, \ldots, t_n$, when the scores are arranged in descending order of magnitude and duplicate scores are removed.
5) The *probability of a tip node*, t of D_i, where α is either 1 or *df* and j denotes either an exploration or convergence step, is denoted by $P(D_i, t, \alpha, j)$, and given by
$$P(D_i, t, \alpha, j) = \frac{\rho(t) \cdot \alpha^{\tau(t)j}}{\sum_{k=1}^{n} \rho(t_k) \cdot \alpha^{\tau(t_k)j}},$$
   where $\{t_1, t_2, \ldots, t_n\}$ is the set of tip nodes of D_i.
   The interpretation of $P(D_i, t, \alpha, j)$ is the probability of a tip node t in the navigation tree D_i. We note that when $\alpha = 1$ then this probability of t is purely proportional to the the score of the trail it induces.
6) $extend(D_i, t)$ returns a navigation tree resulting from the extension of D_i with tip node t.

7) $select(D_i, \alpha, j)$, where α is either 1 or df and j denotes either an exploration or convergence step, returns a tip of D_i chosen by a random device operating according to the probability distribution function $P(D_i, t, \alpha, j)$.
8) $best(D_i)$ returns the trail with the highest score from the set of trails induced by the set of tip nodes of D_i. (If there is more than one highest scoring trail choose the shorter one, otherwise choose any one of them uniformly at random.)
9) $overall\_best(\{T_1, T_2, \ldots, T_M\})$, where $\{T_1, T_2, \ldots, T_M\}$ is a set of M trails, returns the highest scoring trail from this set; we call this trail the *best trail*. (If there is more than one highest scoring trail choose the shorter one, otherwise choose any one of them uniformly at random.)

The Best Trail algorithm, whose pseudo-code is given in Algorithm 2, takes $K \geq 1$ starting URLs, $U_1, U_2, \ldots, U_K$, and a parameter $M \geq 1$, which specifies the number of repetitions of the algorithm for each input URL. It outputs a set of K trails $\{B_1, B_2, \ldots, B_K\}$ one for each input URL.

The algorithm has a main outer for loop starting at line 2 and ending at line 16, which computes the best trail for each one of the K input URLs. The first inner for loop starting at line 3 and ending at line 14 recomputes the best trail M times, given the starting URL U_k. The overall best trail over the M iterations with the same starting URL is chosen at line 15 of the algorithm. We note that due to the stochastic nature of the algorithm, we may get different trails T_i at line 13 of the algorithm from two separate iterations of the for loop starting at line 3 and ending at line 14. The algorithm has two further inner for loops, the first one starting at line 5 and ending at line 8 comprises the exploration stage of the algorithm, and the second one starting at line 9 and ending at line 12 comprises the convergence stage of the algorithm. Finally, the set of K best trails for the set of K input URLs is returned at line 17 of the algorithm.

Algorithm 2 (Best_Trail($\{U_1, U_2, \ldots, U_K\}, M$))
1. **begin**
2. **for** $k = 1$ to K **do**
3. **for** $i = 1$ to M **do**
4. $D_i \leftarrow \{U_k\}$;
5. **for** $j = 1$ to $I_{explore}$ **do**
6. $t \leftarrow select(D_i, 1, j)$;
7. $D_i \leftarrow extend(D_i, t)$;
8. **end for**
9. **for** $j = 1$ to $I_{converge}$ **do**
10. $t \leftarrow select(D_i, df, j)$;
11. $D_i \leftarrow extend(D_i, t)$;
12. **end for**
13. $T_i \leftarrow best(D_i)$;
14. **end for**
15. $B_k \leftarrow overall\_best(\{T_1, T_2, \ldots, T_M\})$;
16. **end for**
17. **return** $\{B_1, B_2, \ldots, B_K\}$:
18. **end.**

One important issue is removing redundancy from the output trails to increase their relevance to users viewing them. We consider a trail to be redundant if all the pages in the trail are contained in another more relevant trail. Within a trail, we consider a pages to be redundant if either it is not relevant to the user query or if the page is duplicated previously in the trail, and removing the page leaves a valid trail with respect to the topology of the web graph.

Optimisation of the algorithm and implementation issues are discussed in [70]. Therein we also report experiments we have run test the behaviour of the algorithm and discuss how to tune its various parameters. The application of the Best Trail algorithm for keyword search with a relational database is discussed in [71].

