Stepping Stones and Pathways: Improving Retrieval by Chains of Relationships between Documents

Fernando Adrián Das Neves

This Dissertation submitted to the faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Science

Dr. Edward A. Fox, Chair
Dr. Naren Ramakrishnan
Dr. Chris North
Dr. Dennis Kafura
Dr. Ron Kriz

September 16, 2004
Blacksburg, Virginia

Keywords: Information retrieval, Literature-based discovery, Combination of sources of evidence, Indexing of scientific literature.

Stepping Stones and Pathways: Improving Retrieval by Chains of Relationships between Documents

Fernando Adrián Das Neves

Abstract

The information retrieval (IR) field has been successful in developing techniques to address many types of information needs. However, there are cases in which traditional approaches to IR are not able to produce adequate results. Examples include: when a small set of (2-3) documents is needed as an answer rather than a single document, or when "query splitting" is required to satisfactorily explore the document space. We explore an alternative model of building and presenting retrieval results for such cases. In particular, we research effective methods for handling information needs that may:

1. Include multiple topics: A typical query is interpreted by current IR systems as a request to retrieve documents that each discusses all topics included in that query. We propose an alternative interpretation based on query splitting. It allows queries to be interpreted as requests to retrieve sets of documents rather than individual documents, with meaningful relationships among the members of each such set.
2. Be interpreted as parts in a chain of relationships: Suppose a query concerns topics t_1 and t_m. Is there a relation between topics t_1 and t_m that involves t_2 and possibly other topics as in $\{t_1, t_2, \ldots t_m\}$? Thus, we propose an alternative interpretation of user queries and presentation of the results. Our interpretation has the potential to improve retrieval results whenever there is a mismatch between the user's understanding of the collection and the actual collection content. We define and refine a retrieval scheme that enhances retrieval through a framework that combines multiple sources of evidence.

Query results in our interpretation are networks of document groups representing topics, each group relating to and connecting to other groups in the network that partially answer the user's information need. We devise new and more effective representations and techniques to visualize results, and incorporate the user as part of the retrieval process.

We also evaluate the improvement of the query results based on multiple measures. In particular, we verify the validity of our approach through a study involving a collection of Operating Systems research papers that was specially built for this dissertation.

Acknowledgments

To my advisor, Dr. Edward Fox, who gave me enough freedom to think by myself, stopped my panic attacks, and had to suffer by fixing my broken English.

To Naren (Dr. Ramakrishnan), who many times worked like my other advisor, and helped me to find and read all the good things I never heard of.

To Dr. Ron Kriz, who funded most of my years here, gave me the opportunity to play with expensive toys for grown-ups, and was more my friend than my boss.

To Kaye Kriz, who proofread my entire dissertation – I truly appreciate all your help.

To my aunt, Dr. Graciela Gelmini, that worked from UCLA as my third unofficial advisor, teaching me many of the tricks of the trade, and cheering me up when things were going badly.

Finally, and most important, to my friends Paula, Carola, Chad, Ali, Joanne, and Hussein, who made the difference between just living here and having the best years of my life.

Index

Tables and Figures

1. Introduction

IR systems have evolved from difficult to use, expert-mediated retrieval systems, to become the gateway for everyday users accessing large collections, like the ACM Digital Library, Citeseer [Lawrence99], or the WWW itself, where browsing alone is impractical. Studies show that in the case of the WWW, the largest collection ever available, though at least 85% use search engines to find material in the Web [Kobayashi 2000], yet the inability to find relevant information is still one of the top reasons for user dissatisfaction [Huberman98]. The causes of this inability are varied. They have been widely studied, and include:

- the different information seeking models of users and the support from the system [Saracevik75] [Bates89] [Borgman85] [Belkin95];
- the problem of identifying information need, and what defines relevance [Hutchins77] [Saracevik75] [Mizzaro97];
- how to express and interpret the representation of the information need as a query [Saracevik75] [Spink97];
- how to infer the relevance of documents for a certain query, which today is system-dependent and varies widely from system to system; and the model used by the system to represent document information and relationships.

We propose to explore a different way to interpret the information request in a user query, and proceeding from that, a different way to present the query result and allow user feedback on it.

The classical interpretation of an information need as a query has been very successful in providing answers for many users, and the IR field has developed techniques proven effective for this scenario. However, while this interpretation works successfully most of the time, there are mismatches between the user's need and the collection content that this interpretation does not address. There are cases in which the query represents a single concept in the user's mind, but it may not be a single concept in the document space. That is, a query that from the user's point of view is a single concept, and is described by one topic, may be better described as the connection of the partial results from more than one topic, when analyzing those topics and their relationships.

1.1 An alternative Interpretation to User Queries

We propose to explore a different way to interpret the information request in a user query, and proceeding from that, a different way to present the query result and allow user feedback on it. The Information Retrieval field has had through most of its modern history a common interpretation of and approach to what is the intention of a

user query. In this interpretation, a user **query** is the practical description of an information need. It describes a concept in which the user is interested. The **objective** of this query is to retrieve all documents that are closely related to that concept. The **input** to the IR system is a set of words describing the query, possibly using logical connectors and other modifiers. The **output** of the system is then a list of documents ranked according to the system's estimate of the odds that the content of the document matches the concept.

While this interpretation has worked successfully most of the time, it neglects a not trivial observation: *that the query represents a single concept in the user's mind, but it may not be a single concept in the document space*. That is, a query that from the user's point of view is a single concept may be more than one topic when analyzing the topics covered by a collection, their description, and relations.

This situation is not unlikely in the case of users who although knowledgeable in the subject matter, are not familiar with the collection content. It has also been observed before:

> *"The documents selected by the user as relevant to his query are often found in two or more distinct groups in the document vector space, and these groups are separated from one another by nonrelevant documents".*
>
> E. Ide, in *"Relevance Feedback in an Automatic Document Retrieval System"* [Ide69]

As a concrete example of such queries and the difficulties faced while trying to answer them with current systems, we propose the following scenario: While reading the Digital Library Conference Proceedings, a researcher became interested in one paper regarding use of mobile devices in the context of Digital Libraries. In particular, the researcher was interested in the possibility of having annotations or selections in electronic documents to work as queries, to retrieve material related to the selection or annotation. The idea was to find previous work combining automatic link generation from annotations in the context of mobile devices for digital libraries. Furthermore, let's say that the query is formulated in terms of "*automatic link generation*", "*mobile device*", "*digital libraries*", and "annotation".

Suppose the researcher decided to search in the ACM Digital Library. Should this search be carried out, he would find out that the relation is not apparent in any of the query results, and in many cases one topic is completely dominating all others in the query results. It is possible to argue that the problem lies in formulating the right query, and that term weighting and citation analysis would have done a much better job. The fact is however that without a deep understanding of the topics, many users would be unable to weight query terms other than by a successive, brute force iterative process.

The real reason these queries failed except when dividing the query in two parts is that the intersection between the topics Automatic Link Generation and Mobile Devices is empty; rather there is a relation that is transitive and involves the topics "*Link*" and "*Query*".

However, there is a relation going from Automatic Link Generation to the use of Annotations as queries in Mobile Devices in Digital Libraries (Figure 1.1), which exists within ACM DL, and can be justified in terms of particular documents (Table 1.1). This chain of relationships was found through hours of manually browsing through the library content, and is as follows, from top to bottom:

Document #	Paper Title	Topics	Appears in
1	"Selective Text Utilization and Text Traversal"	Automatic Linking Generation from text	Proceedings of SIGIR 93
2	"Query-based Navigation in Semantically Indexed Hypermedia"	Combining linking and querying for document retrieval	Proceedings of HT 97
3	From Reading to Retrieval: Free form annotations as queries"	Using annotations as query sources (link anchors) in mobile devices	Proceedings of SIGIR 99
4	"Digital library information appliances"	Study of usage of mobile devices in the context of Digital Libraries, including automatic link generation from annotations	Proceedings of DL 98
5	"Introducing a Digital Library in a Reading Group"	Study of annotation and reading using mobile devices in the context of Digital Libraries	Proceedings of DL 99

Table 1.1. A concrete example of a document sequence connecting "automatic link generation" with "use of mobile devices in digital libraries".

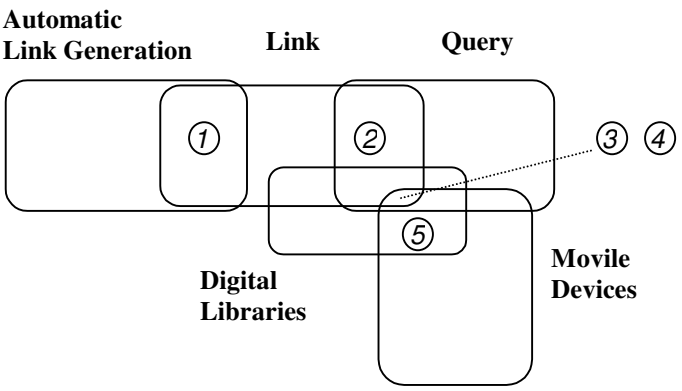


Figure 1.1. Schematic relationship of the query topics in the ACM Digital Library. Overlapping illustrates an approximation to the number of documents related to all topics in the intersection, and area size (very) approximately represents the number of documents in a concept. Numbers represent the documents in Table 1.1.

This relationship, however, is not direct but sequential, and is not detected by traditional retrieval methods. This example is by no means a critique of ACM DL, which is by itself a wonderful example of how the digital availability of content allows looking for and finding new connections among material that would otherwise be very difficult to analyze as a whole. This example is instead a way to show how certain existing DLs, with rich content, metadata and usage information, are prime candidates to become test collections for new research in information retrieval and recommender systems.

We argue for the study of a more general type of query. These are able to:

• **Include multiple subjects.** The typical query as interpreted by current IR systems is taken as a request to retrieve documents that satisfy all the topics included in it. It is often the case that some of the terms in the query dominate the results, or that it is not possible to satisfy all the parts in the query. We propose an alternative interpretation, which is based on what has been studied in the past as "query splitting" [Borodin68], but that allows posing queries made of separate parts, and interpreted as a request to retrieve sets of individual documents, with meaningful relations among the members of each such set of documents. An historian trying to find support for a new point of view relating two historical events is an example of a user information need not addressed by current information systems.

• **Be interpreted as parts in a chain of relationships.** Multiple-part queries can be understood as an ordered sequence. Is there a relation between topics t_1 and t_k that includes $\{t_1, t_2,...t_n, t_m,...t_k\}$ from a certain perspective? A teacher retrieving new material for a course from a digital library like the National Science Digital Library (www.nsdl.org) may also be interested in material that coherently relates the new material with existing topics in his or her curricula.

Based on this observation, we propose a new type of user query, interpreted as two related, separable concepts (see Figure 1.2). By separable we mean that the two concepts are each identifiable from the query formulation. The **objective** of the query then is to retrieve one or more sequences of documents that support a valid set of relationships between the two concepts. The **input** to the IR system is still a set of words, but now it is representing two concepts. The **output** of the system is a connected network of chains of evidence. Each chain is made of a sequence of additional concepts (**stepping stones**). Each concept in the sequence is logically connected to the next and previous one, and the chains provide a rationale (a **pathway**) for the connection between the two original concepts. To increase the user's understanding of the chain, it is desirable that the stepping stones be justified by concrete documents, along with the connections (relationships) among those documents.

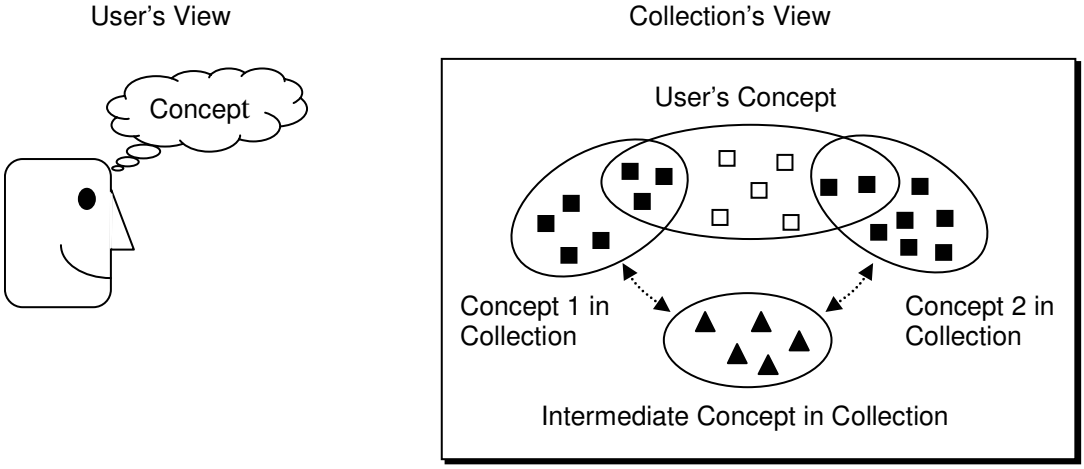


Figure 1.2. What is one concept in the user's mind, versus what the collection really contains. Squares and triangles represent documents in the collection, colored black if they are relevant to the user's information need, white if they are not.

The picture above illustrates the mismatch between the user's understanding of the topic versus the collection content. What the user has in mind (see left), usually will be treated as a single query by the system, and will retrieve similar documents (see oval on the right, in the top center). However, sometimes the retrieved documents may not be relevant (see open squares in the oval). But, if the query is cleverly split rather than "averaged", queries built from two well chosen "sub-concepts" may retrieve mostly relevant documents (see closed squares). Further, if an intermediate concept that relates the sub-concepts is identified, it may both explicate the new query formulation and help locate additional relevant documents (see triangles).

Automatic query expansion techniques like Local Context Analysis [Xu96] can somewhat alleviate this problem, but they have serious limitations. They cannot decide by themselves how to expand to match X or Y optimally, since typically the user gives no information on the overlapping of concepts X and Y. Even if the user has perfect knowledge of the collection content (which is almost always not the case), most users

prefer to browse the top ranked documents and modify the query by themselves if necessary [Spink2001], instead of providing the system with suitable relevance feedback information.

In a sense, a pathway is a generalization of the result of current IR systems: a document is a chain made of a single item, that includes all topics, where the relationships are all contained in the document instead of different documents.

The type of user questions we address in our alternative interpretation differs from those addressed in classic IR systems. Those systems are oriented to what Belnap [Belnap76] calls direct answers: an answer that completely, but just completely, answers the question. Furthermore, the query is interpreted as being (again following Belnap) a whether-question or what-question, for which the user looks for a direct answer. While this interpretation is perfectly valid, it is not the only possible interpretation of user need. We intend to expand the coverage of IR systems to answer ill-defined questions, with non-enumerated answers, as "*How is X related to Y?*", "*What is the X of Y?*", or "*Why is X related to Y?*". Many questions that are not specifically expressed in the above forms can be transformed to such forms when, looking at them from the collection's point of view, the question involves more than one topic.

To illustrate the type of information needs we address, consider three examples, illustrated in Figures 1.3 to 1.5. First, if a user would search the WWW nowadays (we used Google) for Literary Style in Sherlock Holmes, a classic IR system would give mostly low-relevance results, unless it addresses Sherlock Holmes in terms of being a Victorian novel. Our system would retrieve a network like that depicted in Figure 1.3.

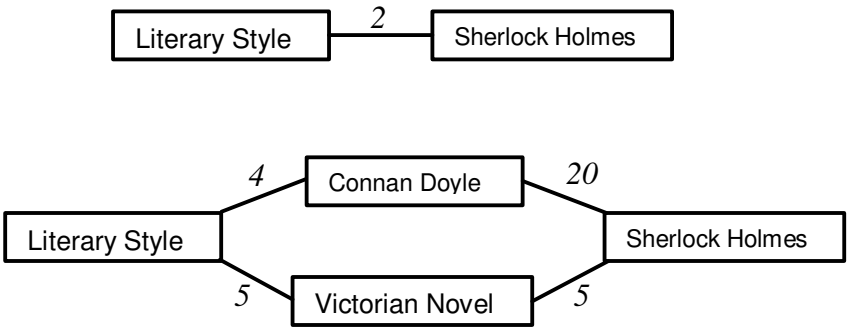


Figure 1.3. Number of relevant WWW pages in the top 20 retrieved using Google. The top portion shows only 2 relevant documents were found when searching for "Literary Style" "Sherlock Holmes". The bottom portion shows the numbers of relevant results when following pathways across stepping stones relating Literary Style to Sherlock Holmes.

Second, Figure 1.4 illustrates another example, taken from a real need from a fellow researcher, interested in applications of machine learning techniques to the building of

virtual environments. Searching for "*Virtual Environments Machine Learning*" in Google returned a long list of results, but few related to the application of machine learning in virtual environments. There were a total of 4 relevant results in the top 20. Splitting the query, and building the chain shown, improved precision considerably, as measured by the numbers of relevant results in the top 20, which in Figure 1.4 appear above the links.

With regard to Figure 1.4, the cause of low recall when using the original query was the high number of casual co-occurrences of the two topics, mostly in the case of faculty or research project list pages. This co-occurrence in the context of university project or departmental pages was only evident after looking at each retrieved document. Our technique makes this result evident for the user by highlighting the reasons the topics were considered as related, by identifying the topics, and by depicting the connections. A network representation helps to differentiate casual co-occurrences from concrete relationships.

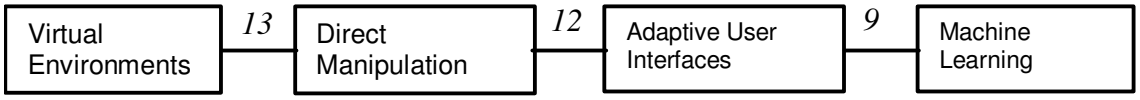


Figure 1.4. Stepping Stone and Pathways network relating "Virtual Environments" to "Machine Learning".

Finally, Figure 1.5 gives another example, this time with a network relating *Data Mining* to *Recommender Systems*. This query is difficult because there are many possible answers, and larger numbers of pages where both topics appear but there is no clear real information about the relation between the topics. Such a mismatch is illustrated by the home page of a computer science department, where Virtual Environments and Data Mining are distinct specialty areas. The top 20 results in Google returned 7 relevant matches. A network of intermediate topics returns a higher number of relevant pages.

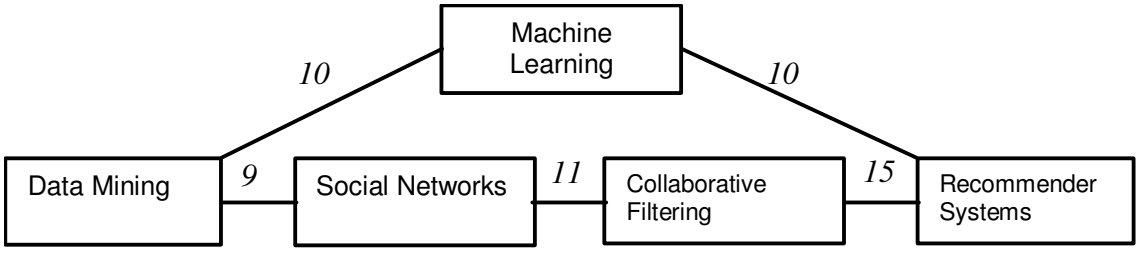


Figure 1.5. Stepping stone and pathways network relating "Data Mining" to "Recommender Systems".

Our approach is useful in that the type of questions it addresses is common during research, or in educational settings. Research involves gathering, relating, and connecting information. It is often the user who has to connect the search results, many times unaware of how his or her needs match the collection content and how the different parts of the collection content are related, according to his or her particular view and needs.

We think that current systems do not handle this type of question adequately, and our technique has the potential to improve both the user's understanding of the collection content, and the ability of the user to obtain relevant information from the collection.

We envision our approach as a retrieval technique in itself, but also particularly fitting as an extra stage in an existing information retrieval process, whenever results in the current systems are not satisfactory. To exemplify a typical usage scenario of the later case, imagine that a user has issued a query for which she only gets low-relevance results. Then, our approach offers a further option consisting of:

1. Under the user's request, the query is split into two sub-queries or concepts (Figure 1.6.a).
2. A group of candidate connections is displayed (Figure 1.6.b).
3. The user analyzes the relevance of the results of step 2 in terms of appropriateness of the level of description (from too generic to too specific), and optionally requests more intermediate steps, or removes topics that are not of interest (Figures 1.6.c and 1.6.d).

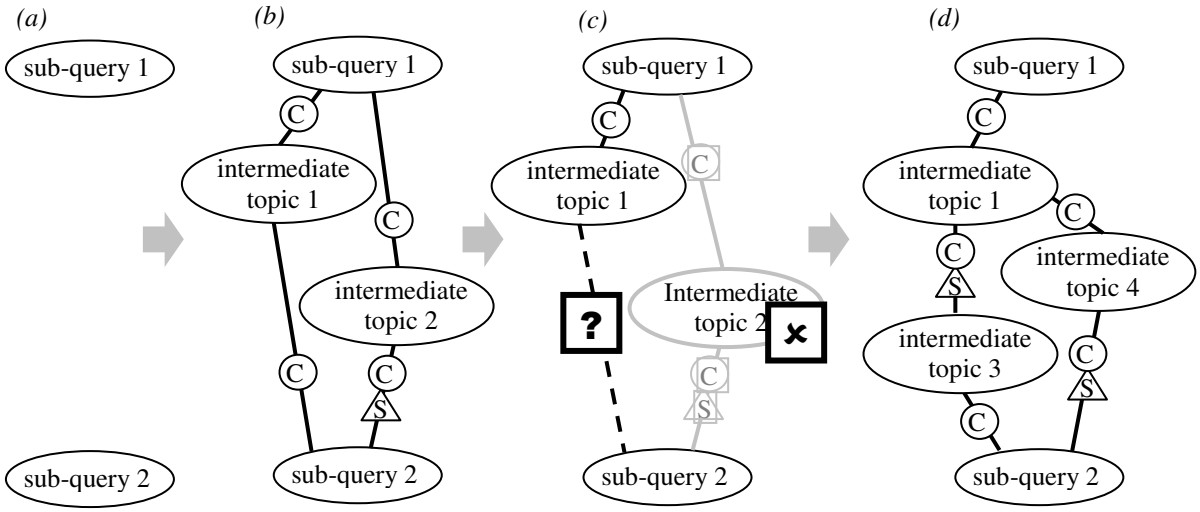


Figure 1.6. Retrieval Process in Stepping Stones. First, the user query is split (a). After splitting, the system presents a number of documents, exemplifying possible candidates for stepping stones (b). Some intermediate topics are related by document content (circled C), while others are related by citations (marked as S, for structure). The user determines which one are relevant or not, and which ones are too generic (c). In this case, the user decided that the intermediate topic 2 is not relevant, and that the connection between intermediate topic 1 and sub-query 2 is too vague, so he requests additional stepping stones between them. The system modifies the network, and new stepping stones (intermediate topics 3 and 4) are generated between intermediate topic 1 and sub-query 2 (d). The retrieval process then continues, with the user deciding which stepping stones are relevant, and requiring more stepping stones when the connection is not clear.

1.2 Need for an Alternative Interpretation

We summarize the problems we address:

- User knowledge or assumptions do not necessarily directly match the collection content and structure.
- The majority of current IR systems insist on a traditional approach to query result (batch retrieval, ranked result) presentation that does not lead to a fully adequate user/system interaction. We will address query presentation both as an HCI issue (presentation, manipulation) and an IR issue (manipulation, connection building).
- Feedback options are reduced to marking documents as relevant/non-relevant, and assuming that anything not marked as relevant is non-relevant. In practice, many times the user lacks context or other information, and so cannot determine the relevance of a reference, document, or author.
- Current information retrieval systems only address the estimation of a relevance score for single documents, but not for relations among documents.

1.3 The Stepping Stone Hypothesis:

We claim: Interpreting a query (that retrieved few relevant results) as the two-topic question "How is X related to Y?", and letting the user manipulate the stepping stones in the resulting network, considering both specificity and relevance, improves the answering capability of the system. It does so by giving a network constructed by both the user and the system, in which an answer need not be contained in a single document, but may be found in a set of documents, along with an explanation regarding the context and relations between attributes of the document sets.

1.4 Research Questions

The research questions we address are:

1) How to build stepping stones and pathways (connections between stepping stones, and to the original sub-queries) from a collection and a query?

2) Do stepping stones + pathways have a positive effect on the relevance of documents retrieved by the system? Do they improve the comprehension of the collection content by the user?

Given a query with low-relevance results, there can be a number of possible scenarios explaining the outcome:

1. An incorrectly formulated query, or
2. A case where the collection does not have pertinent content.

These two possible explanations for poor results are the ones assumed by classic IR systems. Under our model we propose two more scenarios:

3. A low relevance set where the results are dominated by a subset of the query formulation.
4. A query that retrieved a low-relevance result set, but where it is possible to get high relevance for (maybe modified) connected subsets of the query formulation.

Cases 3 and 4 have not been studied in depth. When a query result set has low relevance, a traditional approach is simple query splitting, changing the interpretation of a query to "X or Y", without addressing the problem of coherence from the user's point of view: are X and Y related (possibly by other intermediary concepts)?

We approach the problem of low-relevance sets under scenarios 3 and 4. There are almost no similar studies that we know of, except perhaps [Swanson2001], and none with the approach we are taking. We will study the effects of our hypothesis in two different areas:

1) Create a system that will interpret retrieval under our hypothesis when results under the classical interpretation has low relevance,

2) Perform user evaluation of the system to study the comparative increase in performance against a baseline classic IR system, modifying our system to integrate user feedback that goes beyond the traditional relevance feedback in IR.

2. Literature Review

Summary

In this chapter we have cover different aspects of information retrieval that relate to Stepping Stones and Pathways. We start with the Vector Space Model, the most successful model for document representation and base to many other models. Then we continue with Bayesian networks, a method to consider IR within a probabilistic framework, that we use in Stepping Stones and Pathways to combine different sources of evidence and to calculate scores for pathways.
Since Stepping Stones contain groups of related documents, we overview many clustering techniques of frequent use in IR, with special emphasis on suffix-tree clustering (STC), that we use to create the stepping stones.
Stepping Stones and Pathways uses content and citation evidence to find the similarity between stepping stones and documents. Therefore we cover citation analysis, methods to calculate document similarity and importance using citations, and HITS, a clustering method based on citations.
We finish by giving an overview of user interaction in information retrieval, in terms of involving the user in the retrieval process, and in terms of interfaces for information retrieval systems. We demonstrate why 3D interfaces have not shown the advantage they seem to have in terms or richness of information manipulation, and we briefly describe user interfaces that have inspired the design of the Stepping Stones and Pathways user interface.

Before starting we will give an overview of how all the topics that are about to be developed in this chapter relate to Stepping Stones and Pathways.

Section in this Chapter	Relevance to Stepping Stones
Vector Space Model	Basic model to represent document content
Bayesian Networks	Used to rank connections (pathways)
Document Clustering	Used to create the stepping stones, and also to split and join them
Literature-based Discovery	The area of research that is closer to what Stepping Stones does
Citation Analysis	Used to calculate similarity between documents based on structure (citations)
Interactive Retrieval	Influenced the way the graph of connections can be modified by the user
User Interfaces for IR systems	Discussion of UI designs for interactive IR systems.

2.1 Models of Document Representation and Retrieval

2.1.1 The Vector Space Model for Documents

Most of the information retrieval systems use some variation of the Vector Space Model (VSM) at their core to represent documents, and match document to queries. The Vector Space Model was developed by Salton and his group at Cornell [Salton75]. In this model, both documents and queries are represented as n-dimensional vectors.

The Vector Space Model makes the following assumptions:

1. The frequency of a word in a document is related to the importance of that word in that document.
2. The importance of a word in a document is independent from the importance of other words.

Assumption 1 connects a purely mathematical model of frequency to the concept of what a document is about. Assumption 2 is not true in reality. It is easy to see that the distribution of words over documents is not random, but that some words appear more often after certain other words. Assumption 2 however greatly simplifies the modeling of documents, and when word order is important, extra mechanisms like indexing phrases and recording the word order are added to the basic model.

In VSM all terms appearing in the group of documents (called a *collection*) are assigned an arbitrary order. A document or a query is a vector where the i^{th} dimension is different from zero if and only if the term with index i appears in the document or query. The value of the i^{th} component of a document vector is a function of (among other factors) the frequency of the i^{th} term in the document. Note that in the simplest form of the vector space model, the order of the terms in a document is not important, since each term is mapped to exactly one component of a document vector, regardless of where the term appeared in the document. For this reason this document representation is often called "bag of words", since it resembles the *bag* data structure, where elements have a frequency but not an order.

In VSM, weights associated with terms in a document are often calculated from two quantities:

- The Term Frequency (tf_{ij}), the number of times term i appears in document j.
- The Inverse Document Frequency (idf_i) = $\log(\frac{|D|}{nd_i})$, where $|D|$ is the total number of documents in the collection, and nd_i is the number of documents containing the term i.

Then, if there is a total of W words in the collection, the vector representation of the document with index j is a vector of W dimensions:

$dj=[tf_{1j} \cdot idf_1, tf_{2j} \cdot idf_2, tf_{3j} \cdot idf_3, \ldots, tf_{Wj} \cdot idf_{Wj}]$

The weight $tf \cdot idf$ is used instead of the raw frequency *tf* to weight words to account for the *specificity* of a word, the ability the word to identify a set of documents. The higher the number of documents where a word appears, the less useful that word is to decide if the document must belong to the relevant or non-relevant sets, given a query including that word. Note that in order to calculate *idf*, the number of documents in the collection must be known; that is, the collection must be of a known, fixed size. This assumption is not feasible in all cases, for example for web search engines, since the true extent of the document collection (the whole Word Wide Web) is not known.

When using *idf* to estimate a word's specificity, values that are too low or too high are detrimental to the effective retrieval of documents related to the user query. A very low *idf* value means that the word appears in too many documents to be able to separate that document from other documents.

Words that appear very often in English, like "the", "of", "an", are guaranteed to appear in almost all documents. These words are not specific to any theme in particular, and so they have little or no use in determining the set of documents that are relevant given a query. These words are called *stopwords*, and are usually discarded when indexing and building the document vectors.

A value of *idf* that is too high means that the word appears in very few documents, probably excluding many relevant documents from the relevant set of a query. In order to improve retrieval, researchers have developed techniques to deal with the low values of word frequency. This may be due to synonyms and variations of a word: For example, from a strictly syntactical point of view, *geometric* and *geometrical* are different words, assigned to different dimensions of the document vector, regardless of their closeness in meaning. Since they are mapped to different dimensions, a query that includes one of these words will not match documents including the other. IR systems deal with word synonyms and words with the same root through the use of thesauri and stemming, respectively. Stemming reduces word with the same root to a common form; for example *geometric* and *geometrical* are both reduced to the *geometr* stem, and both are therefore assigned to the same component in a vector.

2.1.2 Calculating the relevance of a document given a query

As we mentioned before, a query in VSM is also a vector, each dimension matching the order and the assignment of words to dimensions in the document set. When a query is issued by the user, the words in the query must have a weight assigned in the query vector. There are many ways of assigning weights to query terms, the simplest one being:

$$q_i = \begin{cases} 1 \text{ if term i appears in the query} \\ 0 \text{ otherwise} \end{cases}$$

Since we have a way to estimate the specificity of a word (its *idf*), then a more sensible weight assignment to query terms is:

$$q_i = \begin{cases} idf(i) \text{ if term i appears in the query} \\ 0 \text{ otherwise} \end{cases}$$

In the simplest form of VSM, the similarity of query q and document d_j is the cosine of the angle between the vectors q and d_j:

$$sim(q, d_j) = \frac{\sum_{i=1}^{w} q_i \cdot d_{ij}}{\sum_{i=1}^{W} q_i^2 \sum_{i=1}^{W} d_{ij}^2}$$

which is a value between 0 and 1, being equal to 0 when the set of words in q and d_j are disjoint (that is, the vectors q and dj are perpendicular to each other), and 1 when q is equal to d_j.

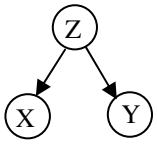
Documents are then ranked by their similarity to the query, via the cosine similarity, and then a subset of the documents with the highest similarity is presented to the user. This subset is determined either by considering a cutoff similarity value, or by limiting the presentation to a maximum number of results.

2.2 Bayesian Networks

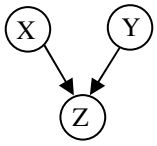
Bayesian networks [Heckerman96] [Jensen96] in information retrieval are a different mechanism to model documents and perform document retrieval. A Bayesian network is a directed acyclic graph *G(V,E,P)* , where *V={v1,v2,...vx}* is a set of vertices, *E* is the set of edges connecting the vertices in *V*, and *P* is a set of conditional probabilities over the edges in E. If there is a subset of edges $\{e_1,\ldots,e_m\}$ such that e_1 goes from vertex v_1 to vertex v, e_2 goes from vertex v_2 to v, and in general vertex e_m goes from v_m to v, then the conditional probability of v depends *only* on $v_1 \ldots v_m$. That is, vertices in a Bayesian network represent random variables, and edges represent conditional probabilities among these variables.

Informally, two variables X and Y are conditionally independent of Z if once the random variable Z has a value assigned to it, knowledge about X gives you no extra information about Y. In other words if X and Y are conditional on Z, once Z is known, whether we know a value for X or not adds nothing to what we know about Y. Bayesian networks encode the assumption that a random variable is independent of all the other variables in

the network, given that the value of its parents is known. In other words, in a Bayesian network $P(v| v_1 \ldots v_m, v_n, v_{n+1}, \ldots v_x) = P(v|v_1 \ldots v_m, v_n)$.



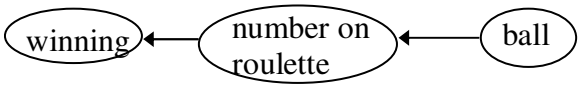
- X depends on Z
- Y depends on Z
- X and Y are conditionally independent given Z
- P(X,Y,Z)=P(X|Z)P(Y|Z)P(Z)



- Z depends on X and Y
- X and Y are independent
- P(X,Y,Z)=P(Z|X,Y)P(X)P(Y)

Figure 2.1 Example of two minimal Bayesian networks, showing the dependencies among nodes expressed in the graph.

A Bayesian network is a graphical representation of the dependencies between variables in a model. For the purpose of building a Bayesian network, it is very important to distinguish between direct and indirect causal dependencies. For example, winning at roulette depends on the number in the roulette wheel matching the number we picked, and the selection of this number in the roulette wheel depends on the path of the ball:



Winning depends directly on a number in the roulette wheel being equal to the one we bet on, while winning depends *indirectly* on the path of the ball, and the number chosen on the roulette wheel depends directly on the path of the ball. A Bayesian network represents only the direct causal dependencies between the variables of the model.