We now briefly describe a navigation system we have been developing, which uses the Best Trail algorithm to construct trails that are relevant to a user's query. The trails are presented to the user in a tree-like structure which he/she can interact with. This is in sharp contrast to a search engine which merely outputs a list of pages which are relevant to the user query without addressing the problem of which trail the user should follow.

The navigation system obtains the preferred trails for navigation, given a user query, from the *navigation engine* and requests pages for display from the web site via a proxy. The navigation engine consists of two main modules: (i) the information retrieval module, and (ii) the best trail module. The information retrieval module does conventional information retrieval over web pages combined with using the potential gain metric to determine starting points

for navigation. The best trail module implements the Best Trail algorithm to compute the preferred trails for navigation given the input query; see [70] for more details.

The main mechanism of the user interface is the *navigation tree window*, which displays the preferred trails given the user query, organised in the form of a tree structure with the trails being ranked from the most preferred, according to their score. The user interacting with the navigation tree window can select any web page on one of the preferred trails by clicking the corresponding link and thus causing the page to be displayed in the browser window. Putting the cursor over a link in the navigation tree window will cause a small window to pop-up displaying the summary of the destination web page. The mechanisms of the user interface provide the user with guidance and context throughout a navigation session. The user interface can be embodied within a web site as a navigation mechanism complementing or replacing a web site search engine. A screen shot for the query "neural networks" is shown in Figure 4; an alternative interface which displays trails in graphical form is described in [72]. The trails clearly indicate that both the BSc and MSc courses support neural network options and that Chris is involved in teaching the subject and doing research in this area. (We note that the single page trail, whose title is "Chris's Home Page", is in fact a web page about neural network application projects, and is thus a case where the HTML title was not used effectively. Similarly, the page on the two page trail with a link to the pdf file nn_prnt.pdf, whose title is also "Chris's Home Page", is in fact a web page about understanding neuronal coding.)

We conclude this section by mentioning a usability study we carried out to test whether our navigation system enhances users' search experience [46]. Our results show the potential of navigational assistance in search tools, since overall, users of the navigation system employed less clicks and took less time in completing their tasks than those using a conventional search engine. One reason for these positive results may be that users of the navigation system did not have to click on the back button as much as users of a conventional search engine but instead could use the navigation tree window.

8 Trail Records

Here we present a model for analysing the record of trails that emerge from either an individual user or a group of users through navigation within the web graph over a period a time. The model is within the area of *web usage mining* [45], which is concerned with finding patterns in "surfers" navigation behaviour.

We define a *web view* (or a *record of trails*) as a collection of trails which are the result of user navigation sessions over a period of time. Thus a web

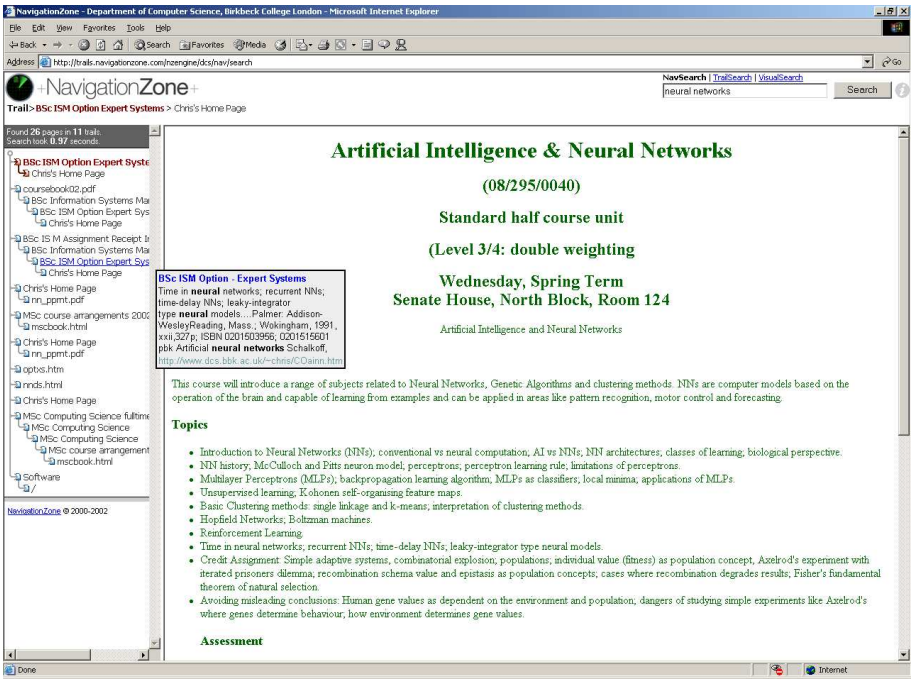

Fig. 4. Navigation engine results for the query "neural networks"

view is a subgraph of the web graph induced by a collection of emergent trails. We limit the trails in a web view by two threshold parameters, as follows:

1) *support* $\alpha \in [0, 1)$; accept into the web view only trails whose initial probability is greater than α.
2) *confidence* $\beta \in [0, 1)$; accept into the web view only trails whose product of transition probabilities is greater than β.

Alternatively, we accept into the web view only trails whose overall probability is above some *cut-point* $\lambda \in [0, 1)$, with $\lambda \geq \alpha \cdot \beta$.

Let $\mathcal{M}$ be an ergodic Markov chain modelling user's navigation behaviour within the web graph. Then a web view over $\mathcal{M}$ constrained by λ is the set of all trails T in $\mathcal{M}$ such that the probability of T is greater than λ. (An alternative formalisation of a web view separating the support and confidence thresholds can also be given.)

We now describe a technique for constructing a web view, which is concerned with finding frequent user behaviour patterns. In $\mathcal{M}$ the high probability trails, i.e. those having probability above the cut-point, correspond to the user's preferred trails. We assume that we have at our disposal web log data; for example, collected from the user's browser or from server logs, which make it is possible to infer user navigation sessions. (The log data could be

collected for a group of users rather than a single user if we are interested in collaborative trails rather than personalised trails.) It is customary to define a navigation session as a sequence of page visits (i.e. URL requests) by the user where no two consecutive visits are separated by more than a prescribed amount of time, which is normally not more than half an hour.

When sufficient such log data is available we pre-process this data into a collection of trails, each trail being represented as a sequence of URLs. Moreover, we assume that the start and end URLs of all trails correspond to the user's home page. We note that a trail may appear more than once in this collection, since the user may follow the same trail on two or more different occasions. We then build an ergodic Markov chain (or equivalently HPA), say $\mathcal{M}$, whose initial probabilities correspond to the frequency the user visited a page present in any one of the input trails, and whose transition probabilities correspond to the frequency that a link was followed in any one of the input trails. We observe that the states of $\mathcal{M}$ are the pages the user visited and the topology of $\mathcal{M}$, i.e. its underlying graph, is induced by the links the user followed. In constructing $\mathcal{M}$ we have implicitly assumed that when the user chooses a link to follow he/she does not base his/her decision on the previous pages visited during the navigation session. That is, we have assumed that $\mathcal{M}$ is a first-order Markov chain. This assumption can be relaxed so that N (with $N \geq 1$) previous pages including the current one are taken into account; the case with $N = 1$ is the first-order case when the user bases his/her decision only on the page currently being browsed. The parameter N is called the *history depth*.

Given a history depth $N > 1$, a higher-order Markov chain can be reduced to a first-order Markov chain by aggregating states. The drawback of such a higher-order Markov chain is the increase in the number of states, which is expected to be $n \cdot b^{(N-1)}$, where n is the number of states in the first-order Markov chain and b is the average number of out-links embedded in a page. Thus there is a trade-off between the history depth and the complexity of the Markov chain measured by its number of states. The decision on whether the gain in accuracy by adopting a higher-order Markov chain is significant can be aided by statistical techniques [15]. (Another approach we are looking at, which increases the history depth yet maintains a complexity as low as possible, is to use variable order Markov chains [58] or dynamic Markov modelling [19]; see [41].)

Once the HPA $\mathcal{M}$ has been constructed from the collection of trails, which have been pre-processed from the log data, we employ a Depth-First Search (DFS) to find all the trails in $\mathcal{M}$ starting from the user's home page and having probability above the cut-point λ. We have run extensive experiments with synthetic and real data to test the performance of the DFS algorithm [8]. It transpires that for a given cut-point there is a strong linear correlation between the size of $\mathcal{M}$, measured by its number of states, and the running time of the algorithm, measured by the number of links it traverses. Moreover, for

a given cut-point, the number of mined trails increases linearly with the size of $\mathcal{M}$. On the other hand, the number of mined trails increases exponentially with the decrease in the cut-point.