2.2.1 Parents, Descendants, and the Rest

Given any variable v_i in a Bayesian network G(V,E,P), we can divide the rest of the variables in four sets:

$Pa(v_i)=\{v_j|$ there is a direct link from v_j to $v_i\}$, the set of direct parents of v_i

$Ch(v_i)=\{ v_j|$ there is a direct link from v_i to $v_j\}$, the set of direct children of v_i

$De(v_i)=\{ v_j|$ there is a path from v_i to $v_j\}$, the set of descendants of v_i

$Rest(v_i)=V-(Pa(v_i) \cup De(v_i))$

The joint probability of a set of variables $\{v_1,v_2,..,v_n\}$ in a Bayesian network is:

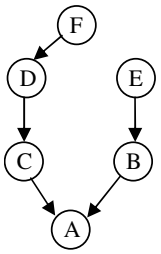
$$P(v_1,v_2,...v_n)=\prod_{i=1}^{n} P(v_i \mid Pa(v_i))P(\bigcup_{x_j \in Pa(v_i)})$$

This formula is known as the **Chain Rule**, and is fundamental in using a Bayesian network as a mechanism to estimate probabilities. Note that the definition of the Chain Rule is recursive, with $\bigcup_{x_j \in Pa(v_i)}$ in itself being a set of variables, and so the way of calculating it depends in turn on the parents of each x_j.

Applying a Bayesian network for inference implies calculating the probability of a set of variables of interest $Q=\{q_1,q_2,\ldots,q_r\}$ given the known values of another subset of variables $S=\{s_1,s_2,\ldots s_p\}$. In other words, we calculate $P(q_1,q_2,\ldots,q_r \mid s_1,s_2,\ldots s_p)$. The strict dependency of the conditional probability of a variable q_i on $Pa(q_i)$ helps in limiting the combinatorial explosion on the number of values when calculating $P(q_1,q_2,\ldots,q_r \mid s_1,s_2,\ldots s_p)$. To see why, if we denote by $V-(Q \cup S)$ all the other variables in the network that are not in the sets of variables Q (the ones we want to estimate) or S (the ones whose value we know), then by the laws of probability, we have

$$P(q_1,q_1,...,q_r \mid s_1,s_2,...,s_p)=\frac{\sum_{V-\{Q \cup S\}} P(q_1,q_1,...,q_r,s_1,s_2,...,s_p,V-(q \cup s))}{\sum_{V-(Q \cup S)} P(s_1,s_2,...,s_p,V-(q \cup s))}$$

that is, the conditional probability of $P(Q|S)$ is calculated by adding all the combinations of $P(q_1,q_1,...,q_r,s_1,s_2,...,s_p,V-(q \cup s))$ and $P(s_1,s_2,...,s_p,V-(q \cup s))$ over all the values that the variables in $V-(Q \cup S)$ can take. This is called *marginalizing* over $V-(Q \cup S)$ *[Jensen96],* and as the variables in $V-(Q \cup S)$ take more values, and the network becomes more complex, it can quickly lead to an enormous number of probabilities to calculate or estimate. However, we can take advantage of the strict direct causal dependencies in the network to reduce the number of combinations. For example, let's say we want to calculate conditional probability $P(X=x|E=e,F=f)$ in the following Bayesian network:



For this network,

$$P(A=a \mid E=e, F=f)=\frac{P(a,e,f)}{P(e,f)}=\frac{\sum_{(b,c,d)\in B\times C\times D} P(a,b,c,d,e,f)}{\sum_{(b,c,d)\in A\times B\times C\times D} P(a,b,c,d,e,f)}$$

If B takes 10 different values, C takes another 5 values, and D takes 10 more values, then calculating $P(X=x|E=e,F=f)$ implies marginalizing over |B×C×D| and calculating or estimating $P(a,B=b,C=c,D=d,e,f)$ over |B×C×D|=500 tuples.

Taking advantage of the strict causal dependencies in the network and applying the chain rule reduces the number of probabilities to calculate:

$$P(A=a \mid E=e, F=f)=\frac{\sum_{(b,c,d)\in B\times C\times D} P(a \mid b,c)P(c \mid d)P(b \mid e)p(f)}{P(e,f)}=$$

$$=\frac{\sum_{(b,c)\in B\times C} P(a \mid b,c)\cdot\sum_{c\in C} P(c \mid d)\cdot\sum_{d\in D} P(b \mid e)p(f)}{P(e)P(f)}$$

which reduces the number of probabilities needed to calculate $P(a,e,f)$ *to* a total of |B×C|+|C|+|D|=65 tuples.

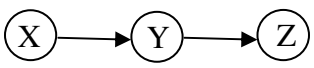
However, it is not always the case that we have to estimate the probability of some variables given some direct or indirect parents. For the general case,

let G(V,E,{$P(x_i|Pa(x_i)$}) be a Bayesian network. For all v_i then

$$P(v_i, Rest(v_i)|Pa(v_i))=P(v_i,Rest(v_i))$$

That is, $Pa(v_i)$ and $\{v_i\}\cup Rest(v_i)$ are conditionally independent. We are not going to prove this rule here but we are going to illustrate it with some examples of its application:

Examples:



X ∈ Rest(Z), Y ∈ Pa(Z)
{X,Z} is independent from Y,

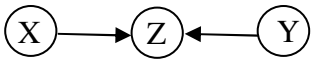
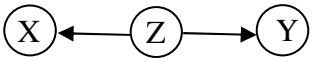
	Y ∈ Rest(X), Z ∈ De(X) {X,Z} is not independent from Y
	Y ∈ Rest(X), Z ∈ Pa(X) {X,Y} is independent from Z

Table 2.1. Examples of the way a node divides a Bayesian network into two conditionally independent sets.

The generalization of this concept is the Markov blanket of a variable v_i, that defines when v_i is independent from any given set of nodes in the network. The **Markov Blanket** of a variable is the minimum set *MB* such that v_i and $\{v1 \ldots v_x\}$-*MB*-$\{v_i\}$ are independent **given** *MB*. It can be proven (see [Jensen96] for details) that the Markov Blanket of a variable v_i consists of $MB(v_i)=Pa(v_i) \cup Ch(v_i) \cup (Pa(Ch(v_i))$.

There are cases in which the dependencies between variables are part of the model. Then the rules above are not enough to reduce the number of probabilities to estimate. In this case there are a number of techniques that introduce additional restrictions in the model. The most commonly used are called "divorcing the parents" and "noisy or".

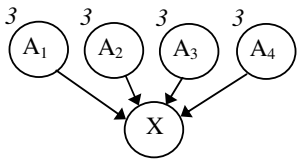
Divorcing:

If there are variables A_1, A_2, A_3, A_4, where A1 takes the values a_1 and a'_1, and A2 takes the values a_2 and a'_2, and is also true that:

$P(X|a_1,a_2,A_3,A_4)=P(X|a'_1,a'_2,A_3,A_4)$, that is, if there are cases for which the probabilities of different values of a subset of variables are equivalent, then:

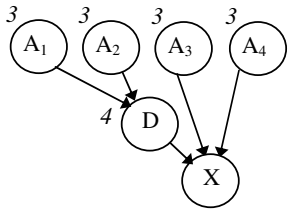
- Add an intermediate variable *D* between *x* and A_1,A_2. *D* has as many states as the number of equivalent classes in $A_1 \times A_2 \times \ldots \times A_n$.

Example: In the following Bayesian network A's are ternary:



Total=81 combinations of A_1, A_2, A_3, A_4

…but, from the knowledge we have about the model, there are 6 of the 9 combinations of values of A_1 and A_2 that are equivalent, so we divorce $\{A_1,A_2\}$ from $\{A_3,A_4\}$:



Total=4×3×3=36 combinations of *D, A_3, A_4*

Noisy-OR:

To calculate *P(X)* we need to be able to calculate the probability of all possible combinations of values of the parents of *X*.

However, if we can assume that:

- All nodes are binary {yes, no}
- For all parents p_i of *X*, p_i =yes implies *X*=yes unless p_i is inhibited. Inhibited means that *X*=no, even though p_i=yes (that is, *X*=no due to other causes).
- Each parent p_i of *X* has a probability of being inhibited q_i: $P(child=no|\ p_i=yes)=q_i$.
- All inhibitor causes are independent: $P(X=no|p_1=yes,p_2=yes)=P(X=no|p_1=yes)\cdot P(X=no|p_2=yes)$

Then under these assumptions:

$$P(X=no|p_1=yes,p_2=yes,\ldots,p_n=yes)=$$
$$=P(X=no|p_1=yes)\cdot P(X=no|p_2=yes)\cdot\ldots\cdot P(X=no|p_n=yes)$$
$$=(1-q_1)\cdot(1-q_2)\cdot\ldots\cdot(1-q_n)$$

since $P(X=yes|p_1=yes,p_2=yes,\ldots,p_n=yes)=1-P(X=no|p_1=yes,p_2=yes,\ldots,p_n=yes)$,

then

$$P(X=yes|p_1=yes,p_2=yes,\ldots,p_n=yes)=1-[(1-q_1)\cdot(1-q_2)\cdot\ldots\cdot(1-q_n)]$$

A Noisy-OR simplifies the calculation of $P(X|p_1,p_2,\ldots,p_n)$, by transforming it into a multiplication involving each $P(X|p_i)$ separately. Even more important, noisy-or has three interesting properties:

1. Absence of evidence from one source ($p_i=0$) does not affect the evidence from other sources, since:
   $P(X)= 1-[1- P(X=yes|p_1=yes)]\cdot\ldots\cdot[1-P(p_i)]\cdot\ldots\cdot[1- P(X=yes|p_n=yes)]=$
   $P(X)= 1-[1- P(X=yes|p_1=yes)]\cdot\ldots\cdot[1-0]\cdot\ldots\cdot[1- P(X=yes|p_n=yes)]$
2. Evidence from more than one source is more effective than evidence from only one source, since any value $v_1>v_2$ of $P(p_i)$ makes $(1- P(p_i)=v_1)<(1- P(p_i)=v_2)$, in turn making $1-(1- P(p_i)=v_1)\cdot\ldots>1-(1- P(p_i)=v_2)\cdot\ldots$

3. If a source of evidence has ($p_i=1$) then all other sources are irrelevant, since:
$P(X)= 1-[1- P(X=yes|p_1=yes)]\cdot\ldots\cdot[1-P(p_i)]\cdot\ldots\cdot[1- P(X=yes|p_n=yes)]=$
$P(X)= 1-[1- P(X=yes|p_1=yes)]\cdot\ldots\cdot[1-1]\cdot\ldots\cdot[1- P(X=yes|p_n=yes)]=1$

2.2.2 Bayesian Networks and IR models for Combining Evidence

Since the early days of work on information retrieval, there has been research on using a probabilistic framework for Ir. While SMART, which used the vector space model with cosine similarity, was the first successful system that implemented free text retrieval, it lacked a more formal framework on how to interpret the results, or on what was the meaning of the similarity measure.

Graphical models [Pearl88] [Buntine94] [Jensen2001] allow for greater flexibility at modeling relations among different sources, when compared with other methods like Latent Semantic Indexing (LSI) [Dumais93], or standard vector space models, because each source can be modeled independently. Graphical models are a generalization of Bayesian networks, where the graph edges still represent dependencies between variables, and the graph is still acyclic, but not all the edges need to be directed. Graphical models rely on a subjective instead of frequentist understanding of probabilities. Such probabilities do not necessarily represent relative frequencies (although they are sometimes calculated like they were), but instead they represent the degree of certainty, belief, or support about a certain feature (a term, a link, etc.), and about one feature given another feature (conditional probabilities).

2.2.3 Inference Networks

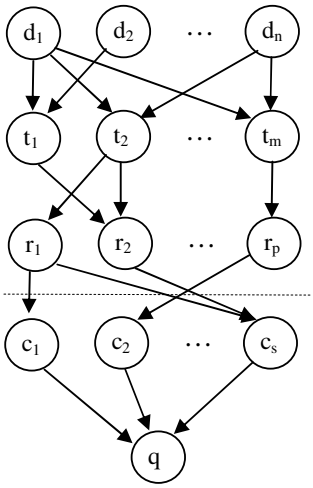
The most common type of graphical model applied to IR is Bayesian Networks. The first successful application of Bayesian networks to IR is the Inference Network Model described in [Fung95], [Turtle90] [Turtle91]. This model uses a Bayesian network to model the document, the relationships between documents and contexts, and the user query. It is built upon two assumptions concerning probabilistic relevance:

Assumption 1 (Binary Independence): Given that it is known if a document is relevant given a set of topics, knowledge of the presence or absence of one document feature does not affect the belief about the presence or absence of other features.

Assumption 2: Given that it is known if a document is relevant given a topic, this does not affect the belief about the relevance of other documents to any other topic.

The network is divided into two sub-networks: The first part is static and is built at indexing time, and consists of the documents, dependencies of terms given the documents, and term relationships (such as polysemy or synonymy). The second part is

dynamic and is built at run-time every time the user submits a query, and consists of the query, the concepts in the query, and the connections between query concepts and term relationships. The graphical representation of this network is as follows:



Here:

$d_1 \ldots d_n$ are the documents

- $t_1 \ldots t_m$ are all the terms in the documents
- $r_1 \ldots r_p$ are relationships between terms (like synonymy)
- $c_1 \ldots c_s$ are concepts in the query
- q is the user query

Figure 2.2. An inference network for information retrieval.

While the generic form of the inference network model is very powerful as a theoretic framework to explain the retrieval process, often in practice a simplified model is used, where concept and term relationships are not present, so the terms connect directly to the query.

2.2.4 Belief Networks

Belief Networks [Ribeiro-Neto96] [Silva2000] are an alternative approach to Inference Networks when explicitly modeling an information retrieval system. While similar in expressive power to inference networks, belief networks can express any inference network used to retrieve documents by content similarity, while the opposite is not necessarily true. Because of their structure, belief networks can reproduce the ranking generated by any inference network. However, generating the ranking produced by belief network from an inference network is not always possible. For example, a belief network can reproduce the cosine-similarity ranking, while this is not possible for an inference network. The key difference is in the modeling of $P(d_j|t)$ (probability of a document given a set of terms or concepts) in belief networks, as opposed to $P(t|d_j)$ used in Bayesian networks (see [Ribeiro96] for details). Figure 2.3 illustrates the differences in the probability structure between the two networks (note the direction of the arrows between the terms *t* and the documents *d*):

Inference Network: Models *P(t|d)* Belief Network: Models *P(d|t)*

Figure 2.3 Examples of Inference and Belief Networks to calculate relevance of a document given a query.

In these graphs, an arrow between two nodes represents the conditional probability of the variable pointed to by the arrowhead, given the value of the variable at the tail of the arrow. Variables not connected by arrows are assumed to be independent.

The very important difference between both networks is the "hammock" structure (Figure 2.4) between a document and a query in the belief network:

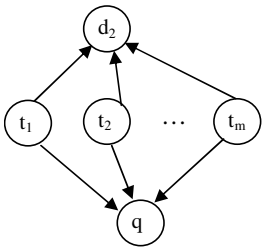


Figure 2.4 Hammock structure in an inference network.
Two nodes represent the endpoints of the hammock, while the connections to a number of intermediate (and disconnected between them) set of nodes constitutes the body of the hammock.

This hammock structure can be made to represent a vector dot-product of $p(d_i|t)$ and $P(q|t)$ (the two vectors of dimension n to multiply), with $P(d_i|t_j)$ and $P(q|t_j)$ being the individual components of the vectors.

Belief networks can calculate any ordering calculated by an inference network, including the traditional cosine similarity used in SMART [Buckley85] and the probabilistic score used in the INQUERY [Callan92] system. Inference networks, on the other hand, cannot replicate the ordering in cosine similarity. Thus, belief networks combine the best of vector and probabilistic operations.

It is worth noting that we are not showing here all the Bayesian networks that can be built. New nodes also arise from other sources of information, such as clusters, or hubs and authorities, like in CLEVER [Kleinberg98]. [Turtle90], [Haines93], and [Silva2000] provide more information on these extensions of the basic network model.

A very innovative approach for IR that allows multiple sources of evidence, using Hidden Markov Models (HMM), is described in [Miller2000]. Their design also starts from a probabilistic setting, needed for the definition of the HMM. Under this model, each document is associated with a particular HMM, where the input sequence is made of the query terms, and at least two states: "General English" and "Document". A query word has a conditional probability of being "General English" or "Document", and an initial a-priori probability of the query starting in either of the two states. Calculating *P(word|General)* is based on the relative frequency of the word in the collection, and *P(word|Document)* is taken from the relative frequency of the word in a particular document. Initial transition probabilities are calculated by using Expectation Maximization on the collection. This approach allows for more complicated combination of possible word sources (states), like bigrams, and different weighting for different parts of the document.

Modeling relationships as a graph and inferring information from it is, however, not exclusive to IR. Starting from the work on techniques modeling the general semantics of a hypertext network [Botafogo91], all structure-based algorithms like Aggregations and Exceptions [Hara91], or Nested Contexts [Casanova91], are closely related to what in IR is called the "Cluster Hypothesis" [Jardine71] [VanRijbergen79]: that relevant documents tend to be more similar to each other than to non-relevant documents, along with the assumption that structure, when considered at a scale that provides enough evidence, is closely related to document semantics.

2.3 IR systems: Evolution and Interpretations

Extending retrieval centered on keywords and Boolean matching, systems nowadays support imprecise matching between the query and a document. Still under the interpretation of documents as mostly unstructured, they try to capture the query concept and document semantics in different ways. However, use of document structure and connections can improve retrieval, as shown in [Fox85] where structure is used explicitly, and in [Brin98] [Kleinberg98] [Dean99] where it is used implicitly to rank the query results. Structural information (links, citations, etc.) is today on a par with content, and cannot be ignored. Google [Brin98], the most successful search engine today, utilizes linking as the most important mechanism to decide the ranking of a relevant document, and CiteSeer [Lawrence99] uses reference counting to rank matching documents.

How to combine and take advantage of richer structure and multiple sources of information has been the area of study of data fusion as well as IR techniques [Fox93] [Dasigi98] [Modha2000]. These extract information from document structure and links, and take advantage of both to formulate queries that have proven to improve the usefulness of results when those techniques are applicable. In particular there is a wealth

of experience on data fusion, using a variety of techniques, including linear combination of sources [Bartell94], logistic regression [Cooper92] [Gey94], Inference [Turtle90] and belief networks [Silva2000], and link-based evidence [Picard98]. These show that combining sources improves recall and/or precision compared to a document term-only approach. [Fox93] experimented with combining ranking results from different query formulations and relevance functions, including the vector similarity and weighting of SMART, and extended Boolean queries using the P-Norm method [Fox83] [Fox85] [Fox88]. They found that combining P-Norm and vector model similarity measures was often more effective than the ranking resulting from the similarity calculated separately by each method. Belkin, Kantor, Fox, and Shaw [Belkin95b] showed how combining evidence from different queries improved retrieval, even when the combined queries were created in different systems. They also found that in some cases different systems found different sets of highly relevant documents, and a weighted OR combination improved retrieval by expanding the result of retrieved documents beyond that which would be retrieved by only one of the systems. The works of Lee [Lee97] and Vorgt and Contrell [Vogt98] are particularly interesting for our purposes, since they find support for the hypothesis that combining evidence is more successful when the different sources of evidence overlap in the set of relevant documents, but differ in the sets of non-relevant documents retrieved (what Lee calls the unequal overlap property).

2.4 Document Clustering

Document clustering (aggregating documents in groups related by a certain criteria) has been of interest since the early days of information retrieval. Jardine and van Rijsbergen expressed the clustering hypothesis in 1971 as: “the associations between documents convey information about the relevance of documents to requests” [Jardine71]. They proposed to cluster documents based on document content, to match user queries with relevant documents, and to expand the list of relevant documents by finding the clusters that are closest to the documents in the relevant set (adding the other documents in the cluster or clusters to the relevant set). Since individually matching queries to documents ignores relations between documents, it was hoped that clustering would improve document recall. The underlying hypothesis is that is possible to separate documents into relevant and non-relevant groups. In practice the results have not been totally successful, with results varying from collection to collection and one clustering method to another. Some studies have showed support for the cluster hypothesis [Rorvig99] [vanRijsbergen75], and success in using clustering for retrieval [Crotf87] [Hearst96]; other studies have shown results contradicting the cluster hypothesis [Voorhees85] [Shaw97].

Clustering retrieval results as a way to help the user explore the result set has been more broadly successful. [Leuski2001] shows that clustering top ranked documents improves the ability of the user to quickly locate documents relevant to his interest from the ones deemed relevant by the retrieval system. Scatter/Gather [Cutting92] [Hearst95] [Hearst96] uses clustering to manipulate a result set and pick the relevant documents. It applies clustering as part of an interactive retrieval process where users discard non-relevant clusters, join relevant clusters that are too generic, and split clusters that are too

particular. [Wang92] combines link and content information to organize documents retrieved from the web, showing improvement over clustering based only on content.

2.4.1 Clustering Methods

The number of clustering methods in the literature is staggering. Here we describe the most popular clustering methods used in information retrieval. Clustering methods are usually classified by the method used to create the clusters, in the following groups:

1. Partition-based Clustering
2. Graph-based Partition Methods
3. Hierarchical/Agglomerative Clustering
4. Non-Disjoint Clustering
5. Other Methods

All document clustering methods share two characteristics. They define:

1. A method to evaluate the fitness of a document with respect to a cluster,

2. The quality of the clusters with respect to a score function.

Since clustering is usually an unsupervised classification method, these two measures are used iteratively to group the documents, and to decide when to stop clustering. The score function of the clustering method implicitly defines the preferred shape of the clusters, either by distance relationships or by assuming that the spatial distribution of the points in a cluster follows a probability function included in the model.

Most document cluster algorithms are based on some measure of similarity between documents. The way to define and evaluate this distance is most commonly to represent documents as vectors of features. A feature can be a word, a citation from one document to another, or in general any characteristic useful to describe the document:

$d_i = [d_{i1}, d_{i2}, d_{i3}, ...d_{im}]$ where d_{ij} is the weight of feature *j* in document *i*.

Once documents are represented as vectors, then some distance measure over the vectors is defined. Euclidean distance and cosine similarity (angle between document vectors) are by far the most popular distance measures, but a variety of measures have been used in the literature. The next table shows the distance measures most commonly used for documents:

Measure	Definition
Manhattan Distance	$\sum_{i=1}^{m} \left| d_{ji} - d_{ki} \right|$
Euclidean Distance	$\sqrt{\sum_{i=1}^{m} \left(d_{ji} - d_{ki} \right)^2}$
Jaccard Measure	$\frac{\sum_{i \in d_j \cap d_k} d_{ji} d_{ki}}{\sum_{i \in d_j} d_{ji} + \sum_{i \in d_k} d_{ki} - \sum_{i \in d_j \cap d_k} d_{ji} d_{ki}}$
Dice's Coefficient	$\frac{\sum_{i \in d_j \cap d_k} d_{ji} d_{ki}}{\sum_{i \in d_j} d_{ji} + \sum_{i \in d_k} d_{ki}}$
Cosine Similarity	$\frac{\sum_{i=1}^{m} \left(d_{ji} d_{ki} \right)}{\sqrt{\sum_{i=1}^{m} \left(d_{ji} \right)^2 \sum_{i=1}^{m} \left(d_{ki} \right)^2}}$
P-Norm Similarity	$\left(\sum_{i=1}^{m} \left| d_{ji} - d_{ki} \right|^p \right)^{1/p}$

Table 2.3 Different measures of document similarity.

2.4.2 Partition-based Clustering

Partition-based clustering methods divide the document set into **disjoint** subsets, trying to distribute the documents as evenly as possible according to the score function. Since in almost all cases finding the optimal combination of points and clusters that maximizes the score function is an intractable problem, these clustering algorithms are usually iterative and include heuristic functions for document/cluster pairing.

K-Means is a partition-based clustering method popular in IR due to its simplicity. The K-Means algorithm divides the documents into K clusters (where the number K is defined before the algorithm is run). There are many variations of the K-Means algorithm to mention here, but all of them generally follow these steps:

1. Initial Document Partition:
Assign one document to each of the K clusters, for example by picking *K* documents at random.

2. Iterative Partitioning:
`while` at least one document is assigned to a different cluster `do`

```
  for each document d:
      assign d to the cluster C where the distance between d and the center of C is the
      smallest. If d is already assigned to another cluster C', then move d from C' to C.
  end
  compute new cluster centers
end
```

K-Means is usually implemented by taking the cluster centroid as the document center, although it is possible to use a different combination, like averaging the distance of the document *d* to the 3 documents in *C* closest to the centroid, if the cluster document space is expected to be very sparse and the centroid is not the best representation of the cluster content. Some implementations of K-Means recalculate the cluster center each time a document is reassigned. A practical consideration in that implementation is that K-Means may not converge to a unique solution, but find two solutions (two cluster sets) that are repeated one after the other, where documents are assigned to one solution, then to the other and then back to the first one. The stopping criteria in this case is either that no more documents change clusters, or that in step *i* the cluster set CS_1 leads to cluster set CS_2, while in step $i+1$ CS_2 leads back to CS_1.

The complexity of K-Means is O(*KnI*), where *I* is the number of iterations, and *n* is the number of documents. In each pass each document is evaluated against each of the *K* clusters, documents are reassigned, and the process is repeated *I* times.

K-Means is sensitive to the initial document partition; different initial assignments of K documents to K clusters may lead to different cluster partitions, since those K documents work as the initial cluster centers.

The main disadvantage of K-Means is that K, the number of partitions, must be decided in advance regardless of the document distribution, and so the inter-cluster distance may greatly vary for any pair of clusters.

2.4.3 Graph-Based Partition Methods:

These methods build a derived graph structure from the document relationships, then try to find the cuts in the graph that keep highly connected components in the graph together. They consider the documents as described by a collection of discrete attributes, so the matching between two document features (terms, citations, etc.) usually ignores any weights assigned to those features, and only considers whether the feature appears in both documents or not.

The ROCK [Guha99] algorithm uses the Jaccard distance with a threshold to establish links between documents. The basic idea is that two documents are similar if they have enough neighbors in common, using not only direct links but 1-step transitive links too. The main disadvantage of ROCK is that it may create clusters with highly variable size.

CHAMELEON [Karpis99] is a hierarchical graph-based clustering method, where two clusters are merged only if the inter-connectivity and closeness between the two clusters are comparable to the internal inter-connectivity and closeness within the clusters. As in ROCK, nodes in the graph represent documents, and weighted edges represent similarities among the documents. The initial graph is formed by taking *k* documents and assigning the rest of the documents to the closest neighbor of each node. CHAMELEON creates clusters of more even size than ROCK, although is also sensible to outliers.

METIS [Karypis98][Kapyris] is a family of multilevel graph-partition clustering algorithms: they reduce the size of the graph by collapsing vertices and edges, partition the smaller graph, and then uncoarsen it to construct a partition for the original graph. The advantage of METIS over other multilevel graph clustering algorithms is that it has proven to be fairly robust at detecting clusters in the presence of noise, and the authors provide a free implementation.

Spectral partitioning clustering algorithms like [Kleinberg98] are hierarchical clustering methods based on the observation that there is a relationship between the second eigenvalue of L(G), the node/edge matrix of a graph G (called the *Laplacian matrix* of the graph), and the partitions of strongly connected components in the same graph. They work by performing successive bisections of the graph following the algorithm:

```
Compute the eigenvector v2 (the second eigenvector)
       corresponding to eigenvalue lambda2 (2nd eigenvalue) of L(G)
  for each node n of G
     if v2.n < 0
        put node n in partition N-
     else
        put node n in partition N+
     end
  end
```

Spectral partitioning clustering is applied in [Kleinberg98] to cluster bibliographical data in the field of databases with good results, finding that the bisections correspond to “areas” in the database field, like theoretical versus practical papers.

2.4.4 Hierarchical Clustering

Hierarchical clustering algorithms proceed either top-bottom or bottom-top. Top-bottom algorithms divide the set into subsets, which are in turn divided until the stop criteria for clustering have been reached, dictating that a further subdivision would split a valid cluster. The spectral partitioning algorithms described above are also top-bottom hierarchical clustering algorithms. In bottom-top algorithms, documents are grouped, and these groups are combined into larger groups until the stop criteria is reached, dictating that joining clusters would fuse document groups that should belong to different clusters. Bottom-top clustering algorithms are called agglomerative clustering algorithms. The most commonly used are: Single Link [Sneath73], Complete Link [King67], and Minimum Variance [Ward67] [Murtagh84] algorithms. Single Link and Complete Link algorithms join groups based on a distance similarity. In the Single Link method, the

distance between two groups is the minimum of the distances of all pairs made from one document from each group. In the Complete Link algorithm, the distance between two groups is the maximum between all pairs. In each step for both Single and Complete Link, the two groups with the lowest distance are merged.

The Complete Link algorithm tends to create tight clusters [Baeza92]. The Single Link algorithm, on the other hand, tends to follow chains of items that are close to each other, creating elongated clusters [Nagy68]. For both algorithms, the general steps are as follows:

```
C = {one cluster for each document in D}
Find the pair (c1,c2) with minimum distance d between all pairs of
clusters in C
while d<max_threshold do
    Cm = merge c1 and c2
    C = C -{c1,c2} ∪ {Cm}
    Find the pair (c1,c2) with minimum distance d
    between all pairs of clusters in C
end while
```

The output is a number of dendograms, classification trees of N levels where leaves are documents, branches at level N-1 are pairs of documents, branches at level N-2 are merged pairs of documents, and so on until level 1, that contains all documents in the cluster.

Ward's clustering algorithm [Ward67] also works bottom-up merging clusters, but the difference is that the selection of the pair to merge is not based on the smallest distance, but on joining groups in such a way that the merging minimizes the increase in a measure of homogeneity. The homogeneity measure of a group *C* is called the inertia of the group, and is defined as follows:

$I_C = \frac{1}{|C|}\sum_{i=1}^{|C|} dist^2(x_i, \bar{x}_C)$, where $x_i \in C$ and $\bar{x}_C$ is the mean of all x_i.

What I_C measures is the average dispersion of the cluster. When two clusters are merged, the inertia of the merged cluster is greater than the inertia of the two clusters. The general form of Ward's algorithm is almost the same as with single and complete clustering, except that in each pass the criteria to find a pair of groups to merge is the minimum increase in inertia, rather than minimum distance.

It is clear that all these hierarchical algorithms are similar. Lance and Williams [Lance67] [Lance67b] found a common formula that unifies these and other hierarchical clustering algorithms. Using the generic form of the hierarchical clustering algorithms we described above, the distance criterion for joining two clusters c_i and c_j is:

$$d(c_i \cup c_j, c_k) = \alpha_1 d(c_i, c_k) + \alpha_2 d(c_j, c_k) + \beta d(c_i, c_j) + \gamma \left| d(c_i, c_k) - d(c_j, c_k) \right|$$

where *d* is a distance (dissimilarity) function between two clusters.

Clustering Method	α_1	α_2	β	γ
Single Linkage	0.5	0.5	0.	-0.5
Complete Linkage	0.5	0.5	0	0.5
Group Average	$\frac{n_i}{n_i + n_j}$	$\frac{n_i}{n_i + n_j}$	0	0
Weighted Group	0.5	0.5	0	0
Centroid	$\frac{n_i}{n_i + n_j}$	$\frac{n_i}{n_i + n_j}$	$\frac{-n_i n_j}{(n_i + n_j)^2}$	0
Ward	$\frac{n_i + n_k}{n_i + n_j + n_k}$	$\frac{n_j + n_k}{n_i + n_j + n_k}$	$\frac{n_k}{n_i + n_j + n_k}$	0

Table 2.4. Lance-Williams coefficients for most well-known agglomerative clustering methods. ni represents the number of documents in cluster i.

2.4.5 Non-Disjoint Clustering Algorithms

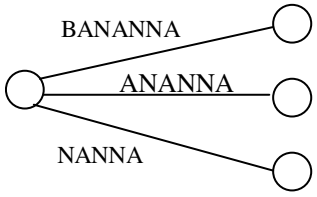
The clustering algorithms described up to now create disjoint clusters: an item belongs to only one cluster, and the intersection of any two clusters is always empty. However forcing elements to belong to only one cluster is not always a sensible option: Biochemistry articles can fit in the Biology or Chemistry section of a journal; scientific papers may fit into more than one category of a classification system. Here we will describe Suffix-Tree Clustering (STC), a method by which documents can be clustered while each document belongs to multiple clusters, which we will use in the implementation of Stepping Stones and Pathways.

2.4.5.1 Suffix Tree Clustering (STC)

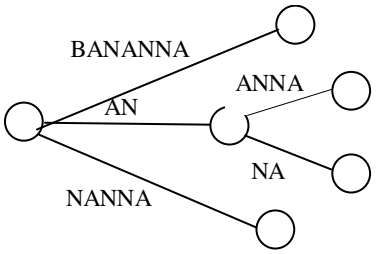
Compared to the clustering algorithms described up to now, Suffix Tree Clustering [Zamir98] uses a completely different criterion to group documents. STC groups documents by common phrases (word sequences), which must appear often enough as not to be consecutive by chance, and must be neither too short (again, may be together by chance) nor too long (create a cluster that is too restrictive, with very few elements). The intuition behind using phrases as clustering criterion is that phrases will represent concepts, and phrases that occur very often will be valid concepts. Since a document is likely to have many significant phrases, the document becomes part of more than one cluster.

As the name implies, Suffix Tree Clustering works by building a suffix tree. A Suffix Tree is a data structure designed to efficiently search for occurrences of a substring within a much larger string *S*. Each path in the suffix tree from the root to a leaf is a suffix of the string *S*. While the problem of finding a substring inside a string could be solved by using one the string search algorithms, the cost of finding a substring when there are many possible cases becomes prohibitive. A suffix tree can match a substring of length *n* in at most *n* steps, one for each element in the string. Of course, the drawback is in building the suffix tree before the search.

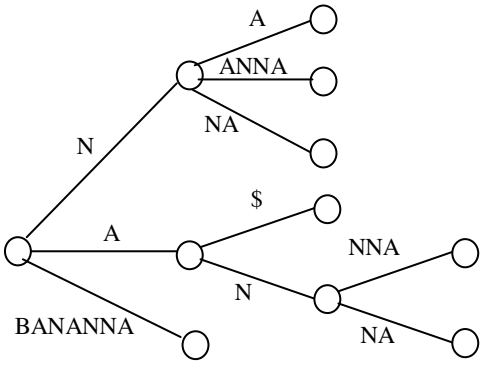
To illustrate the cost of building a suffix tree in the simplest possible way, we consider the string BANANNA. The suffix tree will contain the substrings BANANNA, ANANNA, NANNA, ANNA, NNA, NA, and A. When building the suffix tree one substring at a time, after adding the first three substrings we get



The next step is to add ANNA, for which we have to split the middle path:



Continuing this way, the final suffix tree for the string BANANNA is:



Note that to build the suffix tree this way we had to go through every suffix in *BANANNA*, taking $O(N^2)$, where *N* is the length of the string. McCreight [McCreight76] was the first to design an algorithm to build the suffix tree in O(*N*) while scanning the string from right to left, and then Ukkonen [Ukkonen95] provided a solution in O(*N*) (where N is the number of documents), but this time scanning the string from left to right.

Ukkonen's original paper is very precise but hard to follow to write a concrete implementation; [Nelson96] gives a description of the rationale and function of Ukkonen's algorithm that is much easier to understand from an implementation point-of-view.

Each path in the suffix tree built by STC is a paragraph in a document, with a paragraph represented as a string of word index numbers. The process of clustering takes 3 steps:

1) Formatting the paragraphs: A section of each document is chosen to summarize what the document is about. Stopwords are removed (while adding a 'word here' mark to avoid generating false word sequences). The document section is divided into paragraphs.