The DFS algorithm has two main drawbacks. Firstly, since it is an exhaustive search it will, in general, return too many trails. Secondly, if we increase the cut-point to reduce the number of trails, then on average the returned trails become short, and therefore may not be very interesting. These observations have led us to develop two heuristics for mining high quality trails. Our first approach [9], which we call the *fine-grained* heuristic, limits the numbers of trails returned via a stopping parameter, which is between zero and one, that determines an upper bound on the sum of probabilities of the returned trails. The method used to implement this heuristic is to explore trails, whose probability is above the cut-point, one by one and in decreasing order of probability. When the stopping parameter is zero then the fine-grained heuristic reduces to the DFS algorithm and as the parameter gets closer to one less trails are returned. Our initial results show that for a given cut-point the number of trails decreases almost linearly with the increase in the stopping parameter, indicating that the stopping parameter provides good control over the number of trails. Our second approach [10], which we call the *inverse-fisheye* heuristic, is a method of obtaining longer trails while controlling their number. This is obtained by having a dynamic cut-point which is high at the initial stage of the exploration in order to limit the number of trails, and decreases in subsequent stages in order to allow further exploration of the selected trails. The user specifies a maximum exploration depth, which limits the length of the trails returned. Our initial results show that if the initial cut-point is not too low and the decrease in the cut-point at each step is gradual then we can reduce the number of trails while increasing their average length.

A different approach has been put forward by Schechter et al. [60] with the aim of using web log data to predict the next URL to be requested from a user. Their algorithm essentially constructs a *suffix tree* [3] generated from user trails within a navigation session inferred from the log data. A suffix of a trail is added to the tree only if the occurrence count of the predecessor node of the suffix is greater than a predefined threshold. An algorithm is devised, which finds the maximal prefix of a trail in the suffix tree that matches the current user trail, and then uses the found trail to predict the next URL as the next node in the tree that is on this trail. Using our example web graph shown in Figure 2, a user trail might correspond to

$$\text{SCSIS} \rightarrow \text{Research} \rightarrow \text{Students}$$

matching the prefix of two trails in the suffix tree having Azy and Kevin as their next node. The algorithm will predict either Azy or Kevin as the next page the user will browse, depending on which one was traversed more frequently by the user.

We have extended our model to include physical spaces as well as virtual web spaces [42]. We now distinguish between a *trail*, which may be through a physical space such as a museum exhibit, and a *trail record* which is a hypertextual trail providing an account of the user navigation session, be it physical or virtual. Trail records thus provide us with a model of users' actions, which can be viewed as a spatial/temporal account of their activities. Such records of a physical experience act as a memory aid which can be expanded by further experiences and additional content. Moreover, using the data mining techniques we have described above, user navigation patterns through the space under consideration can be inferred.

We close this section with a brief discussion of a stochastic model of users "surfing" the web [28, 37]. In this model each page a user visits has a value and the user browsing a page has to decide if to continue "surfing" by clicking on a link or to stop "surfing". In Huberman et al. [28] the user will continue "surfing" if the expected cost, to the user, of continuing is smaller than the expected gain obtained from future information in pages which will be browsed. Analysis based on this model leads to a power-law distribution, where the probability of "surfing" by following L links is proportional to $L^{-3/2}$. Huberman et al.'s model does not take into account the topology of the web graph being navigated. In Levene et al. [37] the user is assumed to be navigating within a Markov chain reperenting the web graph, where longer trails are less probable than shorter trails. Again a power-law distribution, which is proportional to the length of the trail being navigated, is derived for the probability of "surfing".

9 Navigation in the Mobile Web

The ability to connect to the internet through mobile devices such as mobile phones and handheld and portable computing devices means that we can be connected "anytime" and "anywhere". The limitations of mobile devices in terms of screen size, computing power and lack of traditional input devices such as keyboard and mouse, means that alternative input modalities such as pen and voice will become prevalent, and innovative software solutions such as voice recognition, hand-writing recognition and predictive text systems will be necessary. Information needs that do not require complex and lengthy navigation such as browsing news headlines, addresses and train schedules, can readily be supported on mobile devices. Other information needs which are more navigational such as information gathering on a particular topic are poorly suited for mobile devices [62]. The issues relating to the user interface design on mobile devices to support search are discussed elsewhere in this book.

As location sensing technologies are already widely available [27], *location-aware services*, which focus the application to the physical location of a user,

can play an important role in narrowing down the information need a mobile user has. For example, technologies such as GPS can assist users in physical navigation, such as helping them to find a local restaurant serving their favourite cuisine. We note that we can also add the time dimension to mobile services to further narrow down the information need; in the local restaurant example, the time of day will obviously have an effect on the answer.

An interesting specialisation of a common navigation tool is that of *dynamic bookmarks* [22], where bookmarked pages may vary from location to location. As opposed to storing the URL of the bookmarked page, a dynamic bookmark is associated with a name and a set of attributes such as the type of service required. One idea which has to be further investigated is the creation of *dynamic authored trails*, which provide the user with a specific set of information needs. For example, the trail

News Headlines → Top Shares → Directions to Hotel → Nearest Restaurant

may be useful on a business trip, where each information need is only instantiated at the instruction of the user within a time and location context.

We briefly mention at two recent attempts to improve navigation from mobile devices. Anderson et al. [2] develop an adaptive algorithm to improve web navigation for mobile devices by adding shortcut links to web pages thus allowing the user to reach a destination page as quickly as possible. Their algorithm for creating high-quality shortcuts is based on the user's navigation history. Smyth and Cotter [64] investigate the navigation problem within mobile portals, where personalisation techniques can reduce the click-distance to relevant information. By collecting the user's past clicks on menu items, conditional probabilities can be computed and more probable paths suggested to the user.

10 Navigation in Social Networks

Viewing the web as a social network has resulted in novel techniques to enhance web search and navigation [34]. In this context several researchers have gathered evidence that the web is a *power-law network* in the sense that its in-degree distribution (and to a lesser extend its out-degree distribution) follows a power-law [12]. Moreover, the web obeys the *small-world* property of having a short average distances between any two connected pages; this distance was shown to be roughly sixteen clicks [12].

The fact that short paths exist between two pages does not imply that they are computationally easy to find. Kleinberg [33] investigated this problem in the context of a two-dimensional lattice structure, where nodes have short-range links to their immediate neighbours in the lattice and long-range links to more distant nodes according to a parameter that determines the probability of a link between two nodes as a function of their lattice distance. He found

that a decentralised algorithm, in which the only information known to the searcher of a short path is purely local, exists if and only if the parameter's value is two, i.e. long-range links follow an inverse-square distribution. The algorithm is "greedy" in the sense that it chooses a link that brings it as close as possible to the target.

Adamic et al. [1] consider a more realistic scenario, where no global information about the position of the target node is available to the searcher, and where the topology is not restricted to a lattice. They show that random walks in power-law networks naturally lead to nodes with a high in-degree but that intentionally following nodes with a high in-degree performs even better. From their analysis they conclude that local search strategies in power-law graphs scale sub-linearly with the size of the graph. Similar results were shown in [31] who formulated the problem in terms of finding short paths between two nodes in the network.

Watts et al. [69] consider the property of *searchability* in social networks, where searchability is defined as the property of being able to find a target quickly. We recast their model in terms of locating pages in the web graph. In this model web pages are the individuals in the social network and their out-going links are their network neighbours. Each web page can have several semantic categories attached to it; these are the set of *identities* of the page. For example, the School's home page may be categorised both in terms of its subject area, i.e. Computer Science, and in terms of its location within the University of London, implying that the page belongs to two *groups* of web pages according to its dual identity. We assume it possible to measure similarity between pairs of pages within an ontology, for example within a category of the Open Directory structure. The similarity between two pages within a category can be defined as the least number of levels needed to reach a common ancestor category; other similarity measures, say involving textual similarity are possible. The *social distance* between two pages, each having several identities, is defined as the smallest distance over all common identities that the two pages may have. So for example, the distance between the School of Computer Science and a non-computing school within the University of London may be one, while the distance between the School of Computer Science and another school of Computing outside the University of London may be greater than one due to different research area specialisations. Under some distributional assumptions, which enable simulation of the model, Watts et al. [69] show that using only local knowledge of the neighbours of a page within the network, and knowledge of the set of identities of the target page, the target can be found quickly. The algorithm to find the target is a "greedy" one similar to Kleinberg's algorithm [33], which chooses to follow the link to a neighbour that is closest to the target in terms of social distance.