2) Building the suffix tree: The suffix tree is populated with pairs (paragraph, document id). The leaves of the suffix tree (the end of a suffix) contain the document ids of all the documents with that suffix from the tree root to that leaf.

3) Scoring the suffixes: Once the construction of the suffix tree is complete, each leaf node of the tree will have a list of documents ids. Each document id represents a document whose text contains the string suffix formed by following the tree from the root to the leaf. All documents ids in the same leaf in the suffix tree become part of a base cluster, one base cluster for each leaf. Note that a document will likely belong to more than one cluster.

Then each base cluster is given a score, derived from the corresponding suffix. The cluster score represents the "appropriateness" of the suffix as a description of the documents whose ids are in the leaf (cluster). The heuristic formula used by Zamir to score the base clusters is:

$score(B) = |B| \cdot f(|P|)$, where

- $|B|$ is the number of documents in base cluster B
- $|P|$ is the number of words in P that have a non-zero score (zero-score words: stopwords, too few(<3), or too many(>40%))

4) Cluster the base clusters: In the last step the base clusters are compared, grouping those clusters that seem to describe very similar document sets. In Zamir's paper, two base clusters, B_1 and B_2, are put together in the same composite cluster when $|B_1 \cap B_2|/|B_1| > 0.5$. and $B_1 \cap B_2|/|B_2| > 0.5$.

An important concern when clustering using STC is what part of the document to cluster: every word sequence that appears often enough becomes a potential cluster. For example, university names may become clusters even though we are only interested in clustering papers based on their content, rather than on author's affiliation.

2.4.6 Other Clustering Algorithms

As we said before, the number of clustering algorithms that can be found in the literature is bewildering. What we have described here are the ones that are found most in the IR literature, and the ones relevant in the context of our research. A final category of clustering algorithms we would like to discuss involves those designed to be used in interactive systems. Clustering is almost always an intensive process: hierarchical clustering, one of the most popular clustering techniques, for example requires:

- Step 1. $O(|D|^2)$ to compute the document/document similarity matrix +

Start by assigning each document to a cluster, so that if you have N items, you we have *D* clusters, each containing just one document. Let the distances (similarities) between the clusters the same as the distances (similarities) between the documents they contain.

- Step 2. $O(|D|)$ to locate the closest (most similar) pair of clusters and merge them into a single cluster. After this step we have one less cluster.
- Step 3. $O(|D|)$ to compute distances (similarities) between the new cluster and each of the old clusters.
- Step 4. Repeat steps 3 and 4 until all items are clustered into a single cluster of size D.

In total, we have: $O(|D|^2) + O(|D|)[O(|D|)+O(|D|)] = O(|D|^2)$

Traditional algorithms in general have an order of $O(|D|^2)$ at least. In practice, this makes them unsuitable for massive data sets. Two algorithms with low asymptotic running time, and thus more appropriate to the context of interactive systems, are Fractionation and Buckshot.

2.4.6.1 Fractionation

Fractionation was developed by Cutting et al. [Cutting92] with the objective of extending hierarchical clustering methods to massive data sets. It works by fixing a parameter called *G*, the number of clusters. Another parameter, *M*, is the maximum size for a dataset of $|D|$ documents to be clustered using hierarchical clustering in a reasonable amount of time. The algorithm proceeds as follows:

1. Split the data into subsets or fractions of size *M*.
2. Cluster each fraction into a fixed number αM of clusters, with $\alpha < 1$. Summarize each cluster by its mean.
3. If the total number of cluster means is greater than *M*, return to step (1), with the cluster means taking the place of the original data.
4. Group the cluster means into *G* clusters.
5. Assign each individual observation to the cluster with the closest mean.

The number of fractions in the i^{th} iteration is $\alpha^{i-1}|D|/M$ and the work involved in clustering a fraction is $O(M^2)$, which is independent of $|D|$. This shows that the complexity of Fractionation is linear in $|D|$ [Cutting92].

2.4.6.2 Buckshot

The Buckshot algorithm [Cutting92] also is based on performing hierarchical clustering on smaller sets, and like Fractionation, has a linear complexity. The goal of Buckshot is to get K clusters from $|D|$ documents in $O(K|D|)$ time. Buckshot (and Fractionation) were developed as part of the implementation of the Scatter/Gather technique to interactively use clusters as a way to improve document retrieval.

The steps of Buckshot are:

1. Randomly select d documents where d is $\sqrt{K|D|}$.
2. Cluster these using any hierarchical clustering algorithm to generate K clusters. Hierarchical clustering takes $O(N^2)$, so this step takes $O(K\sqrt{|D|}^2) = O(K|D|)$. We now have K clusters.
3. Compute the centroid of each of the K clusters.
4. Scan remaining documents and assign them to the closest of the K clusters.

These steps require $O(\sqrt{K|D|}) + O(K|D|) + O(|K|)\ O(|D| - \sqrt{K|D|}) = O(|D|)$ linear time over the number of documents.

2.5 Knowledge Discovery

Researchers traditionally spend many years learning and mastering an area of science, in order to acquire a body of knowledge that allows them to select a promising area of research. Without this knowledge, the scientific process would be a blind travel through all possible solutions or explanations, in search of the very few which are valid and which advance the state of knowledge in that area of science. It can be claimed [Valdes-Perez99] that scientific research is thus guided by heuristic rules, many times not explicit or clear, but nevertheless useful when assessing the potential of a certain approach to the research question to answer. The field of artificial intelligence tried during the 1980s to formalize the translation of knowledge from the researcher to a formal set of rules that could be followed by a computer, in practice applying the knowledge of the scientist. This approach failed, since as we said above many of these rules are not explicit or even clear to the researcher, or are not so well-defined as to be applied by a computer without lots of context. However, it is possible to combine the capability of computers to perform exhaustive searches over the solution space, and then present a candidate subset of solutions to be analyzed by the researcher. This is the approach taken by the field of

knowledge discovery. There is a small group of successful knowledge discovery systems, like MECHEM [Valdes-Perez94], Arrowsmith[Swanson2001] and CARTwheels [Ramakrishnan2004], that find patterns in data, aided by specific knowledge of the structure of the solution space. [Valdes-Perez99] is a good (if not totally up-to-date) survey of the field of computer-aided knowledge discovery.

In particular, CARTwheels and Arrowsmith are two knowledge discovery systems that are closely related to our problem of finding connections. CARTwheels builds connections among topics, that they call story-telling (similar to pathways in our approach). It works under the formalization of objects of interest described as multiple tuples of attributes, where each tuple belongs to one description set. CARTwheels then finds set of common objects that have description attributes in different description sets. Arrowsmith is even closer to our objective of finding connections by automatically using the content in a collection of scientific papers. Because of that we will explore here the field of literature-based discovery in more detail.

2.5.1 Literature-Based Discovery

There have been studies in the past of situations similar to our approach under the name of "literature-based discovery" [Swanson87] [Swanson89] [Swanson91] [Swanson2001] [Webber2001]. These studies proved that it is possible to build meaningful connections among seemingly unrelated concepts or document sets.

There have been analyses [Swanson86] [Spasser97] supporting the results and examining their impact. The main difference between these studies and ours is that they depend on the extensive use of a pre-existing classification system and an accurate classification of documents. Approaches that relied more on free-form text [Gordon1998] [Lindsay99] needed the involvement of user experts who know a great deal about the connections sought after. Furthermore, chains of relationships previously studied have been limited to one intermediate step, with very specific types of relations, like for example, illness→ (symptom, effect)←treatment.

Of particular interest is the work of Don Swanson in the Arrowsmith system [Swanson2001]. Arrowsmith is used to detect indirect relationships between topics in PubMed, by finding common keywords between two document sets. In Arrowsmith the user submits two ranked lists of documents called the A and C-Lists, that are the result of querying PubMed for the two topics in which the user is interested. From these lists, Arrowsmith retrieves the document text and classification metadata, and finds common keywords and phrases appearing in both document lists. For all common keywords (called the B-List), Arrowsmith will then match a word in a document in the A-List that is related to a word in the B-List, and then for the same word in the B-List will pick a word in the C-List. Thus Arrowsmith produces indirect relationships of the form: *term in A-List* ↔ *common term* ↔ *term in A-List*.

Arrowsmith was tested by using it to identify viruses that can be used as biological weapons, under the assumption that if virus X can be used as a weapon, and virus Y is

very similar to the characteristics of virus X, then virus Y also can be used as a weapon. The documents to retrieve are taken from the PubMed database of medical articles. The seed set (the X viruses) to extend are taken from the intersection of the viruses in the CDC list of biological weapons, and the ones in the Gessler report, an extensive compendium of biological weapons. From that set, they found relationships: virus Y ↔ common characteristics ↔ virus X. Since viruses that can be used as weapons must be dangerous, transmissible, and deliverable, the characteristics sought in common were related to these concepts, through the use of the MeSH controlled vocabulary that is part of PubMed. The A-List consisted of documents concerning genetic aspects of viruses, while the C-List concerned the transmission of viral diseases. They found that using Arrowsmith, and starting from the seed set, they could identify most viruses that were classified by experts as "weaponizables", and included in the Gessler list.

2.6 Citation Analysis and Information Retrieval

The use of citations as a means to find relationships among documents in a professional field originated in the area of library and information science, as a means to quantify the importance of authors and journals, and to discover patterns and trends within professional fields. Citation analysis is based on "a hypothesis that any act of citing the author of an earlier paper is always meaningful" [Sengupta92].

The first comprehensive use of citations as a means to explore the research literature was developed by Eugene Garfield in the form of the ISI Science Citation Index. The explosion of scientific literature after World War II, made (and still makes) it very difficult for a researcher to be aware of all the current literature in his or her field of research. In particular, tracing references backwards or finding who referenced a certain paper (tracing forward) was extremely cumbersome, given the lack of automated tools and comprehensive indices. Garfield's purpose in developing the Science Citation Index was that:

> "…the system would provide a complete listing, for the publications covered, of all the original articles that have referred to the article in question. This would clearly be particularly useful in historical research, when one is trying to evaluate the significance of a particular work and the impact on the literature and thinking of the period." [Garfield55]
>
> "In a way such listings would provide each scientist with an individual clipping service. By referring to the listings for his article, an author could readily determine which other scientists were making reference to his work, thus increasing communication possibilities between scientists. It is also possible that the individual scientist thus might become aware of implications in his studies that she was not aware of before." [Garfield55].

The Science Citation Index was a new idea at a time (1963) when unskilled employees could type the references, while the algorithms would perform the computation to find relevant authors and new areas of science.

2.6.1 Automating the Indexing of Citations

Nowadays, the availability of the Internet and the World Wide Web has made it possible to have electronic access to research papers that before were only available in printed versions. In addition, self-publishing of preprints has helped accelerate the knowledge dissemination process. The best example is ArXiv E-Print Services [Ginsparg96] (originally based at Los Alamos National Laboratory, now at Cornell), which originally started covering papers about different areas of physics, and then expanded into including mathematics and computer science, among other research fields. In physics in particular ArXiv has changed the way the dissemination of cutting-edge ideas and result is carried out; researchers go to ArXiv to keep up to date, and to the traditional research publications and journals for peer-reviewed material. ArXiv detects citations in a paper, and references citations to index records.

DBLP [Ley95] [Ley2002] is a citation-only computer science bibliography site that originally indexed database and logic programming papers, but later expanded to all areas of computer science. They found the biggest problem for automatic citation matching is the matching of author's name in all the different forms, in order to avoid duplication.

CiteSeer [Giles98] [Lawrence99] is the most comprehensive database of computer science research papers built from public sources. CiteSeer crawls university and research web sites, downloads papers, and automatically extracts and indexes metadata and citations to index. It has proven extremely succesful as a research tool, despite the error rate in matching different citations of the same paper. They addressed the problem of citation matching by considering references as one long string (without identifying particular fields inside it), and matching words and phrases. Phrases are consecutive pairs of words inside the same part of the citation. References are sorted by length, and consecutive references with similarity above a threshold are considered to refer to the same paper.

2.6.2 Reasons for Citation

A citation is a mechanism to acknowledge previous work and provide support for specific statements. The particular reason why an author cites a paper, however, is far more complicated and usually not explained in the publication. There are different points of view of the reasons for an author to include a citation. The first point of view about the reasons for citation considers science as devoid of human conflict, and citation as a mere method to support or contradict knowledge. The following list of Garfield's 15 reasons

for providing references in scientific papers [Garfield65] provides a list of reasons for citation under this idealistic point of view of science:

1. Pay homage to pioneers
2. Giving credit for related work
3. Indentifying methodology, equipment, etc.
4. Providing background reading
5. Correcting own's work
6. Correcting the work of others
7. Critisizing previous work
8. Substantiating claims
9. Alerting to forthcoming work
10. Providing leads to poorly disseminated, poorly indexed or uncited work
11. Authenticating data and classes of facts – physical constants, etc.
12. Identifying original publications in which an idea or concept was discussed.
13. Identifying original publication or other work describing an eponymic concept or term as, e.g. Hodgkin's Disease, Pareto's Law, Friedel-Crafts reaction, etc.
14. Disclaiming works or ideas of others (negative claims)
15. Disputing priority claims of others (negative homage)

Table 2.5. Garfield's 15 reasons for citing.

Another interpretation of the reasons for citation acknowledges science as a human affair, where conflicts of interest, favoritism, and ego play a role. Under this interpretation, [Thorne77] lists mechanisms for using citations as a way to improve the chance of publication of a paper. The following is a summarization of Thorne's "tips" as in [Larsen2004]:

1. Hat-tipping citations
2. Over-detailed citations (cover all the bases)
3. Over-elaborated citations (overwhelm)
4. Citations chosen to provide evidence supporting the author's point of view
5. Self-serving citations
6. Deliberate premeditation (conscious playing of the citation game)
7. Citation as projective behavior (citations as reflections of author's bias)
8. Conspiratorial cross-referencing (authors citing each other to boost the citation count)
9. Pandering to pressures (citing works because one is expected to)
10. Intra-professional feuding
11. Political consideration (following the party line)

Table 2.6. Thorne's reasons for citing.

The true reason why a writer chooses to add a particular citation lies, of course, between the two extremes. However the human factor cannot be denied. Simkin and Roychowdhury in "Read before you Cite!" [Simkin2003] conducted a study regarding the propagation of citation errors about a famous 1973 psychology paper, finding that there are patterns in which exactly the same error in a citation of the paper (like

misspellings of the author's name) appears time after time. They argue that the exact error propagating over time is a proof that in each case the citation was "lifted" from another publication where the same mistake was made, without the author actually reading the paper. They estimate the ratio of the number of actual readers of the paper over the number of mere citers as the ratio of distinct misprints over the total number of misprints, assuming that the first person who made a mistake in citing, made it from the original paper. The authors found that misprint frequency follows a Power Law distribution, and they conclude that the reader/citer ratio is about 23% (only about 23% of the authors citing a paper actually read it, the rest copied the reference).

2.6.3 Citations, Co-citations and Bibliographic Coupling

The aggregation of references in a single Science Citation Index made possible research in many areas of bibliometrics that were impossible to explore in practice before. In particular the work at ISI explored the importance of different types of relations among documents established by citations. The three different types of citation relationships between documents are:

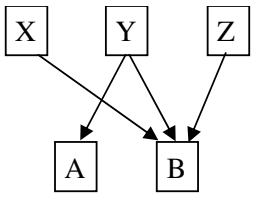
Direct citation: Document A has a reference to document B. Within a document collection, the Journal Impact Factor (JIF) [Garfield72] is a measure of the importance of a document in the collection over a period of time that uses citation count. Commonly it is defined as the ratio between the number of citations of a paper in a period of time over the total number of citable papers in the same period of time, usually two years.

Bibliographic Coupling [Kessler63]: Documents A and B are bibliographically coupled if both have a reference to a document C. The assumption behind bibliographic coupling is that "the number of references that a given pair of documents have in common is a measure of their proximity of subject" [Kessler63]. Bibliographic Coupling is time-invariant; once papers are published their references do not change.

The most common measure of bibliographic coupling is:

$$\text{coupling\_strength (A,B)} = \frac{\left|references(A) \cap references(B)\right|}{\max(\left|references(A)\right|, \left|references(B)\right|)}$$

Note that coupling strength is normalized to take into account the variations in the lengths of reference lists, because as documents have more citations, the chance of having a reference in common increases.



coupling(X,Y)=0.5
coupling(Y,Z)=0.5
coupling(X,Z)=1

Figure 2.5. Example of Bibliographic Coupling. Documents X, Y and Z cite documents A and B. The coupling strength for each pair is shown to the right of the diagram.

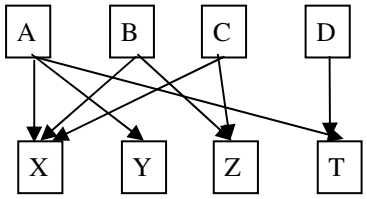
Co-citation [Small73]**:** Papers A and B are co-cited if both are referenced by document C. Co-citations' claim is that "co-citation strength is indicative of strength of intellectual connections" [Garfield88].

The most common measure of co-citation is Garfield's co-citation link strength:

Co-citation Link Strength(A,B)= $\frac{X}{Y-X}$, where

X=total number of co-citations of A and B,
Y= total number of citations of A + total number of citations of B

In this case, the total number of co-citations of A and B is normalized by bringing into account the total number of times that the documents have been individually cited. The Co-citation Link Strength is equal to 1 in the extreme case where A and B are always cited together.



Link strength(X,Y)=1/3=0.33
Link strength(Y,Z)=0
Link strength(Z,T)=0
Link strength(X,T)=1/4=0.25

Figure 2.6 Example of Co-citation. Documents X, Y, Z, and T cite documents A, B, C, and D. The link strength for each pair is shown to the right of the diagram.

The main difference between co-citation and bibliographic coupling is that bibliographic coupling is static (once a document is published, its references do not change), while co-citation measures the frequency in which a pair of documents is cited together over a period of time.

The main use of co-citations in information science is to cluster documents and authors in a particular domain in order to understand relationships in the domain. Author Co-

citation Analysis (ACA) [White81] is a particular way of mapping a research field, where authors are the data points. To perform ACA, each of N authors becomes an index in a N×N matrix. An entry (*i,j*) in the matrix is the co-citation count of authors i and j. From the co-citation matrix, the strength of co-citation between authors *i* and *j* is Pearson's r between the rows *i* and *j*.

From the point of view of information retrieval, citations are an additional source of evidence of relationships between two documents, between two clusters representing topics within an area in a professional field, or to assess the importance of documents. While we do not make use of ranking by citations, we will briefly comment on the underlying ideas behind it.

The concept behind ranking by citations is that the number of times a document is cited (or in the case of the WWW, linked to), is a measure of the importance of the document. This concept can be extended to say that the importance of a document is a function of the importance of all other documents that cite it.

2.6.3.1 PageRank

The PageRank algorithm [Brin98] works by calculating an a-priori collection-wide, search-independent importance value for each page indexed. PageRank works by considering the WWW as a massive Markov Chain, where each page is a node, each link is an edge, and the transition probability from one page *X* to another is 1/*O(X)*, where *O(X)* is the number of outgoing links in *X*. Weight propagation follows the model of a user who goes from one page and will select a next page with probability 1/*O(X)*, except for a small probability that before each transition the user will chose another page randomly. Pages without outgoing links (sink nodes) are considered to have an equal transition probability to all other pages. Then the PageRank *PR()* of a page *P* is defined as:

$$PR(P) = d + (1-d)\left[\frac{PR(P_1)}{O(P_1)} + \frac{PR(P_2)}{O(P_2)} + ... + \frac{PR(P_n)}{O(P_n)}\right]$$

where

- *P* is a web page
- P_i are other pages that link to P
- $O(P_i)$ is the number of outgoing links of P_i
- *d* is the probability that a user will randomly select another page instead of following a link
- *N* is the total number of indexed pages

Calculating the PageRank of each page is done iteratively, by propagating weights from one page to another, and repeating the process a number of times.

2.6.3.2 HITS

HITS [Kleinberg99] also ranks web pages, but with a completely different concept. HITS considers the graph to have two important set of pages: hubs and authorities. Hubs are pages that link to other important pages. Authorities are pages that are linked to by other important pages. In the HITS algorithm, each page in the web graph *G(V,E)* receives a *hub* score and an *authority* score. For a page *i*,

$$hub(i) = \sum_{(j,i)\in E} \frac{authority(j)}{\left|\{(j,i) \mid (j,i) \in E\}\right|} \;,\; authority(i) = \frac{\sum_{(i,j)\in E} hub(j)}{\left|\{(i,j) \mid (i,j) \in E\}\right|}$$

The intuition behind these formulas is that important hubs point to pages with high authority values, and important authorities are linked to by pages with high hub values. Calculating the hubs and authorities matrices is again an iterative process.

2.7 Interactive Retrieval by User Manipulation of Query Results

A refinement of search results beyond the measure of relevance used by the IR system may come from two sources: training data or user feedback. Given that for many real-life collections there is no relevance judgment set available for training, involving the user in refining the query becomes the way in which the system can improve its understanding of the user's information need, disambiguating what was expressed in the query formulation. When we talk here about query refinement we are not talking about automatic query expansion techniques like Local Context Analysis [Xu96] or Phrasefinder [Jing94], which are used to reduce the vagueness of a query, but of techniques to refine the understanding of the user's need.

The simplest type of feedback can support querying and browsing, not only of the immediate search results, but also of documents related to search results. Halasz in his seminal paper "Reflections on Notecards: Seven issues for the next generation of hypermedia systems" [Halasz88] argued for the need of combining browsing and searching as part of the information finding process, that can be seen nowadays partially implemented in some web search engines in the form of the "see similar pages" link under each of the search results.

From a strict IR point of view, Rocchio's algorithm introduced in SMART [Rocchio71] gave an empirical proof that user relevance feedback on the top retrieved documents improves precision. User's feedback can be understood as relevance judgments, and therefore as training data. For example, [Williamson2000] uses the relevance judgments to train a neural network to find the relevant/non-relevant separation surfaces (hyperplanes) as the user browses and judges the results of each retrieved set.

The IR community has adopted a feedback view restricted to relevance information related to the query. Consequently, the work of Williams in RABBIT [Williams94] is significant to our project. RABBIT adopts a wider view of query reformulation in the context of semantic networks, by allowing the user to qualify retrieved items not only as "required" and "prohibited" (relevant/non-relevant), but also by letting the user request a specialization of a part of the query (part-of or contained-in relation, asking for a more restricted interpretation), or to request alternatives to a part of the query (sibling-of relation). These kinds of relationships have been surprisingly ignored in the IR field, but there is nothing that prevents them for being applied: a specialization is a request for more restrictive interpretation, while alternatives are, from an IR approach, topics that co-occur with some item subsuming the part under feedback. Scatter/Gather [Cutting92] [Hearst96] and the work of Sanderson and Croft in building text hierarchies [Sanderson99] partially go beyond relevance. The work on building text hierarchies offers an effective method to automatically discover specialization and containment relationships.

Scatter/Gather addresses disambiguation by offering two operations: scatter, which divides a document cluster into sub-clusters, and gather, which combine clusters. This operation can, in the framework of RABBIT, be interpreted as generalization (scatter) and specialization (gather) operators. Furthermore, it is very important that these operators are closed (the parameters and the result are of the same type) and so can be freely combined. Scatter/Gather's retrieval process consisted of the following steps:

1. From a user query, retrieve a document set. Quickly cluster the document set.
2. The user identifies what clusters are too generic, and asks for them to be "scattered" (split), and which ones are too particular, and asks for them to be "gathered" (joined).
3. A new set of clusters derived from step 2 is presented to the user, and the process goes back to step 2 until the user is satisfied with the result.

The Scatter/Gather approach relies on clustering as a way to allow the user to separate a subset of the documents determined as relevant by the system. Document clusters in Scatter/Gather are labeled by the set of most popular terms in a cluster. While the user can easily group related documents by him or herself when the list of documents is short, as the list gets longer he can no longer identify all documents related to a relevant document without spending a significant amount of time reading all documents in the list. Clustering is a way of automatically identifying these related documents. Since automatic clustering is bound to be imprecise, the scatter/gather techniques are a way of allowing the user to participate in the clustering process.

A user study of Scatter/Gather [Hearst96] showed that users relied mainly on "scatter" and rarely on "gather". Starting from the biggest set in terms of documents, they drilled down until they felt the cluster was specific enough and then they could browse through the documents in the set to identify the relevant ones. Another study [Pirolli96] showed that Scatter/Gather did not perform better than a standard, ranked-based retrieval tool, but provides a useful service as a tool to explore the document set.

Nevertheless, Scatter/Gather showed that trading a 'good enough' approach to document clustering for user interactivity is possible and can in fact improve retrieval.

2.8 User Interfaces for IR Systems

Since the availability of IR systems increased rapidly in the 1990s, it has become apparent that the traditional ranked list presentation of search results is not enough for the user to understand the collection and have an effect on the retrieval process. As such there have been design and evaluation studies of alternative user interfaces for query formulation, display of query results, relevance feedback, and exploration of the collection structure. Early attempts like AI-STARS [Anick90] provided a way for the user to explore variations of the formulation of a boolean query without manual rewriting, using spatial arrangement to represent synonyms and phrase order. VQuery [Jones98] provided a graphical set representation to express queries, while the Lexical Navigation System [Tunkelang97] allows the user to explore the relation between concepts in a collection, and use this network to select terms for query refinement.

Our interest is centered on the representation of collections, query results, and user feedback. Most collection representations adopt one of many spatial metaphors, mapping different collection characteristics to spatial attributes like position, height, or distance. When representing collection content, Kohonen's self-organizing maps [Kohonen95] and Multi Dimensional Scaling [Chatfield80] have been used [Chalmers92] [Lagus96] for their ability to combine proximity and area size as a representation of semantic relation and importance.

SPIRE [Hetzler98] is a particularly successful application of spatial metaphors in an integrated system for IR. SPIRE combines 2D (Galaxies) and 3D (ThemeView) representations to provide document/topic and relative topic importance views, in the context of a full indexing and retrieval system.

The intuition that 3D spaces, because of their capability to represent more information than 2D, would be more suitable for the design of direct manipulation interfaces, have produced a number of systems that try to materialize this advantage. VR-Vibe [Benford95] mapped all document positions relative to up to 3 topics simultaneously, compared to 2 in a traditional interface. Aleph [Das-Neves97] generalized the model to more than 3 topics using a layered approach. Liberworld [Hemmje94], and Nirve [Sebrechts99] adopt a 3D-party "world" approach, where documents are mapped in a sphere (using the volume in Liberword, using the surface in Nirve), and alternative tools and markup are used to show document relationships and allow browsing. The work of the Xerox Group in Cat-a-Cone [Hearst97] and Hyperbolic Views [Lamping95] is probably the most successful and extensive work on 3D interfaces for the exploration of document collections, with Hyperbolic Views proving to achieve a very convenient compromise between detail and context when showing relations among document as a network.

However, except for a few cases, the superiority of 3D interfaces for IR over 2D interfaces has not been experimentally confirmed, with comparative studies like the one in Nirve [Sebrechts99] finding no significant advantage for the 3D representation versus the 2D representation, at the cost of a higher learning curve. Our experience with an immersive interface for Digital Libraries [Das-Neves2000] confirms this finding. This is why our approach will be more in line with 2D network representations, like LiveTopics [Bourdoncle97] and [Fowler91], and integrated interfaces for topic exploration and browsing.

Simple techniques like feedback for query matching in document structure as presented in TileBars [Hearst95] have been tested and proven [Hearst96] to be viable representations a user can understand and use to improve precision. There are systems like Envision [Nowell93] [Nowell96] that, using a 2-dimensional scheme, are able to represent as many dimensions as other 3D systems (see Figure 2.7), with a tested and proven increase in usability and user satisfaction while exploring collections. 2D circular views like the Passage Correlation diagrams in [Salton95] also compete with 3D world views with more ease of manipulation.

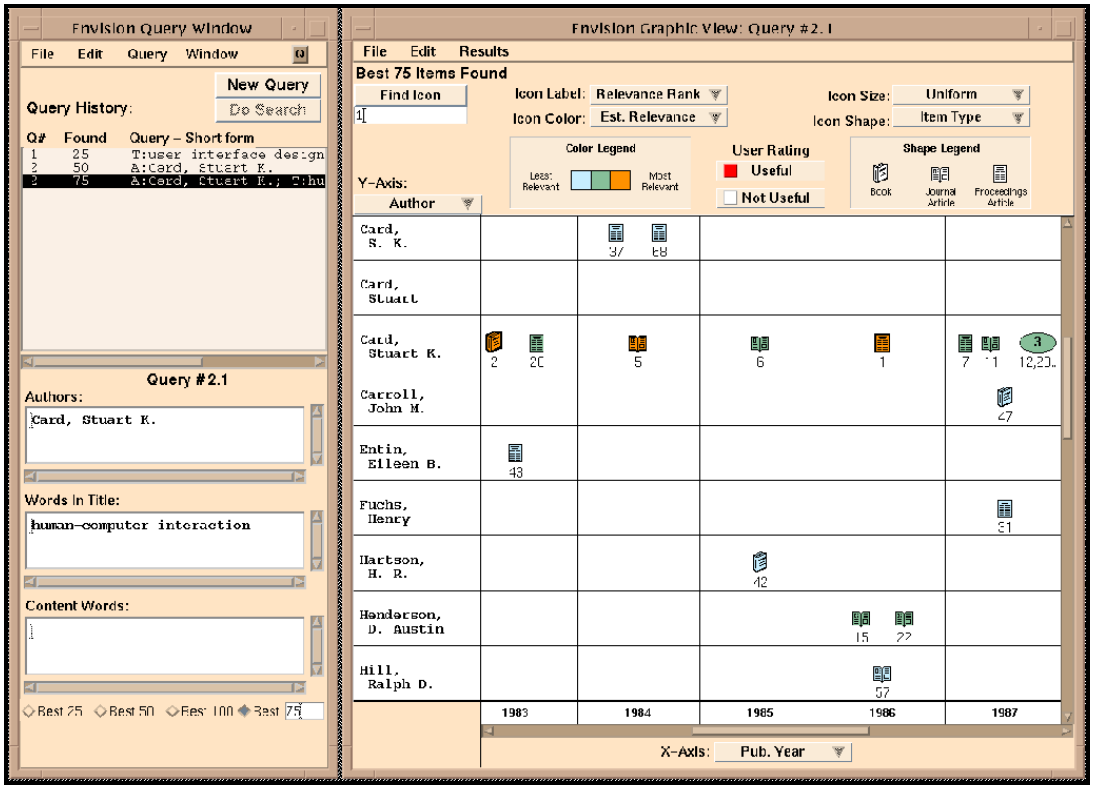


Figure 2.7. An Envision session.

Worth mentioning are the approaches that attacked the problem of displaying hypertext networks, with the enormous complexity of their high connectivity. Most of the approaches work by creating a hierarchy of connected clusters, and visualizing them in different ways. An early example of this technique is Nested Fisheye Views [Noik93],

which applies a fisheye view technique to the cluster hierarchy. Multiple Hierarchical Views [Mukherjea95] are a further elaboration of the nested or hierarchical representation in Nested Fisheye Views, combining structural and content analysis, and allowing the user to choose different criteria to build the hierarchies.

2.9 Conclusion

In this chapter we have covered different aspects of information retrieval that relate to Stepping Stones and Pathways. We started with the Vector Space Model, the most successful model for document representation and base to many other models. Then we continued with Bayesian networks, a method to consider IR within a probabilistic framework, that we use in Stepping Stones and Pathways to combine different sources of evidence and to calculate scores for pathways.
Since Stepping Stones contain groups of related documents, we overviewed many clustering techniques of frequent use in IR, with special emphasis on suffix-tree clustering (STC), that we use to create the stepping stones.
Stepping Stones and Pathways uses content and citation evidence to find the similarity between stepping stones and documents. Therefore we covered citation analysis, methods to calculate document similarity and importance using citations, and HITS, a clustering method based on citations.
We finished by giving an overview of user interaction in information retrieval, in terms of involving the user in the retrieval process, and in terms of interfaces for information retrieval systems. We demonstrate why 3D interfaces have not shown the advantage they seem to have in terms or richness of information manipulation, and we briefly describe user interfaces that have inspired the design of the Stepping Stones and Pathways user interface.

3. A Collection to Test Stepping Stones

Summary

This chapter describes the reasons why we harvested a document collection to test the results of Stepping Stones, the methods used to harvest the collection, and results in terms of topics and documents.

3.1 Requirements for a collection to test Stepping Stones and Pathways

There are 4 requirements for a document collection to be effective in helping the researcher to test the effectiveness of Stepping Stones. These requirements are:

1. **The collection must have the documents labeled with topics, or must have split queries:** In order to test the results of Stepping Stones, query pairs are needed. It has to be possible to answer the relation between the two parts of the query with the content of the collection. One way to do this is to have split queries, or queries than can be split with reasonable certainty that the splitting did not change the meaning of the query. Another way is to have the documents grouped into topics, and use pairs of these topics as query pairs.

2. **Topics must be demonstrably connected or related:** Queries must have an answer within the collection. In order to do this, since a query with two topics (X,Y) is going to be interpreted by Stepping Stones as "how is X related to Y?", topics X and Y have to be demonstrably connected within the collection, either by analysis of content or structure (links).

3. **Queries must have a proven set of valid relations within the collection, or related topics and document content must be understandable by the subject pool available:** The final mandatory requirement is that it must be possible to verify the quality of the results of Stepping Stones and Pathways. This could be done by using a set of relevant connections given by some other source (such as previous experiments), or by subjects validating the connections. Subjects that participate in the experiment have to be able to recognize and understand a valid relation between topics in a query in order to declare it relevant or not.

Optionally:

4. **It is desirable that documents have references (links):** Although this is not a mandatory requirement, having document references available, either in the form

of citations or links, simplifies the analysis of the connectedness of topics, and provides another source of evidence at the moment of finding related documents.

3.2 Candidate Collections

The information retrieval field has a number of test collections available. These collections have been gathered either with the purpose of testing IR methods/systems with the traditional measures of precision and recall, or with the purpose of testing the quality of clustering and routing algorithms. Here we mention some of these collections, and the reasons why they were not suitable to test Stepping Stones.

CACM Collection: The CACM collection [Fox93] is comprised of 3,204 titles and abstracts with two different 50 query sets. It also includes reference, co-citation, bibliographic coupling, category, and some other data. This collection however does not contain topics, and the abstracts are very short (or absent). The problem of document text being short is that while it does not take much text for a person to get an idea about the relevance of a document to a query, it makes more than the title and a few paragraphs for the person to understand how the document is related to other documents.

TREC: The TREC collection is composed of different document collections. Of particular interest to our study are the queries and documents involved in the Short Query Track [Buckley2000] in TREC. The Short Query Track analyzed the behavior of different retrieval systems when the information need is expressed in only 2 or 3 words. This scenario is more realistic than the free-form queries in other TREC tracks, that can consist of more than 20 words. The Short Query Track also included the queries submitted to the participating systems. The collections used in this track were a group of articles from the Wall Street Journal, Associated Press, and Ziff-Davis; in total more than 500,000 documents. One problem with this data is that there are is no citation, co-citation, or bibliographic coupling data available. Another problem with using these in our experiments is that these TREC collections do not contain topics, and the queries are about only one subject. We performed some experiments with the query formulations of topics 51-100 to see if it was possible to combine different pairs of topic formulations of the same subject in one query. We compared the results obtained with the two queries combined, and then compared the results to the individual formulations. Our objective was to try to identify what causes the short pairs of queries to perform better than the combined query, since these causes would give us hints into what characterizes good query splitting. Although about 8 to 10% of the total pairs showed some improvement in the precision when queries were not combined, we could not identify any pattern that explained at least most of these cases.