This approach seems to reconcile directory-based navigation and link-based navigation on the web, where users navigate within the web topology choosing links according to their semantic proximity to the goal. It also has

some similarity with the Best Trail algorithm, presented above, where navigation is link-based but the choice of link is made according to the information need of the user.

11 Open Problems

We close this chapter with a list of several open research problems that warrant further investigation.

- One important issue that is not well understood is how to evaluate navigation trails. In the Information Retrieval community there is a strong tradition of evaluation in terms of precision and recall [4] but is is not clear how these notions carry over to the realm of trails.
- It would be interesting to compare the potential gain and gain rank metrics to other web metrics such as PageRank and investigate whether it is beneficial for them to be combined in some manner.
- Another issue that is still open is how to incorporate personalisation and collaboration within the context of our model of trail records.
- The Best Trail is essentially a probabilistic best first algorithm. It would be worth investigating whether it could be applied within the Artificial Intelligence domain.
- Designing novel user interfaces for trail-based systems is another area which is under developed yet highly important.

References

1. L.A. Adamic, R.M. Lukose, A.R. Puniyani, and B.A. Huberman. Search in power-law networks. *Physical Review E*, 64:046135, 2001.
2. C.R. Anderson, P. Domingos, and D.S. Weld. Adaptive web navigation for wireless devices. In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 879–884, Seattle, Washington, 2001.
3. A. Apostolico. The myriad virtues of subword trees. In A. Apostolico and Z. Galil, editors, *Combinatorial Algorithms on Words*, volume F12 of *NATO ASI Series*, pages 85–96. Springer-Verlag, Berlin, 1985.
4. R.A. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press and Addison-Wesley, Reading, Ma., 1999.
5. M.K. Bergman. The deep web: Surfacing hidden value. White paper, Bright Planet, July 2000.
6. T. Berners-Lee. *Weaving the Web*. Orion Books, London, 1999.
7. M. Bernstein. The bookmark and the compass: Orientation tools for hypertext users. *SIGOIS Bulletin*, 9:34–45, 1988.
8. J. Borges and M. Levene. Data mining of user navigation patterns. In B. Masand and M. Spiliopoulou, editors, *Web Usage Analysis and User Profiling*, Lecture Notes in Artificial Intelligence (LNAI 1836), pages 92–111. Springer-Verlag, Berlin, 2000.

9. J. Borges and M. Levene. A fine grained heuristic to capture web navigation patterns. *SIGKDD Explorations*, 2:40–50, 2000.
10. J. Borges and M. Levene. A heuristic to capture longer user web navigation patterns. In *Proceedings of International Conference on Electronic Commerce and Web Technologies (EC-Web)*, pages 155–164, Greenwich, U.K., 2000.
11. A. Broder. A taxonomy of web search. *SIGIR Forum*, 36, Fall 2002.
12. A. Broder, R. Kumar, F. Maghoul, P. Raghavan, A. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener. Graph structure in the Web. *Computer Networks*, 33:309–320, 2000.
13. F. Buckley and F. Harary. *Distance in Graphs*. Addison-Wesley, Redwood City, Ca., 1990.
14. V. Bush. As we may think. *Atlantic Monthly*, 176:101–108, 1945.
15. C. Chatfield. Statistical inference regarding Markov chain models. *Applied Statistics*, 22:7–20, 1973.
16. E.H. Chi, P. Pirolli, and J.E. Pitkow. The scent of a site: A system for analyzing and predicting information scent, usgae, and usability of a web site. In *Proceedings of ACM Conference on Human Factors in Computing Systems*, pages 161–168, The Hague, Amsterdam, 2000.
17. A. Cockburn and B. McKenzie. What do web users do? An empirical analysis of web use. *International Journal of Human-Computer Studies*, 54:903–922, 2001.
18. A. Cockburn, B. McKenzie, and M. JasonSmith. Pushing back: evaluating a new behaviour for the back and forward buttons in web browsers. *International Journal of Human-Computer Studies*, 57:397–414, 2002.
19. G.V. Cormack and R.N.S. Horspool. Data compression using dynamic Markov modelling. *The Computer Journal*, 30:541–550, 1987.
20. P.M.E. De Bra and R.D.J. Post. Searching for arbitrary information in the WWW: The fish-search for Mosaic. In *Proceedings of International World Wide Web Conference*, Chicago, Il., 1994.
21. D. Dhyani, W.K. Ng, and S.S. Bhowmick. A survey of web metrics. *ACM Computing Surveys*, 34:469–503, 2002.
22. S. Duri, A. Cole, J. Munson, and J. Christensen. An approach to providing a seamless end-user experience for location-aware applications. In *Proceedings of the first international workshop on Mobile commerce*, pages 20–25, Rome, 2001.
23. D.C. Englebart and W.K. English. A research center for augmenting human intellect. In *Proceedings of AFIPS Fall Joint Conference*, pages 395–3410, San Francisco, Ca., 1968.
24. G.W. Furnas. Effective view navigation. In *Proceedings of ACM Conference on Human Factors in Computing Systems*, pages 367–374, Atlanta, Georgia, 1997.
25. M.A. Hearst. Next generation web search: Setting our sites. *Bulletin of the Technical Committe on Data Engineering*, 23:38–48, 2000.
26. M. Hersovici, M. Jacovi, Y.S. Maarek, D. Pelleg, M. Shtalheim, and S. Ur. The shark-search algorithm - An application: Tailored web site mapping. In *Proceedings of International World Wide Web Conference*, pages 317–326, Brisbane, 1998.
27. J. Hightower and G. Borriello. Location systems for ubiquitous computing. *IEEE Computer*, 34:57–66, 2001.
28. B.A. Huberman, P.L.T. Pirolli, J.E. Pitkow, and R.M. Lukose. Strong regularities in world wide web surfing. *Science*, 280:95–97, 1998.

29. T. Joachims, D. Freitag, and T. Mitchell. WebWatcher: A tour guide for the World Wide Web. In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 770–775, Nagoya, Japan, 1997.
30. J.G. Kemeny and J.L. Snell. *Finite Markov Chains*. D. Van Nostrand, Princeton, NJ, 1960.
31. B.J. Kim, C.N. Yoon, S.K. Han, and H. Jeong. Path finding strategies in scale-free networks. *Physical Review E*, 65:027103, 2002.
32. H. Kim and S.C. Hirtle. Spatial metaphors and disorientation in hypertext browsing. *Behaviour and Information Technology*, 14:239–250, 1995.
33. J. Kleinberg. Navigation in a small world. *Nature*, 406:845, 2000.
34. R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. The web and social networks. *IEEE Computer*, 35:32–36, 2002.
35. J. Lamping and R. Rao. The hyperbolic browser: A focus + context technique for visualizing large hierarchies. *Journal of Visual Languages and Computing*, 7:33–55, 1996.
36. S. Lawrence and C.L. Giles. Accessibility of information on the web. *Nature*, 400:107–109, 1999.
37. M. Levene, J. Borges, and G. Loizou. Zipf's law for web surfers. *Knowledge and Information Systems*, 3:120–129, 2001.
38. M. Levene and G. Loizou. Navigation in hypertext is easy only sometimes. *SIAM Journal on Computing*, 29:728–760, 1999.
39. M. Levene and G. Loizou. A probabilistic approach to navigation in hypertext. *Information Sciences*, 114:165–186, 1999.
40. M. Levene and G. Loizou. Kemeny's constant and the random surfer. *American Mathemtical Monthly*, 109:741–745, 2002.
41. M. Levene and G. Loizou. Computing the entropy of user navigation in the web. *International Journal of Information Technology and Decision Making*, 2:459–476, 2003.
42. M. Levene and D. Peterson. Trail records and ampliative learning. Research Report BBKCS-02-10, School of Information Systems and Computer Science, Birkbeck University of London, October 2002.
43. G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7:76–80, 2002.
44. M. Marchiori. The quest for correct information on the web: Hyper search engines. In *Proceedings of International World Wide Web Conference*, pages 265–276, Santa Clara, Ca., 1997.
45. B. Masand and M. Spiliopoulou, editors. *Web Usage Analysis and User Profiling*, volume 1836 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, Berlin, 2000.
46. M. Mat-Hassan and M. Levene. Can navigational assistance improve search experience: A user study. *First Monday*, 6(9), 2001.
47. F. Menczer and R.K. Belew. Adaptive retrieval agents: Internalizing local context and scaling up to the web. *Machine Learning*, 39:203–242, 2000.
48. S. Mukherjea and J.D. Foley. Showing the context of nodes in the world-wide web. In *Proceedings of ACM Conference on Human Factors in Computing Systems*, Denver, Colorado, 1995. Short paper.
49. B.H. Murray and A. Moore. Sizing the internet. White paper, Cyveillance, July 2000.