REUTERS: The Reuters collection [Lewis96] is the closest to our needs. It consists of 21578 news articles donated by Reuters. Articles are classified in categories, with some of them belonging to more than one category. The main problem with using Reuters for our experiments is that is not a focused collection, and we are not aware of any studies of

the connectedness of topics in the collection, thus making impossible to judge whether a any pair of topics in Reuters can provide a successful pair of sub-queries for Stepping Stones. Further, there was no link data available to relate the articles.

PubMed: PubMed is a free, publicly accessible database of medical articles from 1966 onwards. Articles are drawn primarily from MEDLINE and PREMEDLINE. In addition, some participating medical journals have their abstracts included in PubMed. PubMed includes title, abstract, and references, and its size allows one to harvest under many different criteria, thus allowing the extraction of a focused subset of articles that enhances the chance of connectedness among topics. This is the collection used in almost all of Swanson's experiments. The main obstacle for us is the lack of a subject pool available, since papers in PubMed are very technical works, that cannot be fully understood by a person without medical training.

3.3 Building a Test Collection for Stepping Stones

Since no available collection covered the requisites of having documents grouped into topics, and proven relations between topics, we developed a strategy to build such a collection: harvest many course syllabi about a very few areas of science. The course that a suitable syllabus describes must be based on papers, not on a textbook, and must have the papers grouped into areas. Starting from such syllabi, we gathered the papers, and followed the citations in the papers to expand the collection. We used CiteSeer as a source of papers, since it simplified our job by gathering in one place thousands of papers available online, extracting the references for us. To satisfy requisite 2 of the list given in Section 3.1, we use the citations and topics listed in the syllabi to analyze whether the documents in the collection are connected. To satisfy requisite 3, we gather syllabi from computer science courses, since that gives us the biggest pool of suitable subjects available.

3.3.1 Gathering Syllabi

In order to gather a collection of scientific papers that were classified in topics and are available online, we searched for online syllabi of computer science courses. The type of syllabi we were interested in have the following characteristics:

1. They are relatively new (1999 and newer) so the syllabi will not differ just based on the availability of papers. Recent syllabi also are more likely to refer to newer papers, increasing the chance that they and their references are available online.
2. The course must be based on papers, not on a textbook, and the papers must be listed in the syllabi.
3. The papers must be available online in their majority, so we can harvest the papers and extract the text and citations.
4. Each syllabus must group the papers into areas, so we have an association between a paper and an area of the course.
5. The set of syllabi must be about a very small set of courses, so we have multiple syllabi for the same course.

Following these criteria we were able to get a set of 8 syllabi of Operating System courses, 5 of Data Mining, and 5 of Information Retrieval courses. The following tables describe the topics each syllabus covers.

Operating System Syllabi		
Syllabi Number	**Topics**	**Topic Count**
1	Advanced Communications, DFS and DSHM, Distributed Systems, Fault Tolerance, OS Structures, Object Representations, Protection, Real Time, IPC, Shared Memory	10
2	Overview, IPC, SMP	3
3	Bloom Filters, File Synchronization, Storage, Web Search, Web Server, Web Services, Distributed File Systems	7
4	Introduction, Distributed Systems, File Systems, Extensible OSs, Plan 9, Research Tools, Security and Authentication, Virtual Machines	8
5	History of OS, Kernel Structures, Experiences, Fault Tolerance, File Systems, Memory Management, Mobility and Power, Performance Evaluation, Scheduling and Synchronization, Security, Distributed Systems, IPC, Miscellanea	13
6	Client-Server Model, Communication in Distributed Systems, Distributed File Systems and Distributed Shared Memory, Processes and Threads	4
7	Introduction, OS Structures, Extensible OS, Research Tools, Storage Systems, Virtual Machines, Distributed Systems	7
8	File Systems and Distributed File Systems, Concurrency, Reliability, Virtual Memory, Network, Network and File Systems, Operating System Issues, Miscellanea	8

Data Mining Syllabi		
Syllabi Number	**Topics**	**Topic Count**
1	Credit Scoring Applications, Document clustering and classification, Modeling Consumer Transaction Data, Pattern Discovery Algorithms, Predictive Modeling, Scalable Algorithms, Web Mining	10
2	Introduction, Primitives for DM, Advances in Knowledge Discovery and DM, Applications and Trends, Classification And Prediction, Cluster Analysis, Concept Description, Data Preprocessing, Mining Association Rules, Mining Complex Types Of Data	10
3	Association Rules, Classification, Clustering, Database and Mining Integration, Deviation Detection, Extension to Association Rules, Parallel DM, Sequence Mining, Similarity	7

	Search, Spatial DM, Web Mining	
4	Overview, Association Rules, Bayesian Learning, Clustering, Data Warehousing, Inductive Logic Programming, Instance-Based Learning, Neural Networks, OLAP, Temporal Discovery, Text Databases,	8
5	Beyond Association Rules, Clustering and Segmentation, Combining Multiple Models, Database Marketing, Feature Selection, Imputation, Sampling, Scalability Issues, Miscellanea, Support Vector Machines, Web Mining and IR	13

Information Retrieval Syllabi		
Syllabi Number	Topics	Topic Count
1	Document Classification and Filtering, Document Clustering, Feature Selection, Latent Semantic Indexing, Performance Evaluation, Relevance Ranking, Vector Space Model, Web Search Engines	8
2	Co-Training Classification, Collaborative Filtering, Feature Selection, Hypertext Categorization, IR Models, Information Extraction, Information Filtering, Language Models, Model Selection, Music Retrieval, N-Grams, Nearest Neighbor, Question Answering, Unsupervised Clustering, Web Mining, Support Vector Machines	17
3	Overview, Distributed IR, Document Analysis, Hypertext and IR, Matching Techniques, Multimedia IR, Presentation of Search Results, Query Analysis, Relevance Feedback, System Evaluation	10
4	Overview, Information Retrieval, Cross-Language Text Retrieval, NLP and IR	4
5	Conceptual information retrieval , Auctions on the Web, Citation Databases, Clustering, Conceptual information retrieval , Distributed IR, Document Similarity and Plagiarism, Evaluation of IR performance, Formal models of Web queries, Image Search and DLs, Intelligent Agents in DLs, Metadata, Metasearch, NLP and IR, Privacy, Query Expansion, Recommender Systems, Relevance judgements, Scientific literature and Digital Libraries, Search source selection, Softbots, Video information retrieval, Web Usage Patterns, Web analysis search engine comparison, Web crawling, Web graph analysis, Web storage.	27

Table 3.1. Harvested syllabi, and their topics.

As can be seen, there are a few topics that are common to many syllabi (like Distributed Systems in Operating Systems, or Association Rules in Data Mining), but generally the selection of topics varies widely. Table 3.2 shows that most of the topics are unique to each syllabus.

Subject	Total Number of Topics	Number of Unique Topics
Operating Systems	60	46
Data Mining	48	44
Information Retrieval	66	59

Table 3.2. Total and unique number of topics per syllabi subject.

3.3.2 Harvesting Papers and References from CiteSeer

The syllabi provide a set of documents from which to build a collection. We used CiteSeer to expand the collection by following the references in the papers listed in the syllabi. CiteSeer served our purposes of extending the collection by having pointers to the original papers, assigning a limited set of IDs to each paper (thus simplifying the identification of papers), and providing the references already parsed and transformed into IDs.

We wrote a web crawler/scraper that automates the process of expanding the collection from the papers in the syllabi. For each paper in the syllabi, the web scraper searches for the paper title in CiteSeer's web site, and if the document is available, then it parses the web page to gather the metadata (publication, year, authors, etc.), references cited in the paper, and sources indicating where to look for the paper. CiteSeer's search capability was inconsistent, sometimes returning no results when the document could be found with slight variations of the query – one word more or less would make the difference between CiteSeer returning a successful match or failure. The web scraper recognizes the case in which CiteSeer does not return a result with the document sought. In that case the web scraper repeats the query in Google, limiting the search to CiteSeer's web address.

We harvested 3 levels of documents: the ones in the syllabi, the references of those papers, and the references of the referenced papers. Since one of the reasons behind harvesting our own collection is to guarantee connectedness of the topics the papers are about, we do not wander too far from the syllabi papers. Second, following references is going back in time, and as we go back in time it becomes harder to find the papers available online. The third reason is that the number of references grows very quickly, and simply following more and more references does not guarantee that we will gather a more connected set; in fact the citation matrix becomes more sparse.

From the syllabi we harvested the set of documents that we call the "seed subset", because it is formed of seeds that are used to extend each collection subset. The seed subset has the following characteristics:

Area	Syllabi	Total seed papers	Unique seed papers	Seed papers with text harvested	Seed papers with only text+abstract	Seed papers with metadata
OS	8	310	189	177 (93%)	0	177 (93%)
DM	5	160	136	125 (91%)	3	122 (89%)
IR	5	245	229	222 (97%)	27	102 (45%)

Table 3.3. Summary of success at extracting document seeds from the syllabi.

From the operating system, data mining, and information retrieval syllabi, we extended the collection to a total of 2173 operating system papers, 1876 data mining papers, and 1640 information retrieval papers, along with their metadata in almost all cases (except the ones mentioned above), and references in all cases. The number of references is of course greater than the number of papers we harvested; we only have those papers whose text is available from public sources, in particular CiteSeer.

Using the references in the seed papers we can build a map of related topics for each area.

To build a map of related topics, for all pairs of two topics *A* and *B*, and a paper p_A listed under topic *A*, and a paper p_B listed under topic *B*, we counted the cases in which:

- Topics *A* and *B* have a paper in common ($\underline{p_A}=\underline{p_B}$), and
- p_A cites p_B, or p_B cites p_A, and
- p_A and p_B, are co-cited by some other paper in the collection.

Figures 3.1 to 3.3 at the end of this chapter depict the maps of connected topics for Operating Systems, Data Mining, and Information Retrieval according to the topics and paper in the syllabi. These maps only show the topics that are related to some other topic in different syllabi by references; a map including all the topics in all syllabi is impossible to display. It is clear from the maps that topics are not disconnected; topics in different areas are related by references or documents in common.

3.3.3 Analysis of Relations between Syllabi and Topics

From the data that we have displayed in this chapter, there are some observations that are of interest to find relationships between topics:

- Relationships between topics are very rare and not random: most of the time, for a person with knowledge in the subject, it is possible to guess the relation just by looking at the topic titles.

- Most of the relationships have a link value of 1. Because of the type of references we counted, that guarantees some degree of semantic relationship between the

topics, but a link value of 1 cannot guarantee that the relationship can be found and explained just by following the references. There is always the possibility that the only citation connecting two topics is a citation that is not important enough in the context of the citing paper to bridge the topics (there are a variety of reasons why people cite papers, see "Reasons for Citation" in Chapter 2, "Literature Review").

- The higher link values are very often among variations of the same topic, as for example "Classification" and "Classification and Prediction" in Operating Systems, "Association Rules" and "Mining Association Rules" in Data Mining, "System Evaluation" and "Evaluation of IR Performance" in Information Retrieval, ...

- The selection of topics that cover a subject is very subjective. What is an important topic, and the coarseness of topics in terms of content, varies from one syllabus' author to another.

- The criteria used by authors to select papers that are representative of a topic are very subjective. Tables 3.1.a, 3.1.b, and 3.1.c summarize, for all pairs of the same topics or very similar topics in different syllabi, the number of documents in each topic, and the number of documents in common. As can be seen for the tables, different professors (the authors of the syllabi) had very different criteria of what papers are representative. Had they agreed on the papers they chose for each topic, then all the papers in the smaller of the two topics in a pair (table row) should be contained in the topic with most documents in the pair. This is clearly not the case in almost all topic pairs.

Operating Systems: Documents in Common for Same or very similar Topic				
Syllabus/ Topic	**# docs.**	**Syllabus/Topic**	**# docs.**	**# doc. in common**
1/ DFS and DShM	5	3/ DFS	3	1
1/ DFS and DShM	5	6/ DFS and DShM	3	1
1/ DS	2	4/ DS	4	1
1/ DS	2	5/ DS	12	1
2/ IPC	4	5/ IPC	5	1
3/ DFS	3	6/ DFS and DShM	3	1
4/ FS	8	5/ FS	9	4
4/ FS	8	6/ DFS and DShM	2	1
4/ FS	8	8/ FS and DFS	5	2
4/ Research Tools	3	7/ Research Tools	4	3
4/ Virtual Machines	2	7/ Virtual Machines	4	2
4/ DS	4	5/ DS	12	1
4/ DS	4	7/ DS	6	1
4/ Introduction	2	7/ Introduction	4	1

4/ Security and Authentication	10	5/ Security	13	1
5/ FS	9	6 / FS and ShM	2	2
5/ FS	9	8 / FS and DFS	5	2
5/ FS	9	8/ Network and FS	1	1
5/ Memory Management	2	8/ Virtual Memory	1	1
5/ DS	12	7/ DS	6	1
6/ FS and SHM	2	8/ FS and DFS	5	1
1/ DS	2	4/ DS	6	0
1/ Fault Tolerance	4	5/ Fault Tolerance	7	0
1/ IPC	1	2/ IPC	4	0
1/ IPC	1	5/ IPC	5	0
1/ShM	6	6/ FS and ShM	2	0
6/ Comm. In DS	4	7/ DS	6	0

Data Mining: Documents in Common for Same or very similar Topic				
Syllabus/ Topic	**# docs.**	**Syllabus/Topic**	**# docs.**	**# doc. in common**
2/ Mining Association Rules	7	3/ Association Rules	10	2
2/ Mining Association Rules	7	4/ Association Rules	1	1
2/ Classification and Prediction	5	3/ Classification	8	3
2/ Cluster Analysis	11	3 / Clustering	11	5
3/ Association Rules	10	4/ Association Rules	1	1
3/ Clustering	11	5/ Clustering and Segmentation	2	1
1/ Web Mining	1	3/ Web Mining	3	0
1/ Web Mining	1	5/ Web Mining and IR	3	0
2/ Cluster Analysis	11	4/ Clustering	1	0
2/ Cluster Analysis	11	5/ Clustering and Segmentation	2	0
3/ Web Mining	3	5/ Web Mining and IR	3	0
4/ Clustering	1	5/ Clustering and Segmentation	2	0

Information Retrieval: Documents in Common for Same or very similar Topic				
Syllabus/ Topic	**# docs.**	**Syllabus/Topic**	**# docs.**	**# docs. in common**
1/ Vector Space Model	5	2/ Language Models	3	3
1/ Vector Space Model	5	4/Information Retrieval	4	3
1/ Web Search Engines	1	3/ Hypertext and IR	5	1
2/ Language Models	3	4/Information Retrieval	4	3
3/ System Evaluation	12	5/ Evaluation and IR Performance	4	2
5/ Metasearch	3	5/ Search Source Selection	2	1
1/ Clustering	4	5/ Clustering	4	0
2/ Unsupervised Clustering	3	5/ Clustering	4	0
4/ NLP and IR	13	5/ NLP and IR	2	0

Tables 3.4.a, 3.4.b and 3.4.c: Number of Documents in Topic Pairs, and Documents in Common. These tables show, for all pairs of the same topics or very similar topics in different syllabi of the same subject, the title of each topic, followed by the number of documents, and the number of documents in common between both topics.

3.3.4 Extracting Plain Text from Papers

Scientific papers available online come in a variety of file formats and formatting options. In order to extract the text, we need to find the file format, apply the proper extraction tool, and post-process the extracted text to fix errors in extraction, such as ligatures and hyphenated words.

The most common file formats for scientific papers are PDF, Postscript, LaTEX, and HTML. In order to find the file format, we rely on a quick classification by filename extension, followed by analysis of the file content if the text extraction fails. The rules to identify a file format from the file content are similar to the ones in the Unix `file` command, relying on specific strings appearing at the beginning of the file.

Pattern	**File Format guessed**
File starts with `%!PS`	Postscript
File starts with `%PDF` or `PDF`	PDF
File starts with <	HTML
File starts with `\\` followed by a letter	LaTEX

There are a variety of publicly available tools that can extract text from each of these formats. `pdftotext` is very good at extracting text from PDF files (whenever the file actually contains text, and not just scanned images of the pages). `lynx` does a good job at

stripping out HTML tags and returning the plain text in a page. `pstoascii` does the same thing for PostScript, the most difficult format to handle. Unfortunately, it also is one of the most popular file formats. A PostScript file is a program that, once executed by a PostScript interpreter, produces images that can be printed in pages. The difficulties in extracting useful text from PostScript are:

- Document text formatted in multiple columns: In theory, there is nothing in the PostScript format that says that all letters in a word have to appear consecutively in the file; letters can be individually positioned in the page one by one in any order. Fortunately in practice most of the time words appear consecutively, at least partially. It simply happens that the way letters to be printed are coded inside a PostScript file can be easily recognized, and so by recognizing all the letters to be printed, the text can be extracted. The complication arises when the text is formatted in more than one column, as it often happens in scientific papers. In that case, sometimes the PostScript file follows the flow of the text, and sometimes it follows the order of words on screen: the next word in the file after the last word in the first line in a column, is the first word in the next column. The only way to deal with these cases is to try to recognize the columns, and concatenate the text according to these reconstructed columns. The `prescript` extraction tool does exactly that, by computing the bounding box of letters appearing in the same baseline, finding the average width of the bounding boxes, and grouping bounding boxes with the same approximate left margin in the same column.

- Text as captions or part of figures: There is no indication of where in the document structure a section of text appears: it can be part of the main text, or a caption, or a label in an image. The simplest heuristic to recognize these cases is to skip one and two-word lines in lowercase (as to still include titles), and to ignore any short paragraph that starts with the word “Figure” followed by a number.

- Ligatures: PostScript allows for ligatures, pairs of letters that are printed as one glyph. The ligatures that are likely to appear in English texts are fi, ff, fl, ffi, ffl. In The encoding of ligatures in PostScript depends on the typography used, but scientific papers are usually printed using Times Roman, and extracted as character pairs, so with some observation it is possible to build a table of translation from character pairs that represent ligatures, and simply replace them with the correct characters.

After all the documents are harvested, we use the metadata harvested with them to compare similar titles and find cases where the same document was assigned more than one ID (which can happen in CiteSeer when references are not correct, or incorrectly cite the document title). These cases were merged whenever possible so each document has a unique ID.

As a result of the harvesting process we obtained:

1. A list of topics with documents associated
2. A group of documents in a normalized format that includes only title and text, and excludes copyright notes, acknowledgments, author's affiliation, and anything else that is not directly related to the document content
3. For each document, an associated metadata record, listing the paper title, and whenever possible the authors, publication name, and publication year
4. A sparse citation matrix, where for each document ID, there is a corresponding row in the citation matrix with a non-zero value for each document cited by the document with that ID
5. A sparse co-citation matrix, where for each pair of documents ID_1 and ID_2, the matrix is non-zero if the documents with ids ID_1 and ID_2 are co-cited by another paper

The Operating System subset of the harvested collection will be used in Chapter 5 to evaluate the implementation of Stepping Stones and Pathways, discussed in the next chapter.

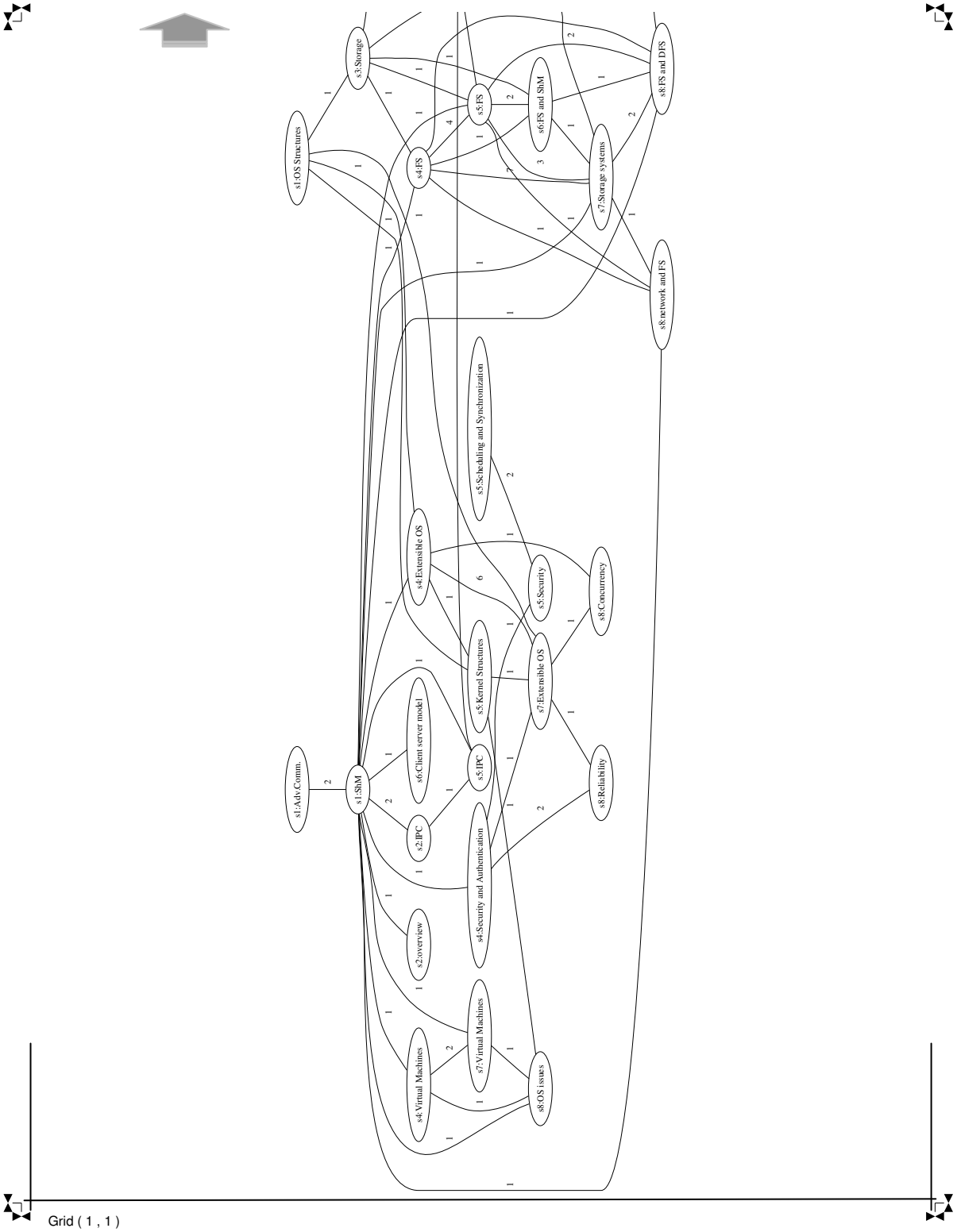


Figure 3.1.a. First half of the map of related topics in Operating Systems, according to the references in the papers listed in the syllabi.

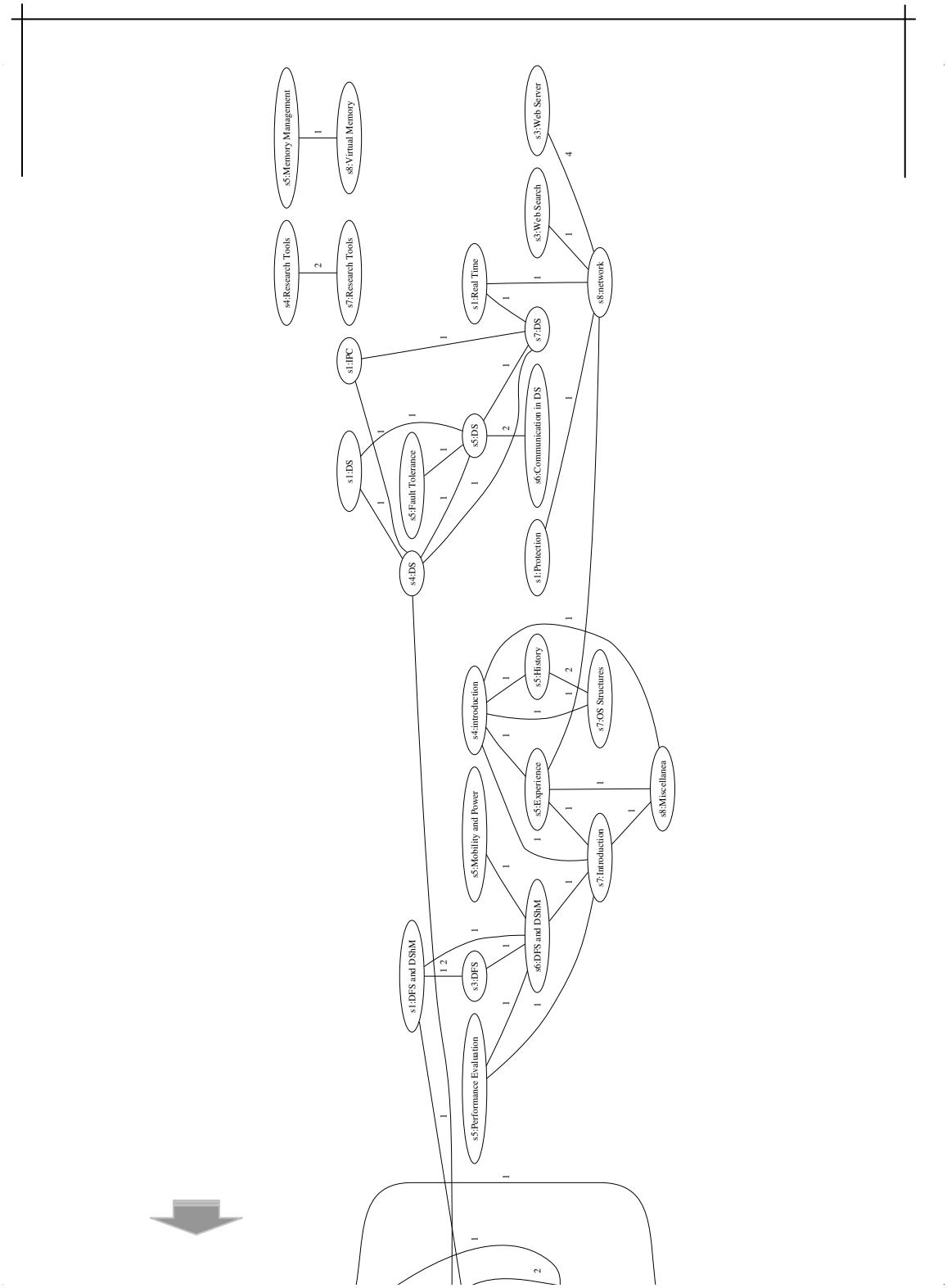


Figure 3.1.b. Second half of the map of related topics in Operating Systems, according to the references in the papers listed in the syllabi.

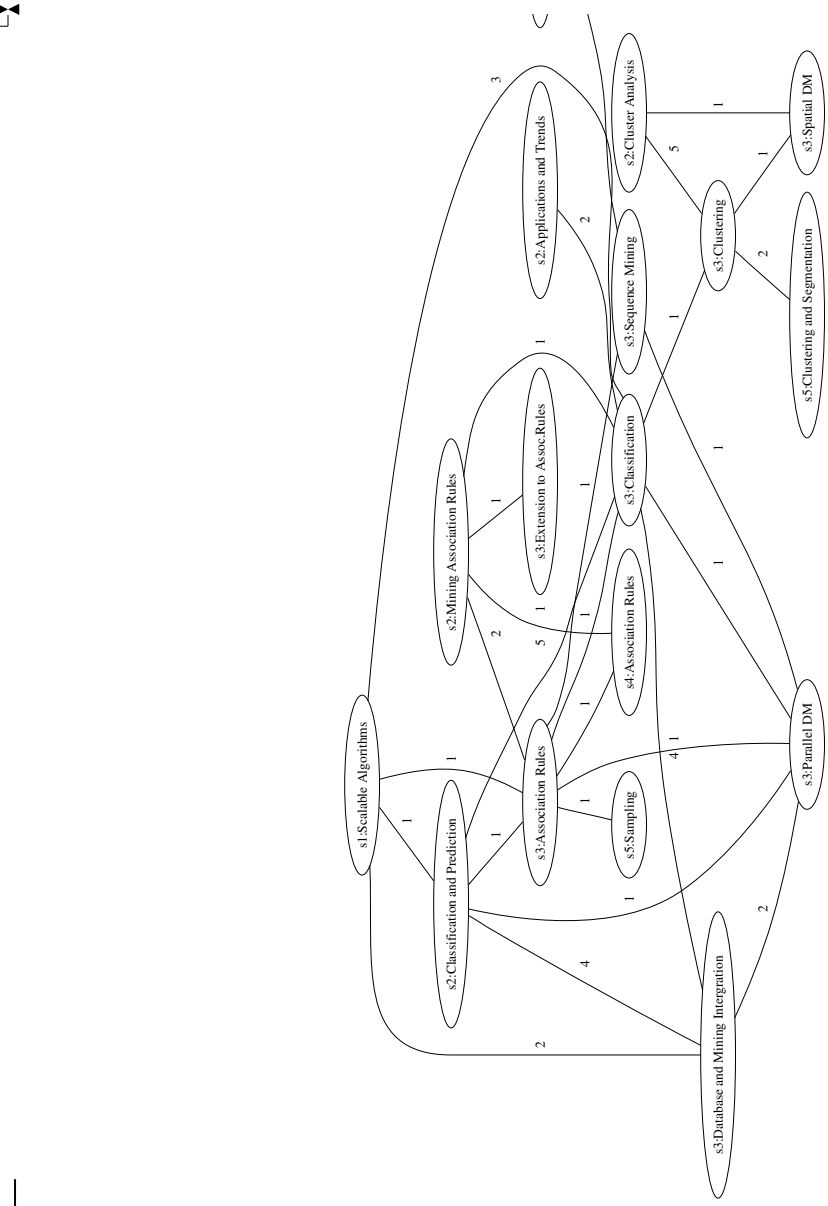


Figure 3.2.a. First half of the map of related topics in Data Mining, according to the references in the papers listed in the syllabi.

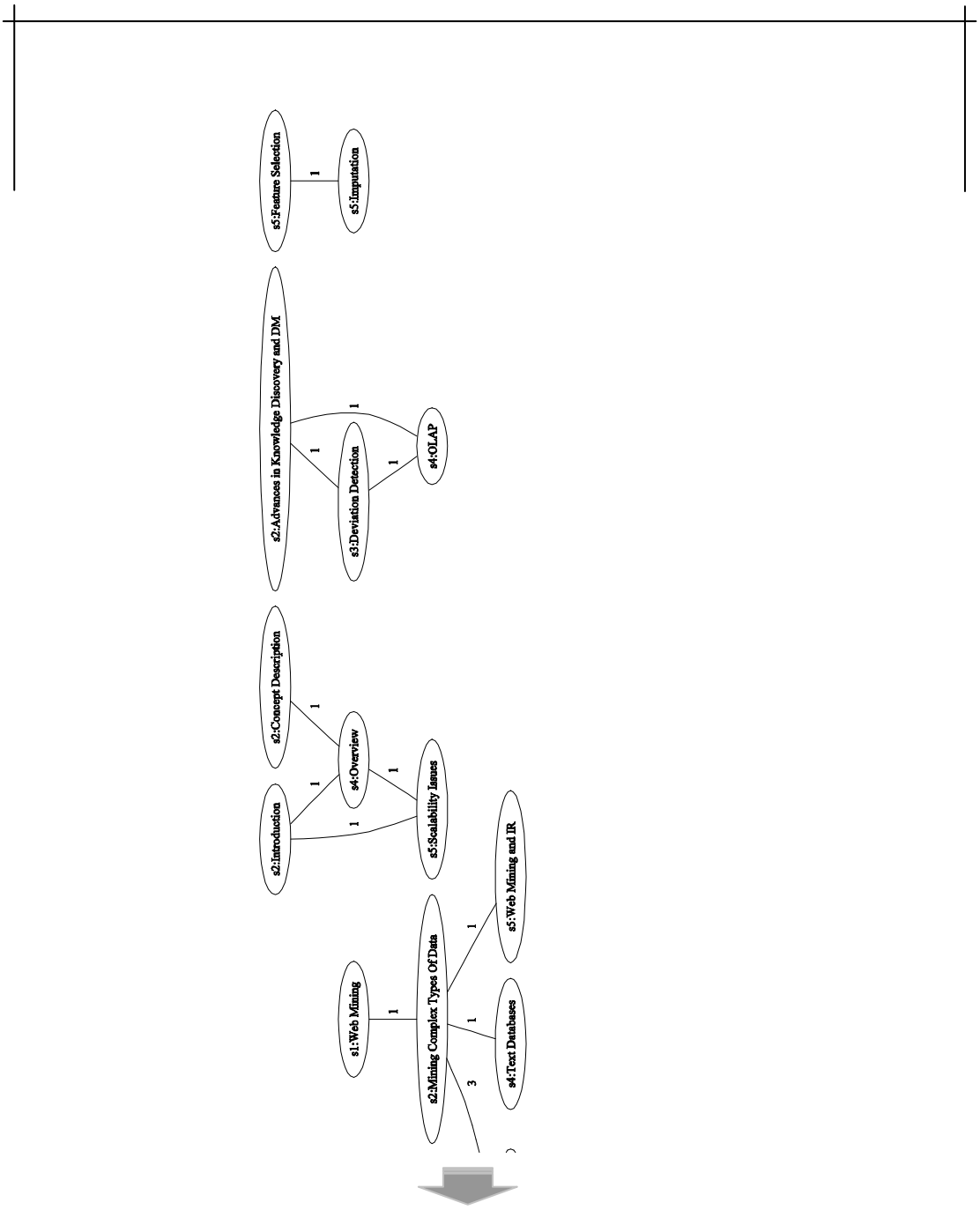


Figure 3.2.b. Second half of the map of related topics in Data Mining, according to the references in the papers listed in the syllabi.

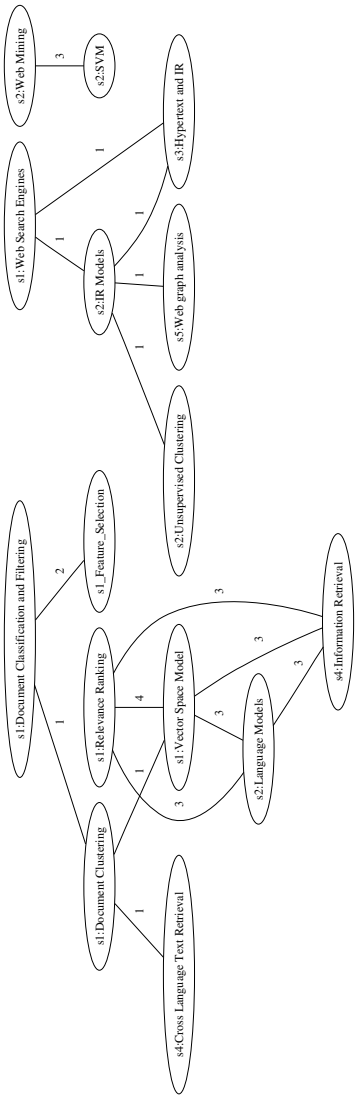


Figure 3.3.a. First half of the map of related topics in Information Retrieval, according to the references in the papers listed in the syllabi.