50. T. Nelson. Xanalogical structure, needed now more than ever: parallel documents, deep links to content, deep versioning, and deep re-use. *ACM Computing Surveys*, 31:1–32, 1999.
51. J. Nielsen. *Hypertext and Hypermedia.* Academic Press, Boston, Ma., 1990.
52. J. Nielsen. *Designing Web Usability: The Practice of Simplicity.* New Riders Publishing, Indianapolis, Indiana, 2000.
53. J.M. Nyce and P. Kahn, editors. *From Memex to Hypertext: Vannevar Bush and the Mind's Machine.* Academic Press, San Diego, Ca., 1991.
54. T. Oren. Memex: Getting back on the trail. In J.M. Nyce and P. Kahn, editors, *From Memex to Hypertext: Vannevar Bush and the Mind's Machine*, pages 319–338. Academic Press, San Diego, Ca., 1991.
55. L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Working paper, Department of Computer Science, Stanford University, 1998.
56. P. Pirolli, J.E. Pitkow, and R. Rao. Silk from a sow's ear: Extracting usable structures from the Web. In *Proceedings of ACM Conference on Human Factors in Computing Systems*, pages 118–125, Vancouver, 1996.
57. E. Rivlin, R. Botafogo, and B. Shneiderman. Navigating in hyperspace: Designing a structure-based toolbox. *Communications of the ACM*, 37:87–96, 1994.
58. D. Ron, Y. Singer, and N. Tishby. The power of amnesia: Learning probabilistic automata with variable memory length. *Machine Learning*, 25:117–149, 1996.
59. G. Salton and M.J. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill, New York, NY, 1983.
60. S. Schechter, M. Krishnan, and M.D. Smith. Using path profiles to predict HTTP requests. *Computer Networks and ISDN Systems*, 30:457–467, 1998.
61. F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34:1–47, 2002.
62. A.J. Sellen and R. Murphy. The future of the mobile internet: Lessons from looking at web use. Technical Report HPL-2002-230, Information Infrastructure Laboratory, HP Laboratories, Bristol, August 2002.
63. F. Shipman, R. Furuta, D. Brenner, C. Chung, and H. Hsieh. Guided paths through web-based collections: Design, experiences, and adaptations. *Journal of the American Society for Information Science*, 51:260–272, 2000.
64. B. Smyth and P. Cotter. Personalized adaptive navigation for mobile portals. In *ECAI2002, Proceedings of the 15th European Conference on Artificial Intelligence*, pages 608–612, Lyon, France, 2002.
65. R.S. Sutton and A.G. Barto. *Reinforcement Learning.* MIT Press, Cambridge, Ma., 1998.
66. L. Tauscher and S. Greenberg. How people revisit web pages: Empirical findings and implications for the design of history systems. *International Journal of Human-Computer Studies*, 47:97–137, 1997.
67. K. Tochtermann and G. Dittrich. Fishing for clarity in hyperdocuments with enhanced fisheye-views. In *Proceedings of ACM Conference on Hypertext*, pages 212–221, Milano, Italy, 1992.
68. R.H. Trigg. From trailblazing to guided tours: The legacy of Vannevar Bush's vision of hypertext use. In J.M. Nyce and P. Kahn, editors, *From Memex to Hypertext: Vannevar Bush and the Mind's Machine*, pages 353–367. Academic Press, San Diego, Ca., 1991.
69. D.J. Watts, P.S. Dodds, and M.E.J. Newman. Identity and search in social networks. *Science*, 296:1302–1305, 2002.

70. R. Wheeldon and M. Levene. The best trail algorithm for adaptive navigation in the world-wide-web. In *Proceedings of 1st Latin American Web Congress*, Santiago, Chile, November 2003.
71. R. Wheeldon, M. Levene, and K. Keenoy. Search and navigation in relational databases. *Computing Research Repository*, cs.DB/0307073, July 2002.
72. R. Wheeldon, M. Levene, and N. Zin. Autodoc: A search and navigation tool for web-based program documentation. In *Poster Proceedings of International World Wide Web Conference*, Hawaii, 2002.