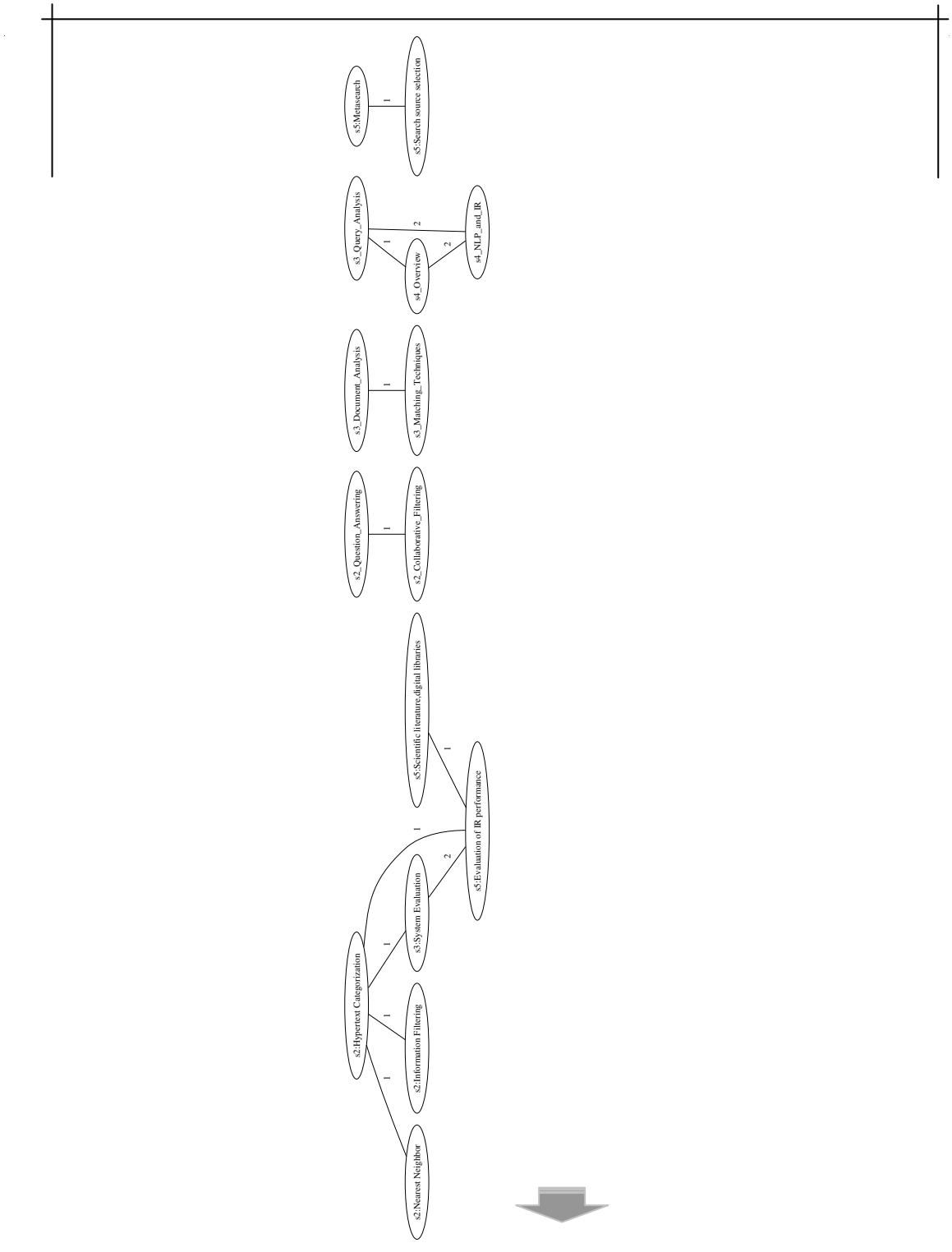


Figure 3.3.b. Second half of the map of related topics in Information Retrieval, according to the references in the papers listed in the syllabi.

4. Stepping Stones and Pathways: How it Works

Summary

In this chapter we show how Stepping Stones and Pathways and Pathways works, both in terms of user interface and algorithms. We start with an example of a session using Stepping Stones and Pathways, and then we explain the user interface in detail. We follow by detailing each step and algorithm involved in the process of indexing the documents, retrieving documents related to the user query (consisting of two sub-queries), and building pathways connecting the sub-queries.
In order to rank pathways, we devise a method to score pathways, in terms of the likelihood that the documents are a good representation of the relationships between the topics in the pathways.
We finish with the results of an evaluation of the quality the labels assigned to clusters by our implementation of Suffix-Tree Clustering, when compared to labels assigned by experts to the same clusters.

4.1 Stepping Stones and Pathways and Pathways: An Overview

Nowadays most internet users are comfortable with search engines and the concept of a ranked list of results. The mechanism is known: type a few words describing the interest, and a list of documents is returned that (hopefully) matches the user's interest. Our tool is not based on the concept of a ranked list of documents; instead it works by returning a list of connections between topics. In order to test our hypothesis we have implemented a full-working version of an information retrieval system based on Stepping Stones and Pathways and Pathways (SSPW). In this section we are going to give an overview of how SSPW works, by going through a simple example of a retrieval session.

For the purpose of this example, let us say that the user is interested in finding the relationship between "*distributed systems*" and "*file systems*". The steps she follows are:

Step 1: Ask for a network of connections:

The user would type the two sub-queries in the text boxes on the top of the screen (Figure 4.1). She would type "*distributed systems*" in one box, and "*file systems*" in the other, or type more words, whatever she thinks describes the topics. The first network of connections appears in a few seconds (center of Figure 4.1):

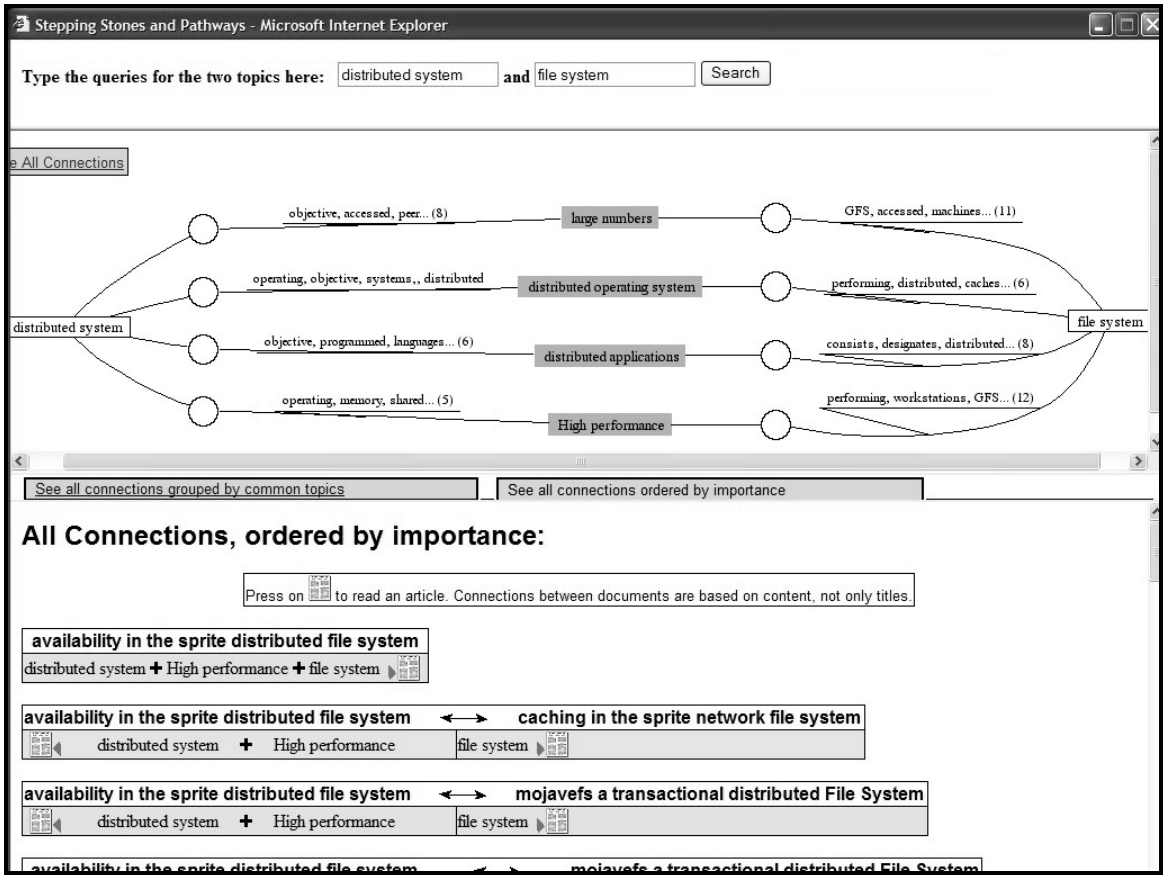


Figure 4.1 A screenshot of the Stepping Stones and Pathways and Pathways system, right after the user issued the query "distributed system" and "file system". The top half of the screen shows the graph displaying all the topics, and the way they are connected. The bottom half displays tables of documents connecting topics. In each table, the header shows the titles of the documents involved, and each row below the table header lists the topics those documents connect.

The user would see that there are 4 intermediate topics connecting "*distributed systems*" and "*file systems*": "*large numbers*", "*distributed operating systems*", "*distributed applications*", and "*high performance*". On the bottom half of the window the system presents candidates for relationships, some involving one document, some involving two, and some involving three documents. The first connection in the list connects "distributed system" to "file systems" via the intermediate topic "*high performance*", by a paper titled "*Availability in the Sprite Distributed System*" [Baker90]. The next step is to explore the connection to see if it is valid.

Step 2: Explore the first connection, through "Availability in the Sprite Distributed System":

We know from the title that Sprite is a distributed system, by the title of the paper. By reading the document, she or she finds the following paragraphs:

> *"In contrast, availability via fast recovery need not slow down a system, and our experience in Sprite shows that in some cases the same techniques that provide high performance also provide fast recovery. In our first attempt to reduce file server recovery times to less than 90 seconds, we take advantage of the distributed state already present in our file system, and a high-performance log-structured file system currently under implementation."*

Therefore the user has found a connection: she or she could say:

"Sprite (A distributed system) relates Distributed Systems and File Systems through High Performance because Sprite, a distributed system, implements a high performance file system."

Step 3: Explore "distributed systems" ↔ "distributed applications" ↔ "file system":

Now the user is going to try a different way. By default the system shows each group of documents, and then under the documents it lists all the topics that are connected through those documents. Now she is going to change the way connections are displayed. By clicking on "See all connections grouped by common topic", the display of connections will change, showing for each group of topics, all the documents connecting those topics.

The first table displays all the documents connecting "*distributed systems*" and "*file system*" through "*distributed applications*".

In particular the user is interested in three documents connecting the topics above: "*Availability in the Sprite Distributed System*" [Baker90], "*mojaveFS: a Transactional Distributed System*" [Frantz2002], and then again "*Availability in the Sprite Distributed System*" (in Figure 4.2, third row from the bottom):

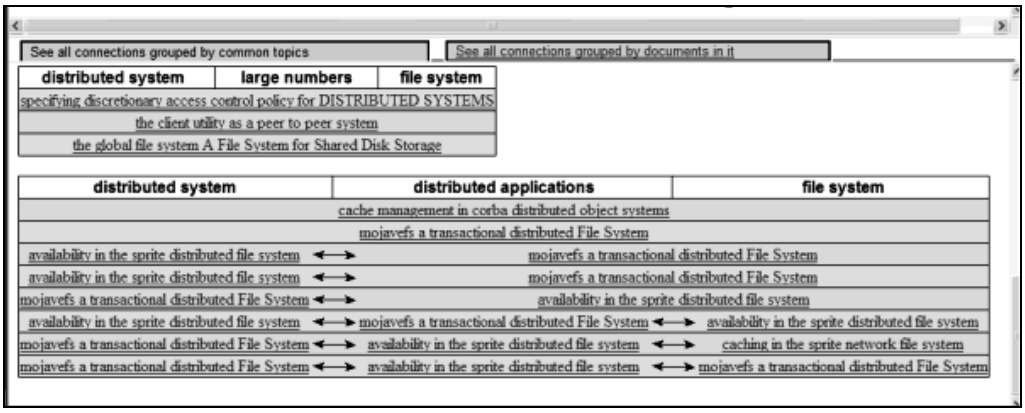


Figure 4.2 Connections for "distributed systems" and "file systems", this time grouped by common topics. In each table, the header displays the topic names, and immediately below all the documents connecting the topics in the header are displayed.

By the first connection explained in step 2, the user already knows that Sprite is a distributed file system. Next she looks at the text of "*mojaveFS: a transactional distributed system*", and finds the following paragraph:

"The design of distributed applications is challenging because many of the usual program abstractions are not valid in a distributed framework."

followed by:

[…] *"The Mojave File System (MojaveFS) is a distributed file system that uses transactions to facilitate reliable concurrent programming."*

Therefore, the user has found another valid connection:
"*distributed systems*" and "*file systems*" are related via "distributed applications" because Sprite, a distributed system, is related to mojaveFS, another distributed file system, that facilitates concurrent programming of distributed applications.

This is an example of a simple session, not involving adding, removing, splitting or joining intermediate topics. Nevertheless, this is an example of a typical session of a beginner user. Our initial design followed Schneiderman's Information Seeking Mantra [Schneiderman98]:

"Overview first,
zoom and filter,
then details-on-demand. "

During iterative design, we found that the initial overview level was too high and did not match the user expectations and mental model for an interactive search session. Our first prototype did not show the list of connections. Instead, it displayed the connection graph. The user was expected to "zoom" by clicking on nodes on the graph to see the documents

associated with a topic, and to click on a link in the graph to see the documents associated with each topic, and how they were connected. Many users in our iterative design were confused by the graph result, and were not able to decide on a search strategy through the graph, in order to find valid connections. By interviewing the users we found they applied the mental model of "type search query → wait for list → click on result", that they developed during the use of internet search engines. This observation led us to modify the user interface to show all the connections along with the graph. The list of connections takes the place of the ranked list of documents, making it simpler for the user to find connections and to understand the system. The combined display of the graph as an overview of the topic connections, plus listing all the connections, works by showing context and detail simultaneously.

4.2 The user interface in detail

The user interface is divided into 3 areas (Figure 4.3):

❶ The Query area: Here the user types two groups of words describing the topics in which the user has interest.
❷ The Connections Display area: Every time a new query is issued in the query area, the connection display area shows the initial graph connecting the topics of interest, through a number of intermediate topics.
❸ The Document and Connections Area: This area displays the list of connections between the topics, and the list of documents in a topic, depending on the user's request.

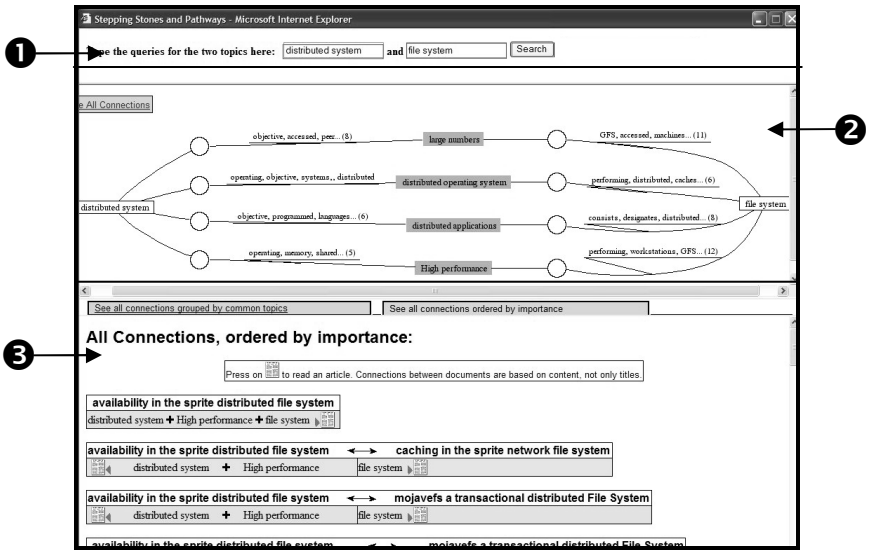


Figure 4.3 Areas of the Stepping Stones and Pathways and Pathways User Interface. The Documents and Connections area presents a list of connections.

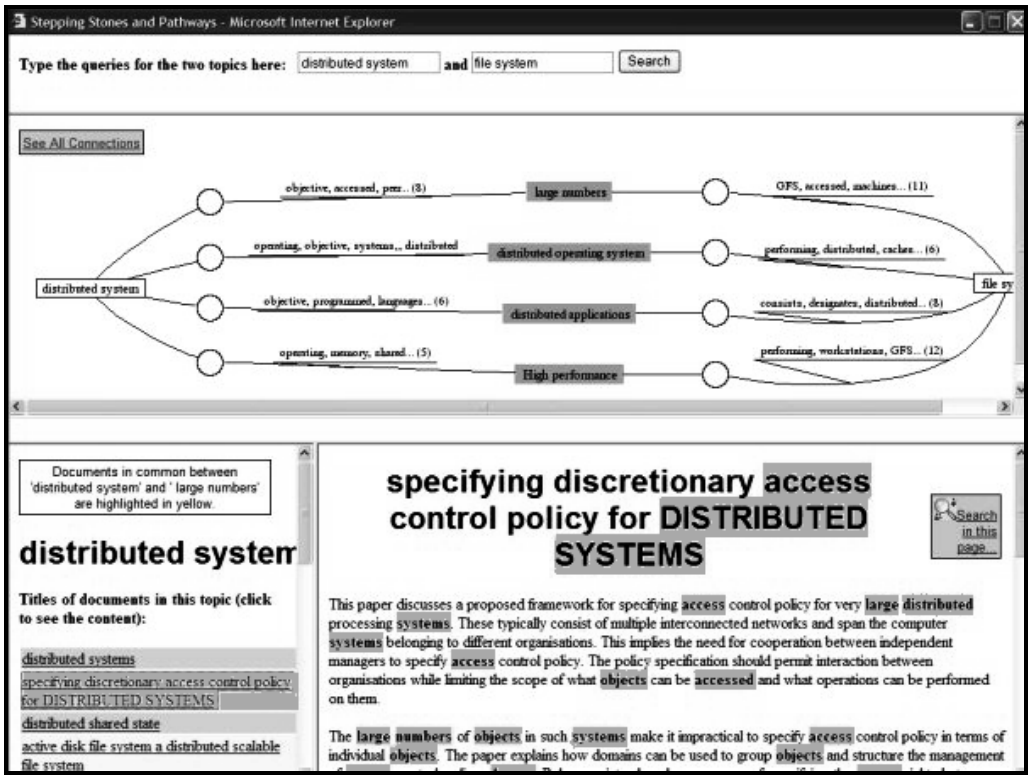


Figure 4.4 Another view of Stepping Stones and Pathways and Pathways User Interface. In this case, the Documents and Connections area is showing a list of documents to the left, and a document text on the right. Query words that appear in the document are highlighted (in gray in this picture, in yellow on the computer screen).

4.2.1 The Connection Graph

Stepping Stones and Pathways and Pathways considers retrieval as the seeking of relationships between topics. The Connection Graph display provides an overview of such connections, along with a list of keywords describing why the topics are connected. It also provides an opportunity for the user to interact in order to explore the connections between the topics.

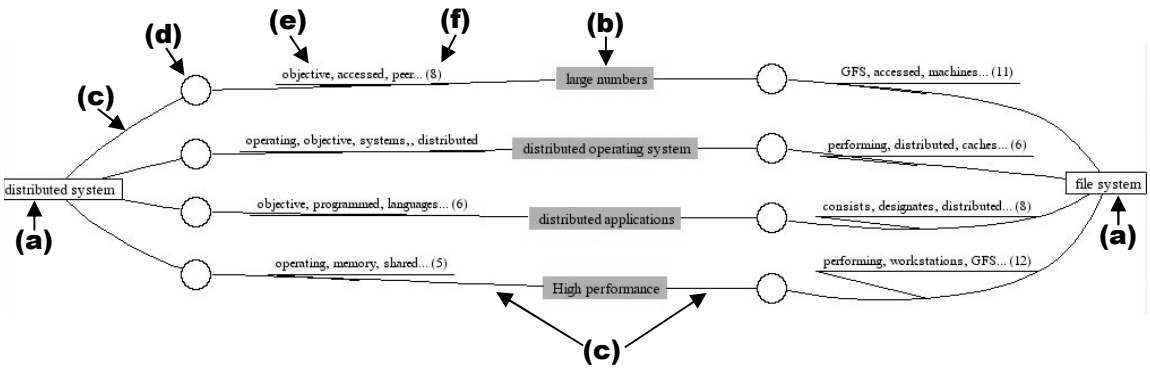


Figure 4.5. An example of a connection network and its parts.

The leftmost and rightmost nodes **(a)** are the topics that the user has given to the tool as two sub-queries, and that she wants to connect. Topics are connected by other intermediate topics **(b)** that are related to them (shown here as gray boxes), and the lines in between **(c)** indicate which topics are related. The circles halfway along the lines **(d)** are the connection "hotspots", the areas to click over to perform an action over the connection – for example, seeing the documents related to the topics in both ends of the connection. The words over the lines connecting the topics **(e)** summarize the terms in common between the topics at each end of the line. Only the first 3 most important terms in common are shown in the graph, but the total number of terms in common between two topics **(f)** is shown in parenthesis right after the first 3 terms.

In the picture above, the network connects the topics "*distributed system*" and "*file system*", shown at the left and right. In between, and from top to bottom, the graph shows the topics *"large numbers"*, *"distributed operating system"*, "*distributed applications*", and *"high performance*", each related to "*distributed system*" and *"file system"*. Every path represents an indirect connection between "*distributed system*" and *"file system"*, for example "*distributed system*" ↔ *"distributed applications"* ↔ *"file system"*. This path in particular means that "*distributed system*" and *"file system"* are related to *"distributed applications"*, the intermediate topic. The words over the line connecting "*distributed system*" and *"distributed applications"*, and between *"distributed applications"* and *"file system"* are terms in common between the topics at each end of the connection.

4.2.2 Designing the User-Tool Interaction

The user interface allows for the following actions:

1. Create endpoints

2. Add Stepping Stones and Pathways between two nodes
3. Remove a stepping stone
4. Join two or more Stepping Stones and Pathways into one stepping stone
5. Split a stepping stone into two or more nodes
6. Undo and Redo a modification of the connection graph
7. List and rank all connections through a node
8. List and rank all connections joining two nodes
9. List the documents related to a node
10. List the documents related to two nodes connected by a link, and rank the documents in each node according to the closeness to the other topic.

The actions available through the user interface were modeled after many iterations of observing the search process in the user studies conducted during development. As we mentioned before, originally we envisioned that the search process would develop around the connection graph. We will briefly describe steps of the search method we originally designed here:

1. The user issued the two sub-queries and waited for the connection graph to appear. The user then inspected the connection graph to get an overview of the topics involved and what topics are connected.

2. After finding a pathway of interest between the topics, the user then explored the topics in the pathway, either by reading the documents titles in each topic in the pathway, or by exploring the document titles in common between two topics in it (the common documents were highlighted by the system). When a document title seemed to indicate that the document supported a connection, the user clicked on the document title to get the document text. All terms in the topics to which the document are related appeared highlighted in the document text.

3. By repeated exploration of candidate topics and the documents connecting topics, the user built a mental picture of the relation between documents in related topics (in practice, most of the inspected documents were the ones suggested by the system, but some of them were found by the user). When a connection was not clear, the user asked for new topics, and those topics that were not of interest to establish a connection were removed. This mental picture was refined until the user understood the connection and found it relevant or not.

4. Steps 2 and 3 were repeated with other topics and pathways.

The original process of exploration of connections we just described was centered on the connection graph. User testing quickly showed two shortcomings of this approach:

1. Users found it difficult to work on the connection graph. The idea of a graph is very different from the ranked list of results he is used to receiving from the web search engines.

2. Exploring connections exclusively through the documents that are common to pairs of topics was too limiting. Asking the user to find related pairs of documents by himself was beyond the ability of most users, because it required the user to memorize good documents, and go back later to them when another document might complete a connection.

We observed that users would quickly switch from exploring one pathway to another, and from one document to another. By asking the user to construct connections in his head we required too much memorization, and as a result users would forget why a document was relevant to a connection.

The single most requested change was for the system to explicitly show how documents connected topics. For most users, the process of finding and validating connections between the topics was some variation of the following steps:

1. The user issued the two sub-queries, and waited for the connection graph to appear, but more important, waited for the list of connections to appear. The list of connections was used significantly more often than the connection graph, probably because it resembled the ranked list returned by conventional search engines. The connections were by default ordered by a system judgment of their importance.

2. The user developed an overview of the available connections by browsing the list of connections. Users usually performed "title scanning", reading the titles of documents that were connected to decide which ones were good candidates for exploration. As a result of this, we weight title similarity more than text similarity at the moment of calculating the similarity between two documents. For those connections that seemed to be good candidates, the user often (but not always) used the topic titles to decide if the potential connection was of interest.

3. He decided on a connection to explore. Exploring a connection implies looking at the documents supporting it, so he clicked on the document icon next to the document title in the list of connections, in order to retrieve the document content. The list of connections was replaced by the text of the document, highlighting all the terms in the topic titles that the document is relating. Two tabs, called "See connections grouped by common topics" (shows documents grouped by topics), and "See all connections grouped by documents in it" (shows topics grouped by documents) were always present to go back to the connection list.

4. The user went back to the list of connections by pressing one of the "See connections grouped by…" tabs, and explored the rest of the documents in the connection.

5. If, after reviewing some connections, none seemed relevant, or the reason why some topics were connected was not clear, the user moved to explore the document graph.

6. Exploring the graph was a combination of browsing, adding, and removing topics. When no connection seemed relevant, the most frequent action was to request to add new Stepping Stones and Pathways between two existing topics. Clicking on a stepping stone lists all the connections going through that topic. It also was possible to see a list of documents related to the topic. If adding nodes did not help, then the user undid the change in the graph to get back to how it was before adding nodes. From there, either he explored other topics or removed the Stepping Stones and Pathways that were not relevant for the purpose of connecting the two original topics.

4.3 Algorithms and Design of Stepping Stones and Pathways and Pathways

In the current version of Stepping Stones and Pathways and Pathways we consider the user query as made of two sub-queries. That is, the system does not perform query splitting, but instead it asks the user to split the query in 2 topics. While there is no reason a query cannot be split into more than two sub-queries, the reason we work with only two sub-queries is the limited number of words in a user query, as found in user studies [Moricz98] [Spink2001], which limits the number of possible split points beyond which the meaning of the sub-queries becomes too general and precision drops dramatically. While there have been many studies of collection clustering, it is clear that there is not just a single correct clustering, but it greatly depends on the user [Deogun87] knowledge and information need. Our approach, by asking the user to split and by using the sub-queries to create the initial clusters, should have a higher chance of matching the user's interest.

4.3.1 Overview of the Indexing, Clustering and Retrieval Process

The steps to create and use Stepping Stones and Pathways and Pathways are:

1. Create the Collection Index, using content and references
2. Process Query, trying to match all words in the sub-queries first, and if that fails, relaxing the sub-queries by making words optional.
3. Create Endpoints by:
    - Retrieve 2 document sets from the user sub-queries
    - Create two clusters (called centroids), one for each document set
    - Calculate a cluster centroid from the top 10 documents in the cluster
    - Label the cluster
4. Create Stepping Stones, and Pathways by:
    - Use the endpoint centroids as queries to find two document sets
    - Create an intermediate document set with the documents that appear in both retrieved sets
    - Find relevant connections between the documents at the endpoints and the documents in the intermediate set
    - Eliminate all documents in the intermediate set that are not part of a connection
    - Cluster and label all documents left in the intermediate set; these clusters become topics
    - Present all pathways (documents + topics to which they belong) to the user
5. Explore documents, topics, and pathways
6. Add, remove, join, and split topics

Here we will describe the considerations, techniques and algorithms used to perform each of these steps.

4.3.2 Index Creation

A document is considered to be divided into sections. A section is any subset of the document such as the title, the document summary, or a paragraph. For the purpose of word weighting, sections are put together in groups that we call *tag groups*. The text of all sections in the same tag group is indexed together, which means the term frequency (*tf*) of a word in a section is the *tf* of that word in all sections belonging to that group. The purpose of dividing the document in sections, and then grouping document sections into tag groups, is to allow for different ways of computing similarity for different parts of the document, while keeping the ability to retrieve different sections of the document independently from each other. For example, think of a document divided into 3 sections: title, summary, and body. We decide that document title goes to tag group 0, while summary and body go to tag group 1. Then we can use a formula to compute the

similarity between query and title, and another formula to calculate the similarity to the rest of the document. At the same time, since sections are kept separate when indexing the document, it is still possible to retrieve the title of the document.

Although the indexing engine supports any assignment of document sections to tag groups, for the purpose of Stepping Stones and Pathways and Pathways, the title is assigned to a tag group, and the rest of the document text is assigned to a second tag group.

All sections are indexed after removing stopwords and noise phrases. Stopwords are words that occur too often to be useful at identifying documents. Noise phrases are word sequences of no use for retrieval, such as copyright notes, acknowledgments, or phrases that are repeated over and over and carry no semantic value to retrieve documents. Examples of the latter type of noise phrase are: "This paper presents", "thesis supervisor", and "The rest of this section". Besides removing word sequences with no value for retrieval, removing noise phrases also improves the quality of cluster labeling. Since we use suffix-tree clustering to automatically find topics, any word sequence with high frequency would be picked as a candidate to be a cluster label. This is not a defect of the algorithm, since in other contexts, finding all the documents that mention "thesis supervisor" can help when locating all documents that are PhD dissertations or derived from them. However, for the purpose of creating Stepping Stones and Pathways, we only want topics that define sub-areas inside a technical field (equivalent to branching points in a classification system), and phrases like "thesis supervisor" or "dissertation committee" do not fulfill that purpose.

Once stopwords and noise phrases are removed, words in a document section are stemmed using a modified version of Porter's stemming algorithm [Porter80]. We stem a word only if the length of the stemmed version of the word is no less than half of the length of the unstemmed word. If the stemmed version is shorter than that, then we find out if the unstemmed word is in plural form and if so, we change it to singular. The purpose of these modifications is to avoid cases in which Porter's stemming is too aggressive, reducing to the same stem two words that are not variations of the same word.

After filtering, document words are added to the posting list, recording not only frequency of the word in the document but also the document section in which the word appears. The text of document sections is stored divided into paragraphs, with each noise phrase in a paragraph replaced by a marker to be able to identify when words really appeared together in the original document. Once all documents are indexed, the *tf·idf* weight of words is calculated, and for each word a list of closely related words is built by using the method described in [Gauch99], except that we exclude stopwords when considering a word context. This word list is used at query time for automatic query expansion.

The last step of indexing is document clustering. Clustering is performed with a suffix-tree algorithm, using document paragraphs instead of words, as in [Zamir98] and as described in the literature review in section 2.4.5. The difference with Zamir's

implementation is that we remove stopwords and we subdivide a paragraph into sub-paragraphs when a noise phrase is removed from the paragraph. We divide paragraphs when a phrase is removed to avoid creating word sequences that were not present in the document.

After adding all paragraphs to the suffix-tree, we prune all paths where at least one word has a df (document frequency)<3, or df/|D|>0.8 (the words are exceedingly rare or too frequent). These values are not collection-dependent, but the higher threshold is higher than in Zamir's paper.

Then all paths are assigned an importance value as follows:

- If the path has up to 6 words, the importance of the path is the number of words times the number of documents that contain the path as a phrase. The limit of 6 words is arbitrary, to provide a cutoff point when a phrase is too long to be useful as a label.
- If the path has more than 6 words, the importance of the path is 0.8 (an arbitrary, very low, but non-zero importance).

Since leaves of the suffix tree (end of a path = phrase) contain a list of all documents with that phrase, clusters are created by grouping all the documents in a path. In this implementation the suffix tree is formed from paragraphs, and each paragraph is a list of word ids with stopwords removed. Because of this, two documents may appear in the same cluster if they contain paragraphs that only differ in the stopwords. For example "Documents that created" and "Document creation" will be, for the purpose of clustering, the same phrase since the pairs (document,documents) and (created,creation) are each reduced to the stems "document" and "creat", and "that" is a stopword. Since we stored the full, unmodified text of each document at indexing time, along with the position of each word in each document, we can retrieve the full text of each occurrence of the phrase in all documents. From all the variations of a phrase, we choose the shortest one (counting words and stopwords) as the cluster label. The rationale for this heuristic is that the shortest phrase will contain the fewest stopwords (or no stopwords). Also, noun phrases are very good descriptors, and rarely if ever contain a stopword.

4.3.3 Calculating Document-Query Similarity

Throughout the process of building a Stepping Stones and Pathways and pathways network we need to retrieve a set of documents based on a query. This query can be one of the user sub-queries, or may be derived from a cluster centroid. To calculate the query-document similarity we combine the word similarity with the citation similarity, with the restriction that word similarity equal to zero means that the document-query similarity is zero regardless of the citation similarity. The reason for this is that, as we will discuss later, similarities based on citations are more prone to misclassification than word similarities, due to limited evidence.

A query is a set of 4-tuples (wordid, tag group, qualifier, weight). The tag_group can have a special value `ANY_GROUP` to indicate that the word is not limited to a particular group. The qualifier puts restrictions on the word: it can be `mandatory` (must appear in the tag group), `forbidden` (must not appear in the tag group), or `none` (no restrictions on the word). The weight is a value equal to 1 by default, but otherwise represents the relative importance of the word with respect to weights in other tuples.

The similarity of a query and a document is a combination of the similarity between the query and each tag group in the document. For example, if the document is divided into group 0 = title and group 1 = summary+body, then similarity of the query and the document is a combination of the similarity between the query and tag group 0 (the title) and between query and tag group 1 (summary + body).

In Stepping Stones and Pathways and Pathways we have implemented two types of similarity between a query and a tag group: cosine and Boolean similarity. Cosine similarity has been already described. The normalized Boolean similarity between a tag group in a document and a query is calculated using the formula:

$$nbool\_sim(q,d,i) = \frac{\sum_{j=1}^{|W|} boolean\_match(d,q,i,j)}{\text{number of different words in tag group } i}$$

with

$$boolean\_match(q,d,i,j) = \begin{cases} 1 \text{ if word } j \text{ appears in } q \text{ and in tag group } i \text{ of } d \\ 0 \text{ otherwise} \end{cases}$$

The normalized word similarity *nword_sim()* between a document *d* and a query *q* is calculated with the formula:

$$nword\_sim(q,d) = \frac{\sum_{i=1}^{|TG|} weight\_nz(q,i)}{\sum_{i=1}^{|TG|} weight\_tg(q,i)} \times \left(1 - \prod_{i=1}^{|TG|} (1 - nsim\_tg(q,d,i))\right)$$

where

- |TG| is the total number of tag groups.
- nsim_tg(q,d,i) is the normalized similarity between the tag group *i* in document *d*, and query *q*.
- *weight_tg(q,i)* is the weight given in the query *q* to the tag group *i*.
- *weight_nz(q,i)* is the weight given in the query *q* to the tag group *i* iff nsim_tg(*q,d,i*) is different from 0, and zero otherwise.

This score function is a combination of the CombMNZ score combination function in [Fox92] that proved to increase precision [Lee97] with a noisy-or combination of the individual scores. The reason we modified CombMNZ is that the latter assumes equal-meaning queries: All query results are judgments over the same part of the document, so a similarity of 1 in one query does not imply that the match is perfect. The weight combination used in *nword_sim()* works under a different assumption: since *nword_sim* is a combination of scores over different parts of the document, any similarity score equal to 1 over that particular part of the document indicates that the "perfect" document segment was found, so the document is relevant, regardless of the other scores.

4.3.4 Calculating Document-Document Similarity:

To calculate the similarity between two documents, we use the following formula:

$sim(d_1,d_2)=1-(1-P_w(d_1,d_2))(1-P_{cocit}(d_1,d_2)(1-P_{reft}(d_1,d_2))$

The similarity of two documents is the noisy-or combination of the word vector similarity, cocitation similarity, and document references similarity.

To calculate the word similarity between two documents (P_W), we combine a Boolean match of the document titles and the cosine similarity of the document bodies.

The citation similarity (P_{ref}) is calculated by the cosine similarity of the citation vectors of each document.

The cocitation similarity (P_{cocit}) is calculated using Garfield's link strength:

$P_{cocit}(d_1,d_2) = X/(Y-X)$, where X=number of cocitations of d_1 and d_2, Y= number of documents citing d_1 + number of documents citing d_2.

We use the same formula to calculate the similarity between a document and a query, or a document and a cluster centroid.

We treat citation and co-citation similarity differently from document content similarity. The reason is that the content similarity of two documents will be 1 if and only if they have exactly the same words regardless of the order. Thus, possibility of misclassifying two documents as being the same when they are not becomes very low after a few paragraphs.

The situation is different with citations and co-citations. There are a number of factors that make it a lot easier to extract text than citations from papers. Citations come in many different formatting styles and conventions, and variation and mistakes in the citations makes it difficult to find all cases where seemingly different citations actually refer to the same document. Also, the number of citations per paper is low, usually from 10 to 20.

Given these problems in finding and assessing the complete set of citations, if two papers have a low number of citations to other papers, then even when it is the case that the sets of citations in both papers is exactly the same, we cannot guarantee that both papers are in fact the same paper. There is a possibility of misclassification which depends on the extraction problems described above, and also on the number of citations considered. However, the function above to calculate the similarity between two documents will be equal to 1 (the documents are identical) as soon as P_{ref} is equal to 1. We need to account for the possibility of misclassification based on references.

Extraction errors ultimately have an effect on the number of citations in a paper and on the number of citations in common with other papers. Therefore, to quantify the error in misclassifying two papers as being the same when they are not (given the citations), we are going to consider the error as a function only of the number of citations in the two papers, and of the number of citations in common between two papers.

Formally, let us assign an index number to each reference in the common set of all references in all papers. Let us call ξ the probability that two documents are classified as the same when they are not, given the set of references in common.

If for two documents d_1 and d_2 we define

$$r_i = \begin{cases} 1 \text{ if the reference with index } i \text{ appears in } d_1 \text{ and } d_2 \\ 0 \text{ otherwise} \end{cases}$$

then $0 \le \xi(n) = P(d_1 \ne d_2 \mid r_1 = r_2 = ... = r_n = 1) \le 1$ means that the probability of document d_1 being different from d_2 (given that all n references between d_1 and d_2 are the same) is a function of the probability of misclassification $\xi(n)$, which in turn is a function of the number of references in d_1 and d_2.

We also want

$$0 \le P(d_1 = d_2 \mid r_1 = r_2 = ... = r_n = 1) \le P(d_1 = d_2 \mid r_1 = r_2 = ... = r_m = 1) \le 1 \text{ for } n<m$$

that is, we want the probability of two documents being the same when they have more citations in common to increase as the number of citations increases.

We define $\xi(d_1, d_2)$, the probability of misclassification, to be $\xi(d_1, d_2) \le \left(1 - \frac{|\{r_i \mid r_i = 1\}|}{n+1}\right) k$,

where

- n=*max*(total number of citations in d_1, total number of citations in d_2).
- $\{r_i|r_i=1\}$ is the set of all references in common between d_1 and d_1. Note that $0 \le |\{r_i|r_i=1\}| \le n$
- k=adjustment factor for the maximum value of $\xi(d_1, d_2)$ (we will explain how k is calculated in a few more paragraphs below)

We chose this particular formulation of ξ because it has the following needed properties:

- It is a monotonically decreasing function.
- It is always less than or equal to 1.
- It decreases rapidly for the first values of $n=1,2,..$, when all citations $r_i=1$.

For simplicity, we will from now on write the maximum possible value of $\xi(d_1, d_2)$ as $\xi(n)$.

We want to have a bound on the maximum value that the error can take. We want $\xi(n) \le \text{maximum error} = \xi_0$. The maximum value of the error (ξ_0) should occur when we are "assuming too much from too little", when we are assuming the most from the smallest possible set of references. This is the case when $n=1$ *and* $|\{r_i|r_i=1\}|=1$. That is, when the documents d_1 and d_2 have only one reference each, which is the same in both documents, then we would assume that d_1 and d_2 are the same document.

Then for $n=1$ and $|\{r_i|r_i=1\}|=1$, $\dfrac{\left|\{r_i \mid r_i = 1\}\right|}{n+1} = \dfrac{1}{2}$

so $\xi(1) = \left(1 - \dfrac{\left|\{r_i \mid r_i = 1\}\right|}{n+1}\right) k = \dfrac{1}{2} k$

Also, by the definition of maximum error above, $\xi(1) = \xi_0$.

Therefore for $n=1$, $\xi(1) = \dfrac{k}{2} = \xi_0$, so there is only one possible formulation of the value of k, the adjustment factor: $k = 2 \cdot \xi_o$.

This gives us the final formulation of

$$\xi(d_1, d_2) = \left(1 - \frac{\left|\{r_i \mid r_i = 1\}\right|}{n+1}\right)(2 \cdot \xi_0)$$

The following table illustrates the behavior of $\xi(n)$ as n grows:

n	$\xi(n)$ for $\xi_0=0.02$	$\xi(n)$ for $\xi_0=0.05$
1	0.02	0.05
2	0.01	0.033
4	0.008	0.02
5	0.006	0.016
10	0.003	0.009

Table 4.1 Examples of the decrease in the error term as the number of references increases, for values of maximum error equal to 0.02 and 0.05.

Since $0 \le P(d_1 = d_2 \mid r_1 = r_2 = ... = r_n = 1) = 1 - \xi(n) \le 1$, and $0 \le sim_{ref}(d_1, d_2) \le 1$, we can define the probability of two documents being the same, given a common set of references, to be:

$$P_{ref}(d_1 = d_2 \mid r_1 = r_2 = ... = r_{j \le n} = 1) = sim_{ref}(d_1, d_2) \cdot (1 - \xi(d_1, d_2))$$

4.3.5 Creating the Initial Stepping Stones and Pathways

The general process to create Stepping Stones and Pathways can be described in Figure 4.6. Stepping Stones and Pathways are created at run-time (query time), using the indexed document collection and clusters built at indexing time. Stepping Stones and Pathways are created from an intermediate set of documents. This set is related to document sets 1 and 2 (called doc sets 1 and 2 in Figure 4.6), that are in turn created from the sub-queries 1 and 2 given by the user. Documents in the intermediate set are then tested to see if they are related to doc sets 1 and 2, and if so they become part of a new network of Stepping Stones and Pathways and pathways.

From the two query parts given by the user, two documents clusters are built. We call them *endpoints*. For each of the two query parts, these clusters are built by trying first to find documents with all the words in the query (all query words are considered mandatory). If no document is found, then a second try searches for the documents with at least a partial match between the query and the documents. If both query results are not empty, then 10 documents are retrieved from each query, and a cluster query centroid with the top 35 words with the highest *tf·idf* score is created for each cluster. Query words that were not chosen to be part of the cluster query centroid are added to it, even if they did not fall into the top 35 highest scores. These two endpoint clusters become the sinks of the pathway graph, although at this point there are not Stepping Stones and Pathways to connect them (yet).

Query-Document matching and Document-Document matching is set to Boolean similarity for document title, and cosine similarity is used for the rest of the document. The purpose of this is to assign a higher score to matching words in the title, since for

scientific papers the titles are short, and almost always a good description of the document content.

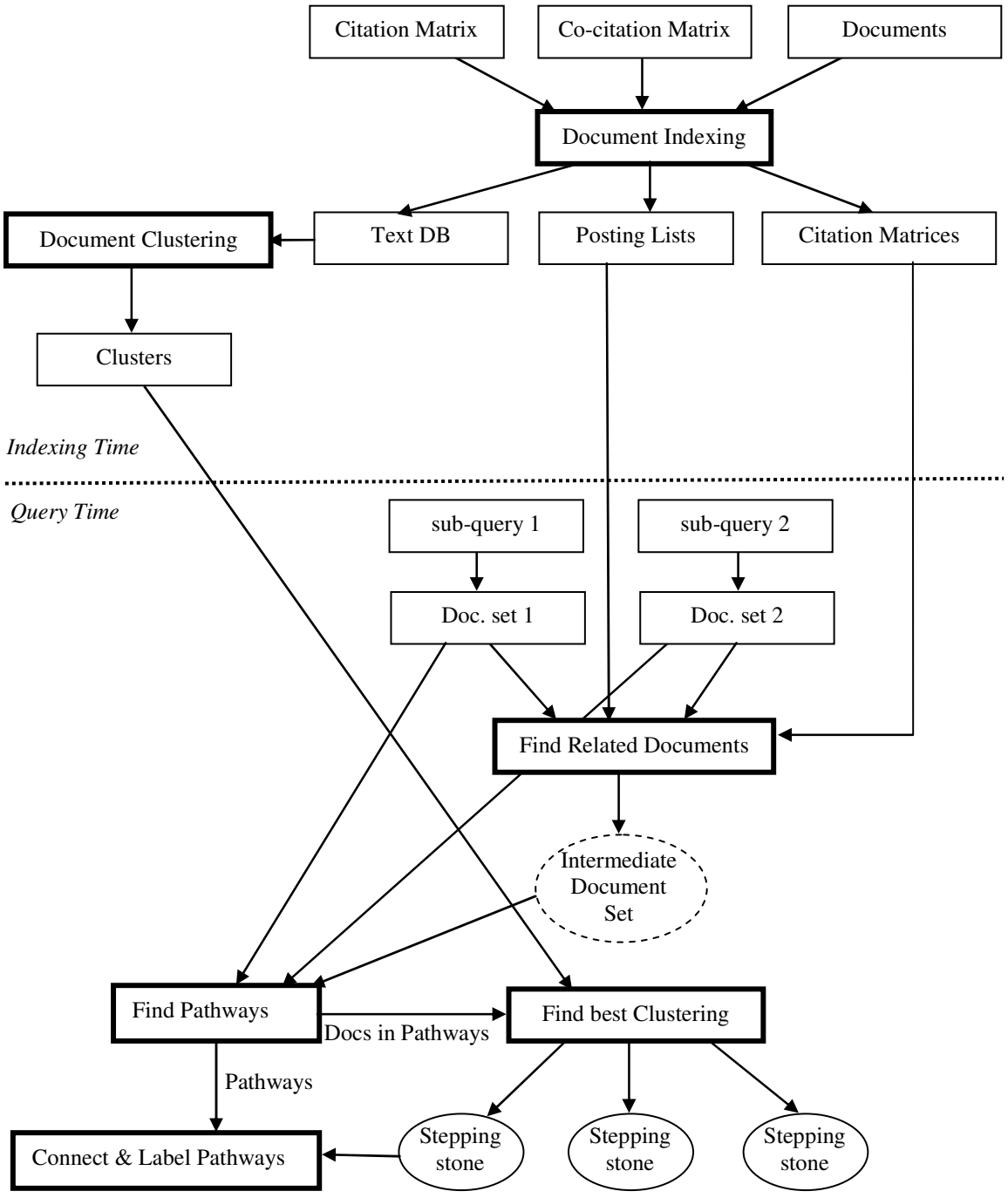


Figure 4.6 Schematic Description of the Process of Creating Stepping Stones and Pathways and Pathways.

Ideally, the set *ST* of Stepping Stones and Pathways connecting the endpoints should contain documents that are as closely related to the two endpoints, while being as different as possible from each other, so each stepping stone represents different relationships between the endpoints.

Formally, each of *i* Stepping Stones and Pathways should contain one document $s_i \in S$ that maximizes

$$\arg\max_{S \subset D} \left(\alpha \left(\sum_{s_i \in S} P(s_i \mid c_1) + \sum_{s_i \in S} P(s_i \mid c_2) \right) - \beta \sum_{i=1}^{|S|-1} \sum_{i=j+1}^{|S|} P(s_i, s_j) \right)$$

Here c_1 is a document that belongs to one of the endpoints, and document c_2 is part of the other endpoint. s_i is a possible document working as a seed to create a new stepping stone. We call *S* the set of all document seeds for c_1 and c_2. α and β are parameters indicating the relative importance of closeness of s_i to the endpoints relative to the separation between the elements in *S*. Note that the similarity between documents is calculated in this model using not only content but also references that are included as part of the model.

Each seed is a document containing a concept that is related to the endpoints. From each seed we would create a group of documents, clustering by maximizing similarity to the seed document. These intermediate clusters connect to the endpoints, forming a "hammock" structure with c_1 and c_2 at the ends of the hammock, and the intermediate clusters and their connections to c_1 and c_2 as the body of the hammock.

Assuming we find a set of document seeds, if the similarity between an intermediate cluster and an endpoint is below a threshold, then a new hammock would be created from extra stepping stones and extra pathways between the endpoint and the cluster in the same way a hammock was created between the endpoints. The process would be repeated until all cluster relationships were significant, or the network reaches a maximum length. All clusters with low similarity to the adjacent clusters would be dropped and the result would be a network of document groups. Each document group then would be labeled.

Unfortunately, finding the set of document seeds *S* is an optimization problem for which we do not know a solution suitable for the response times required in an interactive system. Therefore we choose to approximate *S*. We do so by first limiting the space of solutions (documents), and then performing an exhaustive search over the space of possible solutions (connections). While clearly an optimal construction of the set of connections will not be possible, Scatter/Gather [Cutting92] proved that when users are allowed to be an active part of the refinement process, a 'good is good enough' approach is possible.

The first step to approximate S is to reduce the solution space. The two cluster centroids of the endpoints are used as queries to retrieve two sets E_1 and E_2 of *D* documents, one for each query. E_1 and E_2 must have a high similarity to the corresponding centroid,

greater than or equal to a threshold. We use the intersection of E_1 and E_2 as an approximation to *S*, that we call *S'*.

4.3.6 Looking for Connections

After gathering the set *S'*, the next step is to search the solution space for connections between the topics. In our case, the solution space to explore consists of documents related to the topics to connect, and intermediate documents in S' that may or may not connect those topics. That is, we look for connections between documents in the endpoints (let's call them E_1 and E_2) and documents in *S'*. To do that we retrieve two new sets *L* and *R* (left and right) of documents, one for each endpoint. This time the sets *L* and *R* are not generated by using the corresponding query centroids of E_1 and E_2.. The original user sub-queries used to generate E_1 and E_2 are used instead, since we found that using the original user queries to retrieve *L* and *R* gives more precise results.

We consider a candidate to be a pathway between *L* and *R*, any quartet of documents (d_L,d_{S1}, d_{S2}, d_R) with d_L in *L*, two documents d_{S1} and d_{S2} document in *S'* (note that d_{S1} can be the same as d_{S2}), and a document d_R in *R,* that follow these restrictions:

- $sim(d_L,d_{S1}) >= Tc$
- $sim(d_{S2},d_R) >= Tc$
- $sim(d_{S1},d_{S2}) >= Ts$

Experimentally, for the current collection we chose the values $T_c=T_s=0.6$, which favors rejecting false positives over accepting false negatives. In information retrieval terms, we are favoring precision over recall of the document pairs that will be part of connections. The reason we chose these values is because during our tests in iterative design, one of the first reactions from users was of being bewildered with too many possible connections. Furthermore, we observed that if the first connections they inspected were not strong, they tended to start skipping over the rest of the connections, looking at them less carefully. This suggested to us that for novice users, we needed to show strong connections, even if that meant discarding some possibly valid connections.

The restrictions above allow for a large variety of candidate connections between topics. There are two possible types of connections:

1. A **direct connection** between topics is one where the two topics are connected by the same document.
2. An **indirect connection** is one where the two topics are connected by two different documents.

We consider a valid pathway between *L* and *R* to be a quartet of documents (d_L, d_{S1}, d_{S2}, d_R) that follows the restrictions above, and that fits one of the 4 cases in Table 4.2:

Case	Explanation
$d_L=d_{S1}=d_{S2}=d_R$	The same document connects L to S and S to R
$d_L=d_{S1}$ and $d_{S1}\neq d_{S2}$ and $d_{S2}=d_R$	d_{S1} connects L to S, and d_{S2} connects S to R
$d_L=d_{S1}$ and $d_{S1}\neq d_{S2}$ and $d_{S2}\neq d_R$	(d_L,d_{S1}) connects L to S, and (d_{S2},d_R) connects S to R.
$d_L\neq d_{S1}$ and $d_{S1}\neq d_{S2}$ and $d_{S2}=d_R$	(d_L,d_{S1}) connects L to S, and (d_{S2},d_R) connects S to R.

Table 4.2. Possible types of document relationships connecting topics, and their rationale.

Note that following the above cases, for 3 topics there are 2 connections, for which at most 1 connection is indirect and at least 1 is direct, with a total of at most 3 different documents. We ask that at least one connection be direct (at least one pair of documents is always required to be the same document) to avoid inferring too much from transitive (indirect) connections.

After all possible connections are found, the documents in *S'* that are part of a pathway between *L* and *R* are grouped into topics, and the topics are labeled. Before describing how the documents in *S'* are grouped and labeled, let us show the final form of a pathway between *L* and *R*. Let us call *I* (for intermediate) the label of the topic to which a document that connects *L* and *R* belongs. Since for each pathway there are at most 3 different documents, we have the following possible pathways illustrated in Table 4.3:

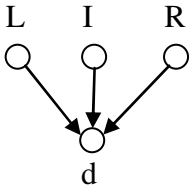
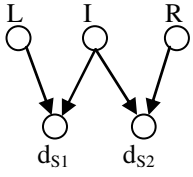
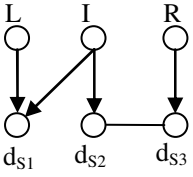
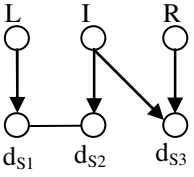
Case	Case 1:	Case 2:	Case 3:	Case 4:
	$d_L=d_{S1}=d_{S2}=d_R$	$d_L=d_{S1}$ and $d_{S1}\neq d_{S2}$ and $d_{S2}=d_R$	$d_L=d_{S1}$ and $d_{S1}\neq d_{S2}$ and $d_{S2}\neq d_R$	$d_L\neq d_{S1}$ and $d_{S1}\neq d_{S2}$ and $d_{S2}=d_R$
Associated Graphical Model				

Table 4.3. Types of connections between topics that are supported by documents, and their associated graphical models. Note that in the graphical models, conditional probabilities (arrows) in the graphical model correspond to direct connections, and joint probabilities (lines without arrows) correspond to indirect connections.

As mentioned in the previous paragraph, the final step to building Stepping Stones and Pathways is to subdivide *S'* (the set of documents that are candidates to belong to Stepping Stones and Pathways) into clusters that will become topics. To do this, we pre-cluster all documents at indexing time using a suffix-tree clustering algorithm, and we match *S'* to this existing collection of clusters by the number of documents in each cluster

that are also in *S'*. The advantages of using Suffix-Tree Clustering (STC) for our purposes are:

1. **STC creates overlapping clusters.** A document may belong to more than one cluster, thus making it possible to have small groups of documents classified in multiple ways, as part of different clusters.

2. **It labels the cluster with phrases.** A cluster is created when two or more documents share a common sentence consisting of 2 to 6 words.

3. **The labels can be matched to one or more specific sentences in the document**, thus making it easy to identify why a document was placed in a certain cluster.

These query-independent clusters are created only once, before starting to create Stepping Stones and Pathways. The clusters are sorted in decreasing order of importance $C_1 \ldots C_n$, using a score computed in a manner similar to the one in [Zamir98], involving the length of the cluster label and the number of documents in the cluster. A higher score indicates that the cluster is a better description of the documents in it.

Then we match the set of documents in S that belong to at least one connection, to the existing clusters $C=[c_1 \ldots c_m]$, in the following way:

```
Inputs:
     S, a set of documents
     C, a set of clusters
Output: S_clusters, a set of clusters
S_remaining=S
S_clusters={}
while(S_remaining is not empty) do
     Find c_i in C such that c_i∩S_clusters=∅ and c_i∩S_remaining≠∅
     that maximizes the score:
     score=|c_i∩S|+2*|(c_i∩S_remaining)|
     S_clusters= S_clusters ∪ {c_i}
     S_remaining= S_remaining − c_i
end while
```

In each pass, this algorithm picks a new cluster c_i from C that maximizes $|c_i \cap S|+2*|(c_i \cap S_{remaining})|$, but with the condition that c_i must contain at least one document from the set of documents that has not already been assigned to a cluster in $S_{clusters}$. In this way, each pass of the algorithm reduces the number of documents to be clustered. At the same time, by clustering S in this way, we create overlapping clusters. The advantage of overlapping clusters is that they improve the quality of the cluster labels. Allowing c_i to partially overlap with the set of documents already clustered enables the algorithm to avoid arbitrary restrictions on the set of documents to cluster in each step. Since the number of documents in each cluster is higher than it would be for non-overlapping clusters, and since the size of S is usually small, by avoiding artificial divisions of the

cluster set, we can have a higher confidence that the label matches the document content. Each of the clusters in $S_{clusters}$ becomes a stepping stone, participating in one or more connections.

Once a connection between two adjacent topics is found, the connection needs to be labeled in a way that expresses what is in common between the topics, and how they are related. These connection labels allow the user to quickly scan the connections for interesting terms. A connection is labeled with the 10 top words that appear in the query centroid of both topics, ranked by increasing absolute difference in weight of the word in both query centroids (that is, the word with the smallest absolute difference first). The intuitive explanation for this rule is that from the set of the top 35 words in each topic (the query centroids of the topics), it ranks the words by similarity in importance in both documents. In other words, this rule chooses words with high weight overall, but with similar weight in both topics. The effect of this rule is to avoid words that favor one document over another, instead concentrating on what the documents have in common, in terms of words-as-descriptors.

4.3.7 Ranking the Connections

The final step, before presenting the new Stepping Stones and Pathways to the user, is to rank the connections. To rank the connections, we give a probabilistic interpretation to the graphical model associated with each rule in Table 4.3.

If each cluster is a topic, and T is the cluster associated with the topic, then we interpret:

- $sim(d_1,d_2) = sim(d_2,d_1)$ = similarity of the content of documents d_1 and d_2
- $sim(T,d) = sim(d,T) = sim(centroid(T),d)$ = similarity based on the probability of document d being relevant to cluster T.

We define the following random variables:

d_j: the document j takes the values $d_j = \begin{cases} 1 = \text{document j is relevant} \\ 0 = \text{document j is not relevant} \end{cases}$

T_i: the topic i takes the values $T_i = \begin{cases} 1 = \text{topic i is relevant to the connection} \\ 0 = \text{topic i is not relevant to the connection} \end{cases}$

We also define the following probabilities:

- $P(d=1)=1/|D|$, the a-priori probability of a document being relevant, where D={set of all documents}

- $P(d_1 = 1 \mid d_2 = 1) = \begin{cases} sim(d_1, d_2) \text{ if } sim(d_1, d_2) > threshold_{sim} \\ 0 \text{ otherwise} \end{cases}$, the probability of the content of document d_1 being relevant, given that document d_2 is declared relevant.

- $P(d_1 = 1, d_2 = 1) = P(d_1 = 1 \mid d_2 = 1) P(d_2 = 1)$, the probability that d_{s2} and d_R are mutually relevant, understanding mutual relevance as *"both documents are related by content and structure"*. The four combinations of possible values of d_1 and d_2, along with their interpretation, are listed below:

d_1	d_2	interpretation
1	1	d_1 and d_2 are mutually relevant
1	0	d_1 is relevant to d_2 , but d_2 is not relevant to d_1
0	1	d_1 is not relevant to d_2 , but d_2 is relevant to d_1
0	0	d_1 is not relevant to d_2 , and d_2 is not relevant to d_1

Since the definition of $P(d_1=1|d_2=1)$ given above is symmetric, and $P(d=1)$ is equal for all d, then $P(d_1=1,d_2=0) = P(d_1=0,d_2=1) = (1-P(d_1=1|d_2=1))P(d=1)$.

- $P(d = 1 \mid T = 1) = \begin{cases} sim(d, T) \text{ if } sim(d,T) > threshold_{sim} \\ 0 \text{ otherwise} \end{cases}$ is the probability of document d being relevant given that cluster T is relevant to the connection.

- $P(T_1 = 1, T_2 = 1, T_3 = 1) = P(T_1 = 1) P(T_2 = 1) P(T_3 = 1)$ (that is, topics are independent)

- $P(d = 1 \mid T_1 = T_2 = ... = T_n = 1) = \prod_{i=1..n} P(d = 1 \mid T_i = 1)$ is the probability of document d being relevant, given that topics $T_1 \ldots T_n$ are relevant. We are assuming here that the probability of a document being related to one topic is independent of the probability of being related to the other topics. We use $P(d=1|T_1=T_2 \ldots =T_n=1)$ as the probability of d being related to $T_1, T_2, \ldots, T_n$.

From now on, when the variables d_j take the value 1, we are going to write d instead of $d=1$. We also do not write the value of T when $T=1$.

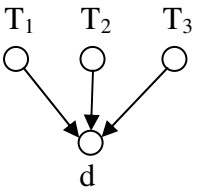
Then,

we have that $P(d_1|d_2)$=sim(d_1,d_2), and since sim(d_1,d_2)=sim(d_2,d_1), it follows that $P(d_1|d_2) = P(d_2|d_1)$.

Note that $P(d_1 \mid d_2) = \frac{P(d_1, d_2)}{P(d_2)}$ means also that $P(d_1 \mid d_2) = P(d_2 \mid d_1) = \frac{P(d_1, d_2)}{P(d_1)}$; therefore it must be true that $P(d_1)=P(d_2)$, which coincides with the definition of $P(d)$ we gave above.

From these probabilities, and re-interpreting the graphical models as Bayesian networks by removing undirected links and replacing them with an equivalent directed subgraph, we can assign a probability to each pathway, as we show below.

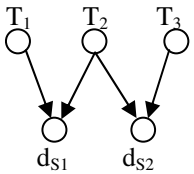
1) For a pathway with $d_L=d_{S1}=d_{S2}=d_R$:



According to this network, $P(d, T_1, T_2, T_3) = P(d \mid T_1, T_2, T_3)P(T_1)P(T_2)P(T_3)$, then $P(d \mid T_1, T_2, T_3) = \frac{P(d \mid T_1, T_2, T_3)P(T_1)P(T_2)P(T_3)}{P(T_1, T_2, T_3)}$, which by the definition of $P(T_1,T_2,T_3)$ and $P(d|T_1,T_2,T_3)$ given above, becomes

$$P(d \mid T_1, T_2, T_3) = \frac{P(d \mid T_1, T_2, T_3)P(T_1)P(T_2)P(T_3)}{P(T_1)P(T_2)P(T_3)} = P(d_1 \mid T_1)P(d_2 \mid T_2)P(d_3 \mid T_3)$$

2) For a pathway with $d_L=d_{S1}$ and $d_{S1} \neq d_{S2}$ and $d_{S2}=d_R$:



$$P(d_{s1}, d_{S2}, T_1, T_2, T_3) = P(d_{S1} \mid T_1, T_2)P(d_{S2} \mid T_2, T_3)P(T_1)P(T_2)P(T_3)$$

which by the definition of $P(d|T_1,\ldots,T_n)$, becomes

$$P(d_{s1}, d_{S2}, T_1, T_2, T_3) = P(d_{S1} \mid T_1)P(d_{S1} \mid T_2)P(d_{S2} \mid T_2)P(d_{S2} \mid T_3)P(T_1)P(T_2)P(T_3).$$

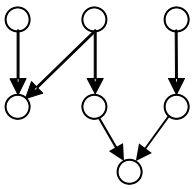
Then

$$P(d_{s1},d_{S2}\,|\,T_1,T_2,T_3)=\frac{P(d_{S1}\,|\,T_1)P(d_{S1}\,|\,T_2)P(d_{S1}\,|\,T_2)P(d_{S2}\,|\,T_3)P(T_1)P(T_2)P(T_3)}{P(T_1,T_2,T_3)}.$$

Then

$$P(d_{s1},d_{S2}\,|\,T_1,T_2,T_3)=P(d_{S1}\,|\,T_1)P(d_{S1}\,|\,T_2)P(d_{S1}\,|\,T_2)P(d_{S2}\,|\,T_3)$$

3) For a pathway with $d_L=d_{S1}$ and $d_{S1}\neq d_{S2}$ and $d_{S2}\neq d_{R:}$



$$P(d_{S2},d_{S3},d_R,T_1,T_2,T_3)=P(d_{S1}\,|\,T_1,T_2)P(c\,|\,d_{S2},d_R)P(d_{S2}\,|\,T_2)P(d_R\,|\,T_3)P(T_1)P(T_2)P(T_3)$$

Here we have eliminated the undirected link in the graphical model, and introduced an extra random variable *c*, that stands for "common content". The probability $P(c=1|d_{s2}=1,d_R=1)$ is the probability that the content of documents d_{s2} and d_R be the same, being that d_{s2} and d_R are relevant. From this definition we choose to estimate $P(c=1|d_{s2}=1,d_R=1)$ by $P(d_{s2}=1,d_R=1)$.

$$P(d_{S2},d_{S3},d_R,T_1,T_2,T_3)=P(d_{S1}\,|\,T_1,T_2)P(d_{S2}\,|\,T_2)P(d_{S1},d_{S2})P(d_R\,|\,T_3)P(T_1)P(T_2)P(T_3)=$$

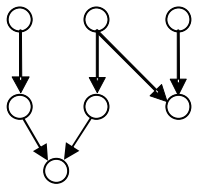
$$=P(d_{s1}\,|\,T_1)P(d_{s1}\,|\,T_2)P(d_{s2}\,|\,T_2)P(d_{s1},d_{s2})P(d_R\,|\,T_3)P(T_1)P(T_2)P(T_3)=$$

$$=P(d_{s1}\,|\,T_1)P(d_{s1}\,|\,T_2)P(d_{s2}\,|\,T_2)P(d_{s1}\,|\,d_{s2})P(d_{s2})P(d_R\,|\,T_3)P(T_1)P(T_2)P(T_3)$$

Then

$$P(d_{s1},d_{s2},d_R\,|\,T_1,T_2,T_3)=P(d_{s1}\,|\,T_1)P(d_{s1}\,|\,T_2)P(d_{s2}\,|\,T_2)P(d_{s1}\,|\,d_{s2})P(d_{s2})P(d_R\,|\,T_3)$$

4) And finally, when $d_L\neq d_{S1}$ and $d_{S1}\neq d_{S2}$ and $d_{S2}=d_{R:}$



$$P(d_L, d_{s2}, d_{s3}, T_1, T_2, T_3) = P(c \mid d_L, d_{s2}) P(d_L \mid T_1) P(d_{s2} \mid T_2) P(d_{s3} \mid T_2, T_3) P(T_1) P(T_2) P(T_3) =$$

$$= P(d_L, d_{S2}) P(d_L \mid T_1) P(d_{s2} \mid T_2) P(d_{s3} \mid T_2) P(d_{s3} \mid T_3) P(T_1) P(T_2) P(T_3)$$

then

$$P(d_L, d_{s2}, d_{s3} \mid T_1, T_2, T_3) = P(d_L \mid T_1) P(d_L \mid d_{s2}) P(d_{s2}) P(d_{s2} \mid T_2) P(d_{s3} \mid T_2) P(d_{s3} \mid T_3)$$

The definitions of probabilities above are not arbitrary, but are chosen to be consistent among themselves, and to make sense with regard to the graphical models above.

There are other definitions of probabilities that at first impression seem more intuitive. In particular, it may seem that defining

$P(d,T)$=sim(d,T) instead of $P(d|T)$=$sim(d,T)$, and

$P(d_1,d_2)$=sim(d_1,d_2) instead of $P(d_1|d_2)$=sim(d_1,d_2)

would make more sense, and lead to an easier interpretation of the probabilities of the graphical models. The problem is that under this interpretation, we get
$P(d_{s1}, d_{S2}, T_1, T_2, T_3) = P(d_{S1} \mid T_1, T_2) P(d_{S2} \mid T_2, T_3) P(T_1) P(T_2) P(T_3) =$

$$= P(d_{S1} \mid T_1) P(d_{S1} \mid T_2) P(d_{S2} \mid T_2) P(d_{S1} \mid T_3) P(T_1) P(T_2) P(T_3) =$$

$$= \frac{P(d_{S1}, T_1) P(d_{S1}, T_2) P(d_{S2}, T_2) P(d_{S1}, T_3) P(T_1) P(T_2) P(T_3)}{P(T_1) P(T_2) P(T_2) P(T_3)}$$

then $P(d_{s1}, d_{s2} \mid T_1, T_2, T_3) = \frac{P(d_{S1}, T_1) P(d_{S1}, T_2) P(d_{S2}, T_2) P(d_{S1}, T_3)}{P(T_1) P(T_2) P(T_2) P(T_3)}$

which leaves us with the problem of estimating $P(T_1)$, $P(T_2)$, and $P(T_3)$, a non-trivial problem. What makes estimating $P(T_i)$ complicated is that we cannot assign an arbitrary

a-priori estimation to it, since $P(T_i)$ appears in the denominator of the expression above. Given that $P(d_{s1}, d_{s2} \mid T_1, T_2, T_3) \leq 1$ because it is a probability, the only way of estimating the value of $P(T)$ while keeping $P(d_{s1}, d_{s2} \mid T_1, T_2, T_3) \leq 1$ is to tie the estimation of $P(T_i)$ to the estimation of the value of $P(d,T)$. Thus, we are not free to assign any convenient a-priori value to $P(T)$.

Therefore starting from $P(d_1|d_2)$ and $P(d|T)$ leads to a calculation of the probabilities of the pathways that is much harder to achieve than when starting from $P(d_1,d_2)$ and $P(d,T)$.

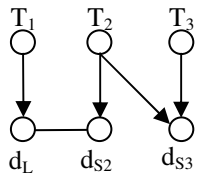
4.3.8 Creating new Stepping Stones and Pathways between other Stepping Stones and Pathways

The process to create new Stepping Stones and Pathways between two existing Stepping Stones and Pathways is almost the same as the one used to create the initial Stepping Stones and Pathways between the endpoints. The difference is that at the moment of creating the set to find the connections (Section 4.3.6), if the stepping stone was created directly from a user query, then we use the original user query to retrieve a set of documents related to the topic. Otherwise, when the topic was created by the system and not from a user query, we use the top 35 words in the cluster centroid as the query, as described in Section 4.3.5.

4.3.9 Extending the Ranking to more than 3 Topics

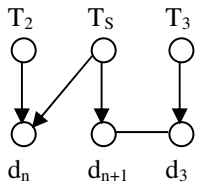
The above formulas can assign a value to any connection made of up to 3 topics and up to 3 documents. However, as a user adds new Stepping Stones and Pathways between existing ones, the number of topics goes beyond 3, and we need a way to calculate $P(d_1 \ldots d_j|T_1 \ldots T_k)$ when k>3.

A new stepping stone S between two existing Stepping Stones and Pathways L and R has one or more documents supporting the relationship between L and S and S and R. By the way a stepping stone is built, it is always the case that there is one such document. If there are no restrictions on how the documents supporting the connections (L,S) and (S,R) are picked, then it is possible the number of documents supporting a connection may grow too fast for the connection to make sense from the user point of view. Let us illustrate this with an example. Suppose that we have a pathway like this:



and we want to add a new stepping stone T_S between T_1 and T_2. If we consider T_1 and T_2 as two endpoints, then any of the 4 cases of a valid connection between two nodes

enumerated above is a valid way to connect T_1 and T_2. Suppose that we find a connection under case 4. Then T_1 and T_2 become connected through T_S as follows:



Since we are working under no restrictions regarding how to choose documents to support the connections between T_2 and T_S, and between T_S and T_3, we have chosen two new documents d_S and d_{S+1}. The final transformation of the pathway after adding T_S is depicted in Figure 4.7.

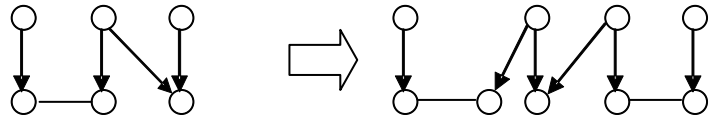


Figure 4.7 An example of pathway growth when there are no restrictions in adding documents to support the pathway.

Note that after adding T_S, three new documents were added (d_s, d_{s+1}, and d_{S+2}), and one document removed (d_3). This is perfectly valid, because it happened that d_s and d_{s+1} connect T_2, T_S and T_3, and d_{S+1} was found closer to d_{S+2} than to d_3, so d_3 was replaced by d_{s+2}. Thus, under no restrictions in the way we choose documents to connect topics, in the worst case each stepping stone would add two new documents to the chain of documents explaining the relationship between topics. As the number of documents grows, the number of documents supporting the pathway grows quickly, and the relation between topics becomes more difficult to understand for the user.

For this reason, we add the following restrictions:

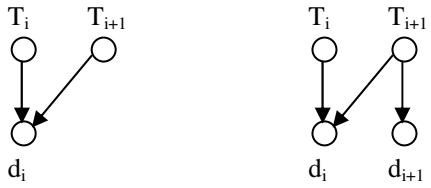
1. A new stepping stone is allowed to add up to one document to the pathway.
2. If topics T_1 and T_2 are connected through document d_1, or through documents d_1 and d_2, a new stepping stone between T_1 and T_2 only modifies the edges (T_1,d_1), (T_1,d_2), (T_1,d_2), (T_2,d_2), and (d_1,d_2) of the Bayesian network between the topics and the documents supporting the connections between topics.

Restriction 1 puts a limit on how fast the number of documents supporting a pathway can grow, while allowing adding a document to support the new intermediate topic. Restriction 2 means in practice that when adding a stepping stone between T_1 and T_2, the only edges in the Bayesian network for the pathway affected are the edges connecting T_1 and T_2, and not any other edge.

4.3.10 Updating the Probabilities

Suppose there is a pathway $T_1 \rightarrow T_2 \rightarrow T_3 \rightarrow \ldots \rightarrow T_n$, supported by documents $d_1, d_2, \ldots, d_m$, with m≤n. The score of the new pathway formed by adding a new stepping stone T_S between T_i and T_{i+1} depends on the way T_i and T_{i+1} were connected before adding T_S, and follows one the following 3 cases:

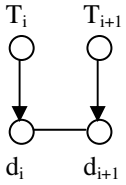
Case 1): If before adding an intermediate stepping stone T_S, T_i and T_{i+1} are connected like one of these two graphs:



then after adding the new intermediate stepping stone T_S, optionally supported by document d_S:

$$P(d_1,..,d_m,d_S \mid T_1,...,T_n,T_S) =$$
$$= \frac{P(d_1,..,d_m \mid T_1,...,T_n)}{P(d_i \mid T_{i+1})} \times \frac{P(d_i,d_S,d_{i+1} \mid T_1,T_S,T_{i+1})}{P(d_i \mid T_i)P(d_{i+1} \mid T_{i+1})P(T_i)P(T_{i+1})}$$

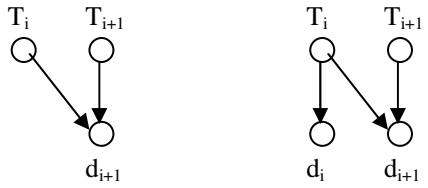
Case 2): If before adding an intermediate stepping stone T_S, T_i and T_{i+1} are connected as in this graph:



then after adding the new intermediate stepping stone T_S, optionally supported by document d_S:

$$P(d_1,..,d_m,d_S \mid T_1,...,T_n,T_S) =$$
$$= \frac{P(d_1,..,d_m \mid T_1,...,T_n)}{P(d_i,d_{i+1})} \times \frac{P(d_i,d_S,d_{i+1} \mid T_1,T_S,T_{i+1})}{P(d_i \mid T_i)P(d_{i+1} \mid T_{i+1})P(T_i)P(T_{i+1})}$$

Case 3) If before adding an intermediate stepping stone T_S, T_i and T_{i+1} are connected like one of these two graphs:



then after adding the new intermediate stepping stone T_S, optionally supported by document d_S:

$$P(d_1,..,d_m,d_S \mid T_1,...,T_n,T_S) =$$
$$= \frac{P(d_1,..,d_m \mid T_1,...,T_n)}{P(d_{i+1} \mid T_i)} \times \frac{P(d_i,d_S,d_{i+1} \mid T_1,T_S,T_{i+1})}{P(d_i \mid T_i)P(d_{i+1} \mid T_{i+1})P(T_i)P(T_{i+1})}$$

The intuitive interpretation of these rules is that the score of the new pathway is the score of the old pathway, divided by the probability of everything that connected T_i and T_{i+1} in the old pathway, multiplied by what connects T_i to T_S, and T_S to T_{i+1}, in the new pathway.

With regard to the denominator $P(d_i \mid T_i)P(d_{i+1} \mid T_{i+1})P(T_i)P(T_{i+1})$ in the score updating formulas, it may seem that having $P(T_i)$ and $P(T_{i+1})$ in the denominator contradicts our previous effort to avoid estimating $P(T_i)$, because of the difficulty of having a precise expression for it. In practice, this is not a problem. The above expressions are generic forms for updating values. In each case, once a stepping stone is added, we know exactly what edges were added between T_1, T_S, T_{i+1}, d_i, d_S, and d_{i+1}. We do not really need to compute $\frac{P(d_i,d_S,d_{i+1} \mid T_1,T_S,T_{i+1})}{P(d_i \mid T_i)P(d_{i+1} \mid T_{i+1})P(T_i)P(T_{i+1})}$, since what dividing by $P(d_i \mid T_i)P(d_{i+1} \mid T_{i+1})P(T_i)P(T_{i+1})$ does is remove the probabilities of all the edges that were already computed, so they do not get computed twice. In effect, $\frac{P(d_i,d_S,d_{i+1} \mid T_1,T_S,T_{i+1})}{P(d_i \mid T_i)P(d_{i+1} \mid T_{i+1})P(T_i)P(T_{i+1})}$ means "only compute all the probabilities of the edges that were not in the pathway before".

As a result of this, to update the score of the pathway, we just replace $\frac{P(d_i,d_S,d_{i+1} \mid T_1,T_S,T_{i+1})}{P(d_i \mid T_i)P(d_{i+1} \mid T_{i+1})P(T_i)P(T_{i+1})}$ by the multiplication of the probabilities of all new edges added to the Bayesian network after adding T_s and d_S. This multiplication does not involve any $P(T)$.

The advantages of the locality of the modification of the Bayesian network when adding a new stepping stone are:

- From the user's point of view, it is easy to understand what has changed after adding a new stepping stone, since at most one document is added to support the pathway, and no documents already in the pathway are removed. As shown in the example in Figure 4.7, without restricting how adding a new stepping stone can modify the Bayesian network, documents may "disappear" after adding the stepping stone.
- From the system's point of view, adding a new stepping stone does not involve recomputing all the probabilities to compute the new score, but only updating the previous score with some more values.

The main disadvantage of the locality of changes when adding a stepping stone is that there is no check for global coherence in the pathway. Since each update only affects the adjacent nodes, it is possible that while all local connections are valid, the meaning of all connections taken together will be difficult to understand.

4.3.11 Splitting and Joining Topics

From observing users with no previous experience using Stepping Stones and Pathways and Pathways, we noted that the most common graph-modifying operation they used was to request new Stepping Stones and Pathways, followed by undo, and then go back to the state before the graph was expanded with new Stepping Stones and Pathways. Add+undo was a common method to explore connections that were not clear, as users became more familiar with the system. The next operation in frequency was to remove topics (nodes) that were not relevant to the purpose of connecting the two topics in the query; however, Add+remove was much less used than Add+undo. The first versions of Stepping Stones and Pathways and Pathways included two more operations to modify the graph: join two or more topics into a common topic, and split a node into two or more subtopics. The rationale for these operations was to allow deeming a topic as "too generic" or "too specific". Two or more nodes that were too specific, yet similar according to the user, would be joined into a common topic that is more general in content than the individual topics. A node that was determined as "too generic" by the user would be split into two or more subtopics, each topic containing documents that are more closely related than the average similarity of the original topic. By splitting a node into more specific nodes, it was hoped that the subtopics could be labeled more accurately, thus allowing to remove the parts of the original topic that were not relevant, while keeping the parts that strengthen the relationship between topics.

The implementation of split and join depends on two different clustering algorithms. Splitting is based on K-Means, while joining is based on the result of suffix-tree pre-clustering used to label Stepping Stones and Pathways.

The rationale behind choosing K-Means and STC for splitting and joining topics is that they are different clustering criteria, yet they can complement each other: When drilling down, K-Means is guaranteed to divide a cluster into *K* sub-clusters as long as there are enough documents in the original cluster, regardless of the similarity between the documents. When generalizing by joining formerly separated clusters, STC will redistribute the documents in clusters in a fashion similar to *Gather* followed by *Scatter* in the Scatter/Gather retrieval method, except that STC will group the documents according to the easiness by which they can have a label assigned. STC is likely to put together any group of documents that are related, given that it can create a group from any set of documents that share a phrase.

From the point of view of the user, it is important that splitting and joining be consistent operations over the graph: joining a group of nodes that were created by splitting one node should recreate that original node with the same labeling that it had before being split. We can prove that this is exactly what happens with the combination of splitting with K-Means and joining with STC:

Let D be a set of documents, $L=(\{L_1,L_2,....L_k,\},<)$ a set of labeled overlapping clusters of D with a partial order defined on them, and $score(S,L_i)$ a function that assigns a value to a label L of subset S of D.

Assumption 1: If S is a set of documents $\{d_1,d_2,\ldots,d_n\}$, $C=\{c_1,c_2,\ldots c_m\}$ with $m \leq n$ is a set cover of S, and L_i is the label of c_i, then there is no label L_x such that $score(L_i,c_i)<score(Lx,ci)$. In other words, we assume that each set of document c_i has the best possible label according to the score function.

Assumption 2: If $score(L_i,c_i)=score(L_x,c_i)$, and L_i is the label of c_i, then $L_i>L_x$.

Then: if L_i is the label of document set c_i, and $split(c_i)=\{c_{i1},c_{i2}\ldots c_{is}\} \Rightarrow join(c_{i1},c_{i2}\ldots c_{is}) = \bigcup_{j=1}^{s} c_{ij} = c_i$ with label L_i. In other words, joining a set, that comes from a split document set, results in the original set.

Proof:

Suppose that $\{c_{i1},c_{i2}\ldots c_{is}\}$ comes from splitting the document set c_i, and c_j is the result of joining $\{c_{i1},c_{i2}\ldots c_{is}\}$, and that there is a label L_x such that $score(L_i,c_j) \leq score(L_x,c_j)$. We consider two cases:

Case 1) $score(L_i,c_j)<score(L_x,c_i)$:

Joining c_j creates a set $c_j=c_i$. Therefore case 1 is impossible since by assumption 1, $score(L_i,c_i) \geq score(Lx,ci)$.

Case 2) $score(L_i,c_j)=score(L_x,c_j)$:

Then by assumption 2, it is true that $L_i > L_x$, and L_i is the label of c_j. Since $c_j = c_i$ and L_i is the label of c_i, by assumption 1 is must be true that $score(L_i,c_i) \geq score(L_x,c_i)$, therefore there is no L_x such that $score(L_i,c_i) = score(L_x,c_i)$.

Since case 1 and 2 are proven false, there is no label L_x such that $score(L_i,c_i) \leq score(L_x,c_i)$, and it is always true that if L_i is the label of c_i, and $split(c_i) = \{c_{i1},c_{i2}\ldots c_{is}\}$, then $join(c_{i1},c_{i2}\ldots c_{is}) = c_i$ with label L_i.

The STC-based clustering algorithm we use satisfies assumptions 1 and 2.

For assumption 1: STC-based clustering works with a pre-existing overlapping clustering of the document set into clusters $L = \{L_1,L_2,\ldots.L_{k,}\}$, and each cluster has an importance score value associated with it. Therefore we can use this importance value as <, the partial order function. Given a set C, the STC-based clustering algorithm produces a set cover of C, since in each step it chooses a subset c_i only from the elements of C. This subset c_i is chosen by giving a score to the fitness of c_i and each cluster in the set of clusters L. In each step, c_i gets assigned the label of a cluster L_j from L. Since the algorithm chooses the Lj that is the cluster with the best score to c_i from all clusters in L, all parts of assumption 1 are granted.

For assumption 2: The STC-clustering algorithm always picks for c_i the cluster in L with the highest score. Since by definition of the algorithm the elements in L are ordered by decreasing importance value, then for two clusters L_j and L_x in L such that $score(L_i,c_i) = score(L_x,c_i)$, the algorithm chooses the first one found in decreasing importance order, therefore $L_j > L_x$ by order of importance.

4.3.12 Handling duplicate Topics after Splitting or Adding Topics

Every time topics are added to the connection network, either by adding new Stepping Stones and Pathways or by splitting one, there is the possibility that more than one cluster will have the same label. This may even happen when creating the initial set of Stepping Stones and Pathways between two endpoints, when the endpoints are closely related. The choices when handling duplicate topics are: to leave them as they are, or to merge duplicates. In our current implementation we chose not to merge duplicates because:

a) Merging duplicates changes the shape of the connection graph, potentially confusing test subjects.

b) Our testing showed that people had few problems when the connection showed more than one stepping stone with the same name. In a few cases it was possible that one stepping stone between the endpoints would be labeled matching the label of one of the endpoints. They indicated it would be better to avoid duplicated topics, but many of them interpreted the content of duplicate topics as just a more specialized version of the endpoint.

Therefore it seems that having topics with the same label does not constitute a problem for inexperienced users (these users do not create complicated networks of connections).

For the general case, however, in which more complicated networks may include duplicated topics, different Stepping Stones and Pathways with the same label may be merged when:

1. The content (document set) of the Stepping Stones and Pathways is similar, and
2. For endpoints A and B and duplicated Stepping Stones and Pathways D_1 and D_2, D_2 does not belong to the paths A—…—D_1 or D_1—…—B , and D_1 does not belong to the paths A—…—D_2 or D_2—…—B.

Restriction number 2 is needed to keep the same interpretation of the connection graph before and after merging nodes. When merging two topics that are in different paths, the effect is for both paths to share a segment after merging, as shown in Figure 4.8. This merge did not change the interpretation of the graph: all paths between endpoints A and B are still present, and no path between A and B has been modified or added by the merge.

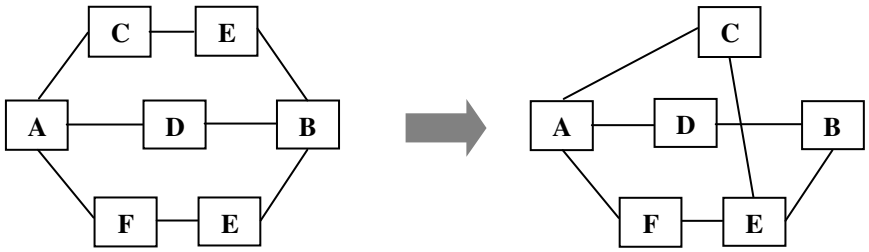


Figure 4.8 An interpretation-preserving topic merge in a network connecting two topics A and B. Occurrences of the same topic E appeared in different paths between the endpoints A and B.

However, merging two nodes that do not comply with restriction 2 leads to loops in the graph that change the interpretation of the graph. Take for example the graph illustrated at the left of Figure 4.9. After merging duplicated topics B, the graph would not only end up with a modified path from endpoint A to endpoint D, but with an additional endpoint that was not there before merging, therefore altering the interpretation of the connections between the endpoints.

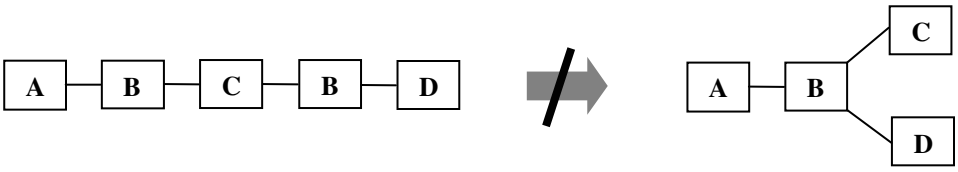


Figure 4.9 An example of merging topics over the same path that changes the interpretation of the network connecting the two topics A and D. Note the appearance of a new endpoint C, and the removal of C as an intermediate topic between A and D.

4.4 Implementation of Stepping Stones and Pathways and Pathways

The implementation of SSPW is derived from Essex, a search engine [Krowne2001] developed for the CITIDEL project [Fox2003]. The architecture of SSPW is divided into two parts: the engine and the front-end. Engine and front-end communicate by a protocol, that while specific to this application, is independent of details like byte order and size of the machine data types. The following diagram depicts the complete architecture of SSPW:

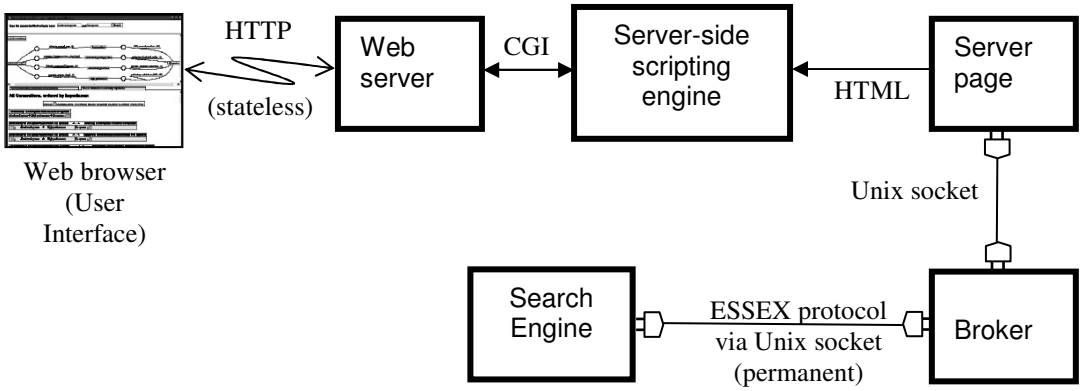


Figure 4.10 Architecture diagram showing the major implementation modules of Stepping Stones and Pathways and Pathways.

The architecture of Stepping Stones and Pathways and Pathways is divided into the following parts:

- The User Interface, running inside a web browser
- A Web Server, which handles requests from the user interface and passes them to the server pages

- A Server-Side Scripting Engine, that takes a server page, executes the code embedded in it, and sends the result to the web server

- The Server Pages, that interpret commands embedded in the URL, contact the Search Engine through the Broker, and produce the user interface with what the search engine returns as the result of the commands

- The Broker, that maps session ids in the URLs to a permanent user session in the search engine, passes the commands to the Search Engine in the proper format, and reformats the results returned by the Search Engine

- The Search Engine, which implements Stepping Stones and Pathways and Pathways as a retrieval method, and keeps the state of a user session

The user interface in SSPW is implemented with a combination of HTML and Javascript. We chose to implement a web-based user interface to offer SSPW to the widest possible audience (as long as the user has a modern web browser, he can use SSPW), to take advantage of the simplicity of HTML in prototyping, and to allow for remote system testing without the user having to install software or having to run SSPW under a particular operating system.

At the server side, each web page is a combination of HTML and scripts to be executed by Spyce [Barr2002], a server-side scripting engine that executes Python scripts embedded within the web page. Each user action or change in the interface state that requires talking to the back-end is mapped to web page requests. The script inside each web page connects to the broker's Unix socket, sends a request though it for an operation to be performed by Essex (like create a stepping stone, for example), reads the answer, closes the connection to the broker, and dynamically formats the answer to display it as an HTML page, where each action to be performed in the user interaction is coded in the page as a URL. Scripts inside web pages also map the graph description returned by Essex to a visual representation of the connection graph that the user can click on. The graphic representation of the connection graph is created using Graphviz [Gansner99], a graph layout and rendering program, and then the graph map is post-processed to map available user actions on graph nodes to URLs.

As is common to all user interfaces that talk to a remote server via HTTP, the implementation challenge is to maintain state information when working with a stateless protocol like HTTP. In all systems where the back-end is operated via a web interface, it is the back-end that keeps the system state, since the interface, in the form of HTML pages, must be regenerated after each HTTP request: a web browser has no concept of partial update within a page.

We attacked the problem of how to keep state information in a partially stateless architecture by dividing the problem into two parts: user interface state (the actions allowed based on the current objects on screen and the interaction history) and system

state (what has the user requested from the system, and what it implies in terms of creating a connection graph, modifying the graph, and retrieving documents, but not on how to present any of them).

From a conceptual point of view, the behavior of the user interface is divided into 4 states, each restricting a subset of the available actions. Figure 4.11 and Table 3.1 list the user interface modes, along with their description, transitions, and limitations on the set of user actions.

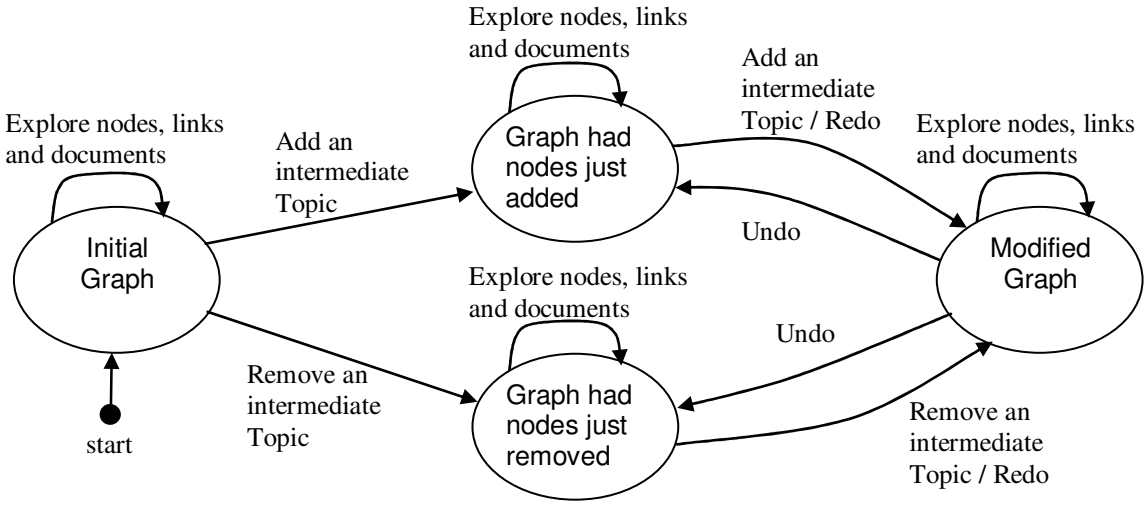


Figure 4.11. Modes and transitions of the Stepping Stones and Pathways and Pathways user interface.

Interface Mode	Description	Restricted Operations
Initial graph	The graph has not been modified yet in this user session.	• Undo and Redo graph modification (nothing to undo or redo) • Removing Stepping Stones and Pathways (would disconnect the graph)
Graph had nodes just added	The user has requested new Stepping Stones and Pathways to be created between two nodes in the graph.	• Redo (no graph changes to redo)
Graph had nodes just deleted	The user has requested a stepping stone to be removed from the graph.	• Redo (no graph changes to redo)
Modified graph	The graph has had nodes added or removed during the current user session.	• Redo (previous action was not undo)

Table 4.4 List of user interface modes, and actions restricted in each mode.

Keeping the user interface state is implemented by having separate web pages for different interface modes. Each change in mode is generated by a user action, but not all user actions lead to changes in mode. Each change in mode leads to the request for a new web page, that may look exactly like the previous page, but the code embedded in the server page is different, dealing with restrictions of user actions.

Keeping the system state is solved by identifying each user session with a unique session id. This session id is codified as part of the URL, which makes it easy to retrieve all of the user-system interactions in a session for further analysis. Since each action will be logged by the web server as part of the standard server log, each log entry corresponding with the session will have the session id in it. The Essex search engine has the concept of session in it, but Essex only performs retrieval, not presentation of results, so the conversion from session id in the web browser to an Essex session in the server must be performed somewhere else.

The handling of session ids is performed in two steps: session id creation, and session id mapping.

Session ids are created when the user starts SSPW. The script inside the web page detects if a session id is present in the URL of the page, and if not, creates a unique string that will be the session id. The initial page creates a session id and then redirects the web browser to the SSPW user interface URL, appending the session id to it. Separating session id creation from interface presentation prevents a page reload to create a new session id.

Mapping session ids to Essex sessions is the job of the Broker, a Unix daemon that waits for requests in a particular Unix socket. Each request must include a session id string. If the session id is not present in the current set of session ids handled by the broker, then the broker opens a new connection to Essex via a Unix Socket, and keeps the mapping from this session id string to the corresponding Unix socket. The broker is also responsible for timing out user sessions. In practice the concept of exiting or logging off from a web application does not exist; users do not log off or exit a web application, they simply close the web browser, or shift to a new URL. Since in our case almost every user interaction with the interface generates a new communication to the web server, the only sure way of recognizing an exit of the application is to assume that if the user has not done anything for a while, it means the user is not using the application any longer and so it is safe for the broker to assume that the session can be terminated, removing the session id from the list of active sessions, and closing the corresponding communication with Essex.

4.5 Evaluation of Cluster Labeling

The success in finding good pathways between topics lies in finding documents that connect topics appropriately, and in being able to properly label Stepping Stones and Pathways that serve as intermediate topics. A label for a stepping stone that the user cannot identify as a topic, or that seems to indicate an unrelated topic, may lead the user to skip an otherwise valid pathway.

It is for this reason that we performed an evaluation experiment over the quality of cluster labels, even if the clustering algorithm is not among the contributions of this dissertation. As mentioned before, we cluster using Suffix-Tree Clustering based on its ability to describe the clusters. The experiment we performed was to compare the algorithm-generated cluster labels against labels assigned by an expert.

4.5.1 Evaluation Method and Participants

We clustered the 2173 papers in the Operating System collection of papers using STC. For each paper, we indexed the title plus approximately the first 4 kilobytes of plain text of the document's body. We excluded acknowledgments, copyright notes, and other parts of text that are not the title or the document body. We did not make use of references for this experiment.

From the result of clustering the Operating Systems collection, we randomly chose 100 clusters, each of which should have a size in the range 5 to 30. Then we recruited 10 computer science graduate students, who have taken at least one operating system course. Each of these 10 graduate students was asked to label 10 clusters. In order to label the clusters, they used a system that presented the list of documents in a cluster on the top half of the screen, and the content of the currently selected document on the bottom half of the screen. The result then was 100 expert-assigned labels, one for each cluster.

Sixteen undergraduate students were then recruited as the label-quality judges. All these students have taken the operating systems undergraduate course. These students were then presented with 10 clusters in sequence. For each cluster, they used the same interface the experts who assigned the labels used to explore the clusters, except that on the top of the screen there were two labels. One of them was assigned by the algorithm, the other by the expert. The order of the label on the screen was random. The judges did not know how these labels were generated, and they were asked to grade each label in a 1-5 scale for appropriateness of the label as description of the content of the cluster. They also were asked to grade their confidence in the scores they assigned to the labels in a 1-5 scale (1=Not confident at all, 2=Vaguely confident, 3=Confident, 4=Very confident, 5=Totally sure of the scores).

It is totally unrealistic to expect the labels assigned by the algorithm to be as good as labels assigned by the experts. Experts employ knowledge and understanding of the document content; the algorithm only works in a mechanical fashion. Our target was for the algorithm labels to be within 1 point of the expert-assigned label. For example, if the average score for expert labels is 4, we wanted the average score of labels by the algorithm to be 3 or more.

4.5.2 Results

The results of the labeling experiment were 160 pairs of score labels (some cluster labels were scored twice). The following table and histogram summarizes the results for the 160 pairs of score labels.

Cluster Labeling: Summary of Expert-Labeled vs. Algorithm-Labeled

		Score of label by Expert				
		1	**2**	**3**	**4**	**5**
Score of label by Algorithm	**1**	2	9	9	4	1
	2	1	6	11	31	3
	3	0	8	11	22	1
	4	0	5	11	10	5
	5	0	2	5	3	0

Table 4.5. Score Count for Algorithm vs. Expert labeling. The number inside each cell is the count for each pair of values of scores given by subjects to the human-generated labels (hand labeled) and the labels generated by the algorithm. For example there were 8 pairs of labels for which the human-assigned label got a score of 2, and the algorithm-generated label a score of 3.

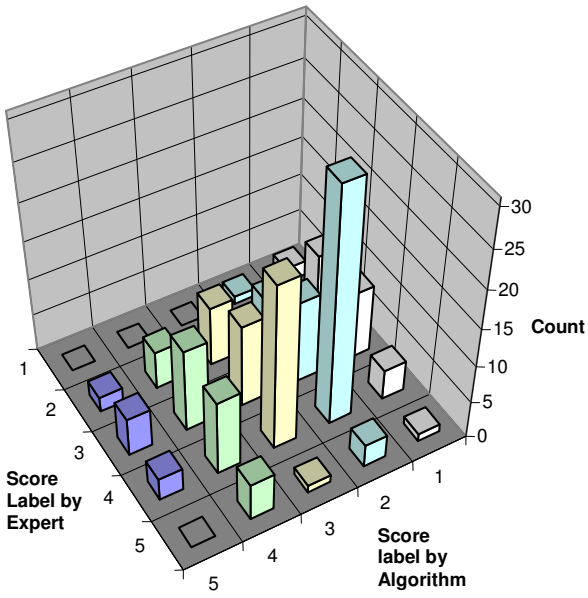


Figure 4.12. Another representation of table 4.5. Height of each column is the count of the number of pairs of scores.

As it can be seen in Table 4.5 and Figure 4.12, the scores that judges assigned to the algorithm-generated labels vary for the higher scores of the expert-assigned label. In the histogram it can be seen that for expert labels that were scored as 3 or 4, the corresponding score for the label assigned by the algorithm varies from 2 to 4, with a significant number of scores where the expert label got a 4 but the algorithm label got a 2.

The general distribution of scores for both labeling methods is shown below:

Score distribution for Hand-Labeled

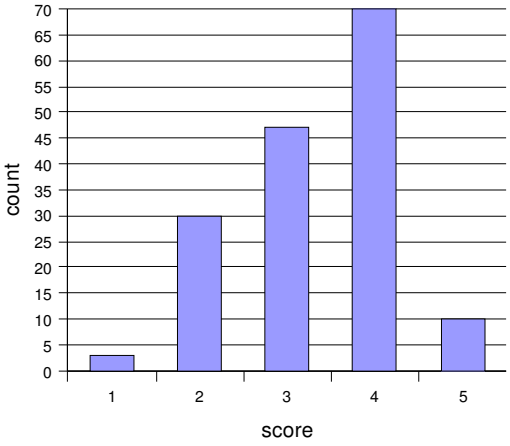


Figure 4.13. Marginal distribution of Table 4.5 for human-assigned labels.

Score distribution for Algorithm

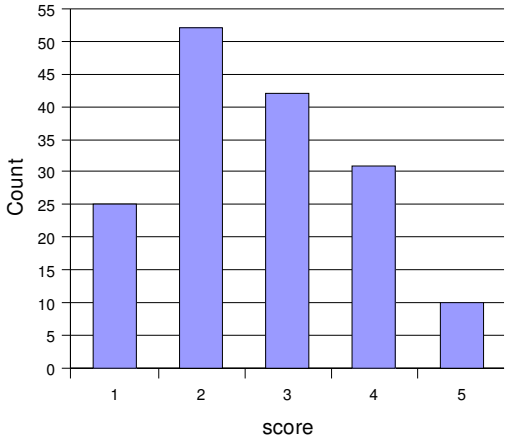


Figure 4.14. Marginal distribution of Table 4.5 for labels assigned by the algorithm.

The histograms for score distributions reveal that expert labels were scored higher than the labels assigned by the algorithm, while scores for the algorithm-generated labels are more evenly spread. Both score distributions are roughly bell-shaped, but the score distribution for the algorithm is not as steep as the one for the expert labels.

4.5.3 Comparing Scores of Expert vs. Algorithm Labeling

The difference in average scores was tested using a paired test of difference in means.

Summary Statistics for Cluster Label Scores

	Mean (at 95% conf.)	**Median**	**Std. Dev.**
Labels by Expert	3.3375 ± 0.143	3.5	0.0725
Labels by Algorithm	2.68125 ± 0.178	3	0.0901

The alternative hypothesis was "Average score for Expert-Average Score for Algorithm<1". $T_{observed}$=1.3505, which locates the real mean difference between 0.8 and 0.85 points, thus within the margin we aimed at. The mean of the difference between expert and algorithm-generated labels is less than 0.85.

5. Evaluation of Stepping Stones and Pathways

Summary

In this section we describe the methods, participants, and results of evaluating Stepping Stones and Pathways. We start by enumerating the problems and factors to consider when evaluating Stepping Stones. Then we describe the setup of the experiment, and the results in terms of quality of connections between topics, user's effort in finding connections, the ability of the system to make clear how the topics are related, and the quality of the support of connections by documents in the collection. We end by analyzing the comments of participants about their experience using Stepping Stones.

5.1 How to test the quality of pathways between topics

The standard measures used to evaluate an IR system are precision (the ratio of retrieved documents that are relevant over total documents retrieved) and recall (the ratio of retrieved documents that are relevant over the total number of relevant documents). These measures were developed assuming the result of a retrieval session is a ranked list of documents, and depend on every document in the collection being marked as relevant or not relevant with respect to a query. In the case of Stepping Stones and Pathways, we are returning connections, not just documents, and so in order to apply the recall/precision measures, we would need for each query an exhaustive list of all possible connections that are relevant. Lists of relevant documents have been built by pooling results from multiple IR systems (as in TREC), or by having a set of people going through every document and query, and assigning it a relevant/non-relevant value (as in the Cranfield experiments [Cleverdon67]). Lack of systems similar to ours to use to pool results and lack of manpower made these approaches impossible for us to follow. There also is another reason why recall/precision measures are not the best way to analyze if our hypotheses about Stepping Stones and Pathways can be supported by experimental data. Our interest is in measuring the advantage of Stepping Stones over standard query methods in helping the user understand the collection through the connections between topics. Measuring understanding and knowledge gained has always been a complicated matter. As Pirolli et al. wrote, when trying to evaluate Scatter/Gather [Pirolli96]:

> *"Although there are some subtle issues involved in the assessment of precision and recall (such as deciding on how to evaluate documents as being "relevant"), even more difficulties arise when trying to assess whether a browsing system is communicating knowledge about the structure of a document collection."*

What we are going to measure then, is not an absolute measure of understanding or knowledge gain or satisfaction of results. Instead we are going to measure relative results compared to the familiar ranked list.

Our approach to evaluation is to compare Stepping Stones and Pathways to an implementation of a retrieval system that uses exactly the same ranking algorithm, but returns a ranked list of documents instead of pathways. We call the latter the Standard Query tool, to differentiate it from the implementation of Stepping Stones.

5.2 Testing Stepping Stones

5.2.1 Method and Participants

One way to test the quality of the results of Stepping Stones when compared to the results of a Standard Query at helping users find connections between topics, is to ask the user to find connections using both systems, and score them. If Stepping Stones connections are valid, and they elicit the relation between topics better than what the user can find by exploring a ranked list, then the "connectedness" of the topics should be higher in Stepping Stones.

In other to do that, we used the same operating systems collection we used to evaluate cluster labeling. From the syllabi, we derived 4 pairs of topics that were related by references. We asked 12 subjects to find and score connections between the topics in each of the 4 query pairs.

Each of the 12 subjects were computer science students, have taken and passed an Operating Systems course, and ranked their knowledge of operating systems as at least 3 on a scale from 1 to 5, as follows: 1=very basic, 2=limited 3=average, 4=good, and 5=expert. They volunteered to participate in the experiment, and were paid after completing it.

Each subject was asked to perform 4 retrieval sessions, that we call runs. The four runs can be considered as divided into two pairs. Each pair of runs corresponded to a query pair (pair of topics), where one run was carried out using Standard Query, and the other run was carried out using Stepping Stones. This means each subject carried out a retrieval session over 2 query pairs (2 runs for each query pair), for a total of 4 runs.

Since each subject performed 2 retrieval sessions with each query pair, even when they were asked to "forget" from one session to another, it is likely that the result of one run with a query pair could affect the result of the next run with the same query pair. To account for the learning effect, the order of runs given by the user alternated between using Stepping Stones and Standard Queries. For each subject that started with Stepping Stones, there is another subject that started using Standard Query. Also for each pair of query pairs (qp_i,qp_j) where there is one subject that started with pair qp_i and then followed with query pair qp_j, there is another subject that did the opposite, starting with query pair qp_i and following with query pair qp_j.

The following table describes the runs and the order of runs assigned to each subject. In the table, each run is coded with 2 letters and a number. The 2 letters in each run represent the retrieval methods used (SQ for Standard Query, and SS for Stepping Stones), and the number between 1 and 4 represents the query pair; thus for example SQ2 in the fourth run of subject 2 means "Subject 2 must use Standard Query with query pair 2 as his/her fourth task".

Participant	Run Sequence			
1	SQ1,	SS1,	SQ2,	SS2
2	SS1,	SQ1,	SS2,	SQ2
3	SQ1,	SS1,	SQ3,	SS3
4	SS3,	SQ3,	SS1,	SQ1
5	SQ2,	SS2,	SQ3,	SS3
6	SS3,	SQ3,	SS2,	SQ2
7	SQ1,	SS1,	SQ4,	SS4
8	SS4,	SQ4,	SS1,	SQ1
9	SQ2,	SS2,	SQ4,	SS4
10	SS4,	SQ4,	SS2,	SQ2
11	SQ3,	SS3,	SQ4,	SS4
12	SS4,	SQ4,	SS3,	SQ3

Table 5.1. List of all tasks for all participants in the experiment, in the order in which they were executed.

Each subject received an email with instructions (see Appendix A for an example of the instructions sent to the participants). The instructions started with a description of the generic task to accomplish (to find and describe how pairs of topics were connected), followed by an example of a connection, to make clear what we expected as a result. The only difference in the email received by different users was the list of URLs leading to the tasks to perform.

Next, the two retrieval methods (Standard Query, Stepping Stones) were briefly described. Since there were two interfaces (one for the Standard Query and one for Stepping Stones), participants were asked to follow two tutorials, each explaining the operation of each tool.

After reading each tutorial, participants had the opportunity, and were advised, to become familiar with the tools before starting the retrieval session. In order to allow the participants to familiarize themselves with the operation of the tools and the task, but at the same time avoid affecting the results of the experiment, they were given a pair of topics that were not any of the topics in the query pairs used for the experiment. We recommended that users spend at least 20 minutes using the tools (in particular Stepping Stones, since the Standard Query returned a typical ranked list of documents they are used to in web search engines).

After they felt they understood how to use both tools, they were asked to perform the 4 retrieval runs. We gave each participant the 2 pair of topics that she was going to be asked to connect, and then the instructions included 4 URLs that the user should go to, one for each run. Additionally, we included a URL to a web page that defined exactly each of the topics in all the query pairs.

The first time they were asked to use Stepping Stones, the run was preceded by a short multiple-choice quiz with 4 questions. The purpose of the quiz was to be sure users did not skip the tutorial (since we found that they did so in pilot tests), and that they understood fundamental aspects of the working of the Stepping Stones tool. Participants were allowed to start the retrieval session with Stepping Stones only after successfully answering the 4 questions. Since subjects may not necessarily complete a run without interruptions or even the same day, if users restarted the same run with more than 30 minutes without actions, then they were asked to answer the quiz again to make sure they did not forget how the Stepping Stones tool worked.

For each run, each subject was asked to return up to 3 connections between the pair of topics, for a total of up to 12 answers per subject. The limit of 3 connections was set, to cap the amount of time participants had to spend on each task and on the whole experiment. We also asked that each run with Stepping Stones should contain at least one answer involving more than one document connecting the pair of topics, whenever possible.

For each connection found, users were asked to answer the following questions:

User Questionnaire (one set of answers per connection reported):
1. What are the titles of the topics involved? 2. Are the topics related? (1=Not at all, Vaguely, Partially, Related but not strongly, 5=Strongly Related) 3. List the titles of one or more documents involved in the connection. 4. Are the documents related? (1=Not at all, Vaguely, Partially, Related but not strongly, 5=Strongly Related) 5. The documents are related to the topics (1=Strongly disagree, Disagree, Neither agree nor disagree, Agree, 5=Strongly agree). 6. Describe the connection between the topics. 7. In a scale from 1 to 5, describe your confidence in your knowledge of Operating Sytems (1=Very basic, 2=Limited 3=Average, 4=Good, 5=Expert).

Table 5.2 Questions asked to the user every time he reported a connection.

The purpose of question 6 was to discard answers that the subject could not explain: If she could not explain the relationship, then the score she assigned to a relation between the topics is really the score of his/her previous knowledge, or of the general understanding gained with the tool, but not of the connection in itself. Fortunately, only 1

connection had to be discarded for this reason. The same user had reported 3 other connections for the same run, so there was still enough data available for analysis after discarding that connection.

5.2.2 The Problem of Bias

A final consideration in the design of the experiment was how to compare the sets of connections for Stepping Stones versus Standard Query. Having a set of queries that are neutral – that is, they do not favor any of the tools – is impossible without a big query set. Any small set of queries like we used is bound to favor one tool over another, because either:

1) Not enough connections are present in the set returned by the Standard Query (bias against Standard Query).

2) The connections are so obvious that the user can quickly identify them with Standard Query (bias against Stepping Stones, since the user is familiar with the operation of Standard Query).

3) The connections are not obvious in Standard Query, but all connections could be found using it (probable bias against Stepping Stones, depending on how easy it is to find a connection with Standard Query).

A further factor that favors Standard Query is that in a sense, subjects are qualifying themselves when finding a connection with Standard Query: since the connection between topics is not elicited by the tool, but must be found by the user, once they find a connection, and in particular if it was a difficult one to find, it is not likely that they give it a low score. In a sense, giving a low score to the connection is to give a low score to their ability to connect the topics.

In view of all this, we decided to put the burden of proof on Stepping Stones, and followed option 3: Connections are not obvious or trivial, but all queries can be answered with both tools.

5.2.3 Query Set

Stepping Stones and Pathways can be used in two ways:

1. to find relationships between specific topics in the user's mind, and
2. to find relationships among topics as a method to explore the collection.

Since our queries were derived from the topics in the syllabi, they will belong to case 2, since the topic names in the syllabi will be generic – they need to be, as they are going to be a high-level description of what is in common among the papers on the topic. To

explore case 1, we would need experts in the content of the collection to split queries for us. After that we would have to check if the connections can be found with the current content of the collection. We chose to address only case 2 in order to have a guarantee by the structure of the collection and syllabi that the topics were connected through other topics, and that we had documents in the collection associated with all the topics involved.

The four query pairs used in the experiment are listed in Table 5.3:

Query Pair Number	Query Pairs	Number of documents in result of Standard Query	Number of connections in result of Stepping Stones	Number of documents in result of Stepping Stones
1	"web server" "fault tolerance"	18	3	4
2	"distributed system" "ipc"	20	45	22
3	"extensible operating system" "virtual machine"	20	50+	26+
4	"file system" "web server"	20	27	26+

Table 5.3. Query pairs used in the experiment, along with the number of documents and connections returned by Stepping Stones and Standard Query.

The limit of results in Standard Query was set to 20. Query 1 in Table 5.3 returned 18 documents because queries were tried first by matching all the words, in which case the result set of query 1 in Standard Query had 18 documents. "Number of Connections" refers to the number of connections between topics returned. "Number of documents in result of Stepping Stones" refers to the number of unique documents involved in all connections reported by Stepping Stones.

A cursory view of the number of documents retrieved with each technique in Table 5.3 above reveals that the number of documents in the result of Stepping Stones is higher than in Standard Query. The reason to have more documents returned with Stepping Stones than with Standard Query is that we wanted to use the count of number of documents examined as a measure to compare the effort to find a valid connection with each tool. If Standard Query is better than Stepping Stones to find connections, and Stepping Stones connections are not valid, then the number of documents explored in Stepping Stones should be higher, since it should take more connections explored (and hence more documents) to find a valid connection. Alternatively, if Stepping Stones

connections are valid, then the user should need to explore fewer connections (and in turn, less documents) in Stepping Stones, even though the total number of documents involved in Stepping Stones is higher.

Having more documents involved in Stepping Stones may seem like a way to favor it. It could be argued that involving more documents implies more opportunities for connections. However this was not the case, for two reasons:

1) We only asked for up to 3 connections, thus limiting the number of documents that needed to be involved.

2) Up to 3 connections could be found with both tools (we checked this in a preliminary study).

5.2.4 Query Preprocessing

To improve the quality of results with both Stepping Stones and Standard Query, a query without qualifiers was treated as meaning "all words are mandatory". That is the reason why query 1 in Standard Query only returned 18 documents.

When the result set with all words mandatory was empty, all word qualifiers were relaxed to "all words are optional" with the Standard Query. In the case of Stepping Stones, each sub-query was relaxed, one at a time, until a non-empty result set was returned.

5.3 Results

5.3.1 Quality of Connections between Query Topics

For each user, we obtained two pairs of sets of scores through question 2 of the user's questionnaire: one pair of set of scores for each pair of topics (query pair). One of those score sets in each pair belongs to Stepping Stones, and the other to Standard Query. Each pair of score sets came from the same user, to account for previous user knowledge.

In order to be able to test scores comparing pairs from the same user, we need to reduce the two pairs of score sets to 2 pairs of scores (4 values). Since a user returns multiple results for a query pair, and the number of results varies not only within query pairs but for each run, we normalized the score sets as follows:

1. We assumed the user performed the runs by doing his/her "best effort". For example, if for query pair 1 the user found 2 connections with Standard Query and 3 connections with Stepping Stones, we assume that he/she did his best effort, but was unable to find more than 2 connections with Standard Query.

2. An additional consideration is that for each pair of sets of scores, taking all the values in the sets with more scores would penalize that set, since a set with more scores is likely to contain lower values than a set with only 1 or 2 values. Then, for each pair of sets of scores, for the smaller set we take all the scores, and for the bigger set, we take the top number of scores equal to the size of the smaller set. This leaves us with two sets of each size. The final score representing each set is the median of each score set. The reason we used the median instead of the mean is that scores are user's judgments, and although coded in a 1-5 scale, the distance in the user's mind between a connection score of 1 (not at all) and 2 (vaguely) does not need to be the same as between 4 (related, but not strongly) and 5 (strongly related). Thus we use the median, since it is a synthetic score, but a value that appeared (except when the number of value is even) in the set of user judgments.

The final result is 2 pairs of scores for each participant, for a total of 24 score pairs. The following scatter plot shows all the pairs. This scatter plot reads as follows: each point in the scatter plot is a pair of score values. For each user and query pair, the value in the X axis is the score assigned for Standard Query, and the Y value is the corresponding score for Stepping Stones. The number next to the dot in the scatter plot is the number of points with that score combination of score values, for all participants. For example, the dot at (4,3) represents the number of score pairs for which participants assigned a median score of 4 to Standard Query and a Median score of 3 to Stepping Stones. In the scatter plot below (Figure 5.1), the number next to (4,3) is 1, meaning there is only 1 pair of scores like it.

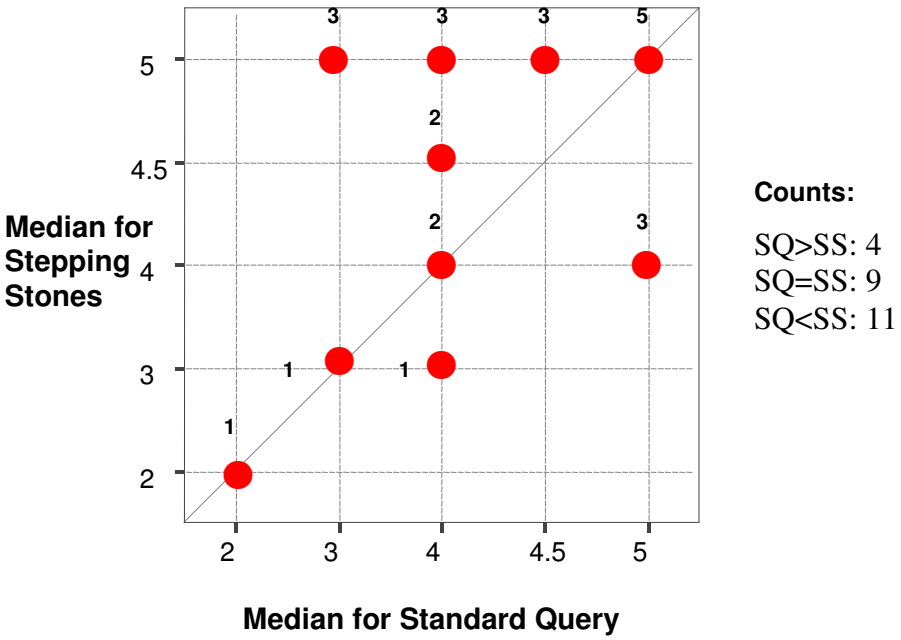


Figure 5.1. Scatter plot of the count of Score-pair of Medians for each query in Stepping Stones and in Standard **Query**. Each dot is a combination of scores, one for Standard Query and one for Stepping Stones. The number next to each point indicates the count of pairs with that score combination. Numbers over the diagonal are counts where Stepping Stones performed better than Standard Query; numbers below the diagonal indicate the opposite.

The scatter plot shows that there are more score pairs where the median value of the score of connections for Stepping Stones is higher than for Standard Query. The following two histograms show the score distributions for each case:

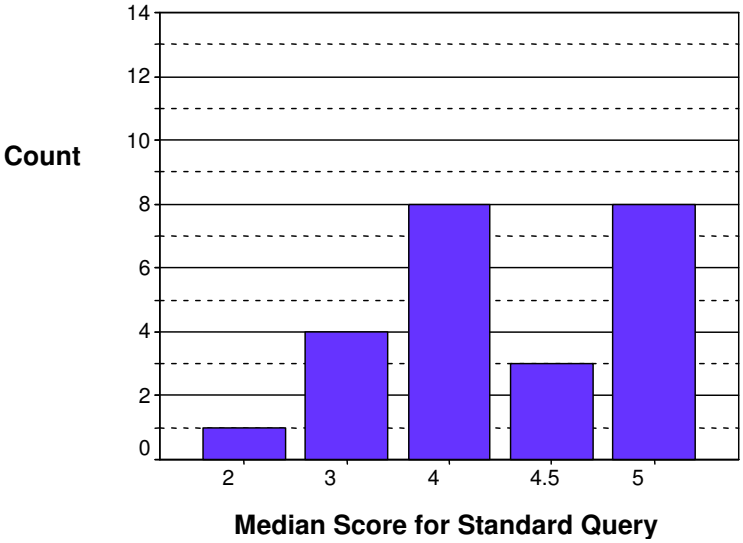


Figure 5.2. Distribution of median of scores of connectedness between topics for Standard Query.

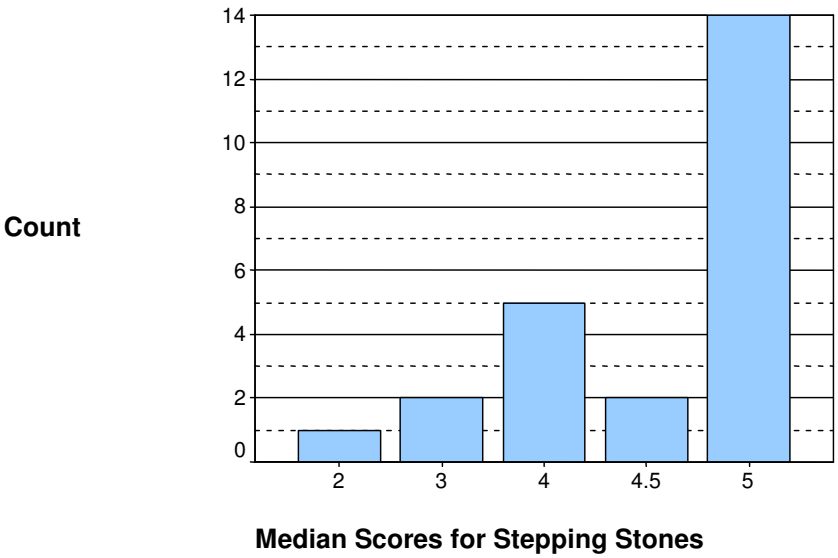


Figure 5.3. Distribution of median of scores of connectedness between topics for Stepping Stones.

The difference in medians for each pair of scores is approximately normal with mean 0.3 and standard deviation 0.88, with a significance of 0.321 in the Kolmogorov-Smirnov test of normality for difference of means (do not reject normality). Thus we can test the alternative hypothesis "average of median scores for Stepping Stones is higher than for Standard Query" with a paired t-test. Tables 5.4a and 5.4.b show the results of the test.

Summary Statistics for Median of SQ vs. Median of SS

	Mean	Std. Deviation
Median Standard Query	4.146	0.8403
Median Stepping Stones	4.458	0.8198

Paired Samples Test for difference of means: Median of SQ vs. Median of SS

Paired Differences					t	df	Sig. (2-tailed)
Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference				
.3125	0.88234	0.18011	0.685	-0.0601	1.735	23	0.096

Tables 5.4a and 5.4b. Summary statistics and results of hypothesis testing for topic connectedness.

The critical value when testing the alternative hypothesis "average of median scores for Stepping Stones is higher than for Standard Query" is $t_{critical}$ (0.05, 23)=1.714<$t_{observed}$=1.735, so we affirm that Paired T-Test shows difference between SQ and SS scores, with the average median of SS being higher than the average median of SQ at α=0.05. Scores in each pair, however, are weakly correlated (correlation 0.435 with significance 0.34). Moreover the significance of the critical value is very close to α, so perhaps a safer assertion to make with confidence would be that connections between topics in Stepping Stones are at least as good and no worse than the ones found by the user with Standard Query.

5.3.2 Difference in Pathways found by Standard Query and Stepping Stones

Participants reported a total of 61 different pathways between topics. The sizes of the biggest set of connections reported by any user for each query and search method are:

Query Pair	Standard Query	Stepping Stones
1	5	5
2	5	3
3	3	3
4	3	3

Table 5.5. Highest number of connections reported for each query and each search method.

If all users had agreed on the set of pathways reported, they would have reported a total of 36 pathways. This number comes from assuming that all users agreed on the answers: for any two participants p_1 and p_2, returning the sets of answers (SQ_{p1}, SS_{p2}) and (SQ_{p2}, SS_{p2}) respectively, then either $SQ_{p1} \subset SQ_{p2}$ or $SQ_{p2} \subset SQ_{p1}$, and either $SS_{p1} \subset SS_{p2}$ or $SS_{p2} \subset SS_{p1}$. In other words if all participants would have agreed on the pathways, then all sets of pathways should be a subset of the biggest set returned for each query and search method. It is important to take into account, though, that not all participants returned the same number of pathways.

The following diagram depicts the overlapping of unique pathways in Standard Query and Stepping Stones for all queries:

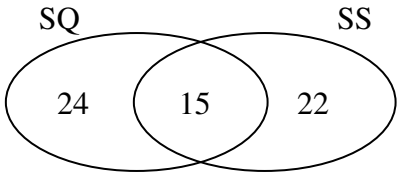


Figure 5.4. Overlapping of unique pathways reported by user in Standard Query and Stepping Stones for all queries.

There were 51 pathways in Standard Query (12 of them duplicates of another pathway reported with Standard Query), and 50 pathways in Stepping Stones (13 of them duplicates of pathways only in Stepping Stones). The intersection had a total of 47 pathways, 32 of them duplicates. Testing the difference in scores for SQ and SS in the intersection did not give any statistically significant differences when taken as a whole or split by query. This effect, however, can be due to the small set size.

From Figure 5.4 it is clear that users found different sets of connections by themselves from the set suggested by Stepping Stones. Interestingly, as we mentioned before with regard to Tables 5.4a and 5.4b, the score of connections between topics for Stepping Stones was at least the same as for pathways found by the user with Standard Query. Therefore we can affirm that Stepping Stones was useful at providing users with connections that are valid, non-obvious, and different from the ones the user would find by himself.

As an example, we list here all documents in all pathways reported by all users for query pair 1, “Web Server” and “Fault Tolerance”. Observations on the pathways in this example are common to the pathways all other query pairs: there is a certain agreement on the single-document pathways, but pathways involving many documents differ greatly between Standard Query and Stepping Stones. While this result is in part expected (longer pathways allow for more combinations of documents), it is interesting to note how different users consider different pathways as relevant, even if they were asked to report the best (or first) three pathways they could find.

Query 1: "Web Server" and "Fault Tolerance"		
Documents in Pathway	**SQ**	**SS**
Overload control mechanisms for Web Servers	1	0
Flash: An efficient and portable web server	1	0
Implementation and Evaluation of Transparent Fault Tolerant Web Service with Kernel-Level Support	3	2
Enhancing Web Performance	1	0
Using Fault Injection to Evaluate the Performability of Cluster-Based Services	1	1
Extensible Virtual Machines + System Call Support in an Extensible Operating System	1	0
Enhancing Web Performance + memory servers for multicomputers + Over Infrastructure for Survivable Servers and Routers	1	0
Implementation and Evaluation of Transparent Fault-Tolerant Web Service with Kernel-Level Support+ A Fault-Tolerant Video Server using Combined Raid 5 and Mirroring	1	0
Fault-Tolerant Distributed Garbage collection in a Client-Server Object-Oriented Database + fault tolerant matrix operations for Parallel and Distributed Systems	1	0
Fault-Tolerant Sequencer + fault Tolerant Matrix Operations for Parallel and Distributed Systems	1	0
Fault-Tolerant Sequencer + Implementation and Evaluation of Transparent Fault Tolerant Web Service with Kernel-Level Support + Using Fault Model Enforcement to Improve Availability	1	0
Algorithm-based Diskless Checkpointing for Fault-Tolerant Matrix Operations + Availability in the Sprite Distributed file system + Fault tolerant Matrix Operations for Networks of workstations: Using Diskless Checkpointing + Lessons from FTM: An Experiment in Design and Implementation of a Low-Cost Fault-Tolerant System.	0	1
Exokernel: An Operating System Architecture ;operating system resource reservation for Real-Time and Multimedia Applications + shared libraries in an exokernel operating System + Ecosystem: Managing Energy as a First-Class Operating System Resource	0	1
Implementation and Evaluation of Transparent Fault Tolerant Web Service with Kernel-Level Support+ evaluating the impact of communication architecture on the Performability of Cluster-Based Services	0	1
Implementation and evaluation of Transparent Fault-Tolerant Web-service with Kernel-Level Support + Using Fault Injection to Evaluate the Performability of Cluster-based Services	0	1
Client-Centered Load Distribution a Mechanism for Constructing Responsive Web Services+ Document-replication and Distribution in Extensible Geographically Distributed Web Server + Web Workloads: Influencing Disconnected Service Access	0	1
Measuring the Capacity of a Web Server under Realistic Loads + Load	0	1

Balancing in Distributed Web Server Systems with Partial Document Replication + Instantaneous Offloading of Transient Web Server Load + Client Centered Load Distribution a Mechanism for Constructing Responsive Web Services		

Table 5.6. All documents pathways reported by all users for query pair 1. . The second and third columns are the number of users that have reported that particular set of documents for a pathway when using Standard Query and Stepping Stones, respectively.

5.3.3 Effort in Finding Connections

We used the number of unique documents inspected by the subjects before reporting each connection as a measure of the effort it took to find a valid connection; the fewer unique documents examined, the less effort it took for the user to find a valid connection that she could explain. From the log of user actions we counted the number of documents the user requested before reporting a new connection with each tool.

By "number of unique documents" we mean that each document was counted only once for every connection reported by the user, even if it was explored more than once by the participant.

For each query, we calculated the average number of documents inspected per connection reported by the user. We paired the data as with the connected score study, but this time we used the averages, since the data source (document count) is numeric, not just ordinal. Again, we have 2 pairs of values for each of the 12 participants, 1 pair of values for each query pair the user went through, for a total of 24 data points. The alternative hypothesis to test was "The average number of unique documents inspected per connection reported by the user while using Standard Query was higher than the same average with Stepping Stones".

The distribution of differences is approximately normal (Kolmogorov-Smirnov Z=1.164 with significance 0.133, do not reject normality), so we tested again using paired samples. Tables 5.7.a and 5.7.b show the results of the test.

Summary statistics for average number of documents inspected per connection

	Mean	**Std. Dev**
Standard Query	5.9063	4.5401
Stepping Stones	3.2308	1.5611

Paired Samples Test

Paired Differences					t	df	Sig. (2-tailed)
Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference				
2.6754	4.55162	0.92909	0.7534	4.5974	2.88	23	0.008

Tables 5.7.a and 5.7.b. Summary statistics and results of hypothesis testing average number of documents inspected per connection.

The critical value is $t_{critical}$ (α=0.05, df=23)=1.714<$t_{observed}$=2.88, so we affirm that the Paired T-Test shows a statistically significant difference between SQ and SS scores, with the average number of unique documents inspected per connection reported by the user while using Standard Query being higher than the same average with Stepping Stones, at α=0.05. Scores in each pair are very weakly correlated (correlation 0.165 with significance 0.442), with the highest variation corresponding to Standard Queries (compare with the standard deviations in Table 5.7.a). On average, it took about 80% more documents to find a connection with Standard Query than with Stepping Stones.

The following histogram shows the distribution of the average number of documents explored per connection. By looking at the histogram, it can be noted that the number of cases where Stepping Stones is higher than Standard Query is for an average of 4 unique documents of less. After that, Standard Query has a higher number of cases than Stepping Stones, and in particular the right half of the histogram (the one with the higher average of documents explored) is almost completely dominated by Standard Query.

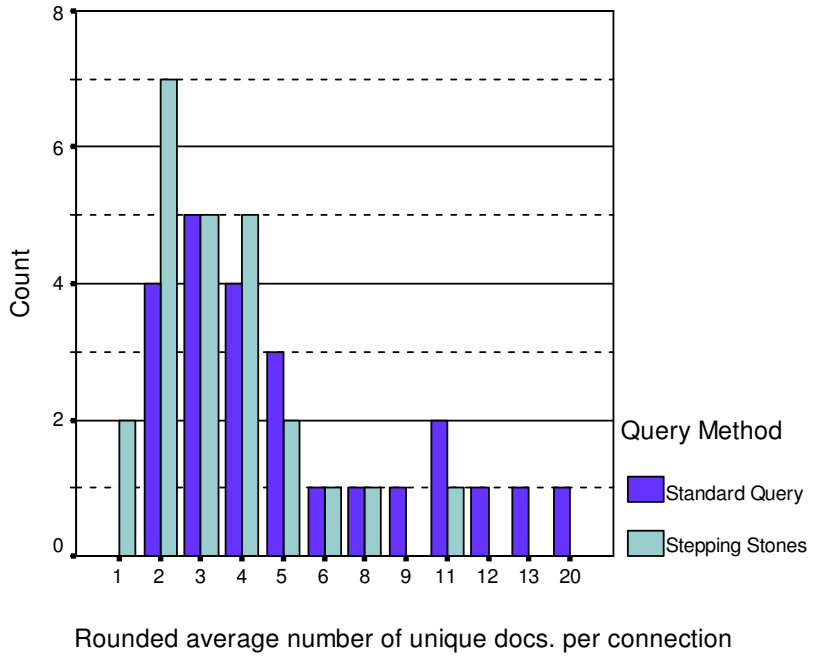


Figure 5.5. Distribution of average number of unique documents inspected per reported connection

Somewhat of an anomaly was the difference in the number of documents for Query Pair 1 (see Table 5.3). This was the only query pair where Stepping Stones returned fewer documents than Standard Query.

The summary statistics for every query pair are shown below in Table 5.7:

	Method	Min	Max	Mean	Std. Dev.
Query Pair 1	**Standard Query**	1.75	20	6.875	7.249
	Stepping Stones	1	4.2	2.506	1.155
Query Pair 2	**Standard Query**	2	13	6.888	4.481
	Stepping Stones	2.33	4.5	3.314	0.807
Query Pair 3	**Standard Query**	3	12	5.361	3.436
	Stepping Stones	2	10.5	4.139	3.253
Query Pair 4	**Standard Query**	3	9	4.5	2.324
	Stepping Stones	1.33	8	4.306	2.522

Table 5.8. Average Unique Number of Documents Inspected per Connection for Query Pairs 1-4.

As can be seen in the fifth column of Table 5.9, the mean for Standard Query is higher than for Stepping Stones for all 4 query pairs, although it is higher in query pair 1 than in the other queries.

The same paired test of means, but this time excluding all results with query pair 1 (total of 18 pairs of data points), returned the results shown in Tables 5.9.a and 5.9.b:

Summary statistics for average unique documents inspected per connection, excluding query pair 1

	Mean	Std. Deviation
Standard Query	5.583	3.463
Stepping Stones	3.473	1.62999

Paired Samples T-Test for average unique documents inspected per connection, excluding query pair 1

Paired Differences					t	df	Sig. (2-tailed)
Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference				
2.1106	3.45094	0.81339	0.3944	3.8267	2.595	17	0.019

Tables 5.9.a and 5.9.b. Summary statistics and results of hypothesis testing average number of documents inspected per connection after excluding query pair 1.

Then, once again $t_{critical}$ (0.05,17)=1.74 < $t_{observed}$=2.595, and at α=0.05 we reject the null hypothesis and conclude that the average number of documents explored per connection, even after excluding query pair 1, is higher in Standard Query than in Stepping Stones at α=0.05.

Thus, query 1 cannot account for the difference in the average number of documents inspected per connection. The higher number of documents inspected cannot be attributed to the number of documents returned, since except for query 1 all other queries returned significantly more documents in Stepping Stones than in Standard Query, yet it took fewer documents to find a valid connection with Stepping Stones than in Standard Query. Therefore we attribute this effect to the fact that the connections in Stepping Stones were almost always valid, therefore requiring less effort for the user to find a valid connection.

5.3.4 Average Time to Find a Connection

Unfortunately, since the experiment was performed online, time was not a controlled measure, and was found to be very difficult to estimate correctly from the logs. Even after measuring the time since the first document is explored until a connection is reported, and trying to account for variations in the usage of the tools, 8 of the 24 data pairs had one average time of more than 20 minutes between reporting each connection (not 1 task, but 1 connection), strongly indicating that the user interrupted the task and continued later. After eliminating the 8 pairs with average time per connection reported greater than 20 minutes, leaving 12 data pairs, the difference in means for average time per connection in SS vs. SQ was found to be not significant ($t_{observed}$=0.924, 15 df), although we cannot conclude that this indicates that the time spent per connection was about the same, due to the 8 point pairs that needed to be removed as outliers.

5.3.5 Explaining the Connections: Degree of Relationship between Documents in a Connection

The connection between topics is embodied in the text of the documents that make the connection. A connection between topics is as clear as the user's understanding of how the documents supporting a connection elicit that connection.

Due to users' misunderstanding of the instructions to answer question 4 of the questionnaire for each connection, we were not able to get data pairs for all connections reported. Some participants did not report this number, or reported them inconsistently. Out of 42 connections with more than one document, we only received a document relation score for 21 of them; 11 for Standard Query, and 10 for Stepping Stones. The

score distribution for the relation between documents in SQ is not normal (Kolmogorov-Smirnov Z=1.442, significance 0.031), so we relied on non-parametric testing.

Summary statistics for score of relationship between documents in connections

	Sample Size	Mean	Std. Deviation
Standard Query	11	3.36	1.3652
Stepping Stones	10	3.6	1.075

Table 5.10. Summary statistics for score of relationship between documents in a connection.

For the two groups of document relationship scores, one for Standard Query and one for Stepping Stones, we observed a Mann-Whitney *U*=51, for a 2-tailed significance of 0.809 at α=0.05. Although the difference in means is in favor of Standard Query, we cannot statistically conclude that the score connection between documents is higher in Standard Query than in Stepping Stones, probably due to the small sample size. The reason for the higher average score is likely to be that, as shown in the boxplot below (Figure 5.6), scores for Standard Queries are less spread than scores for Stepping Stones.

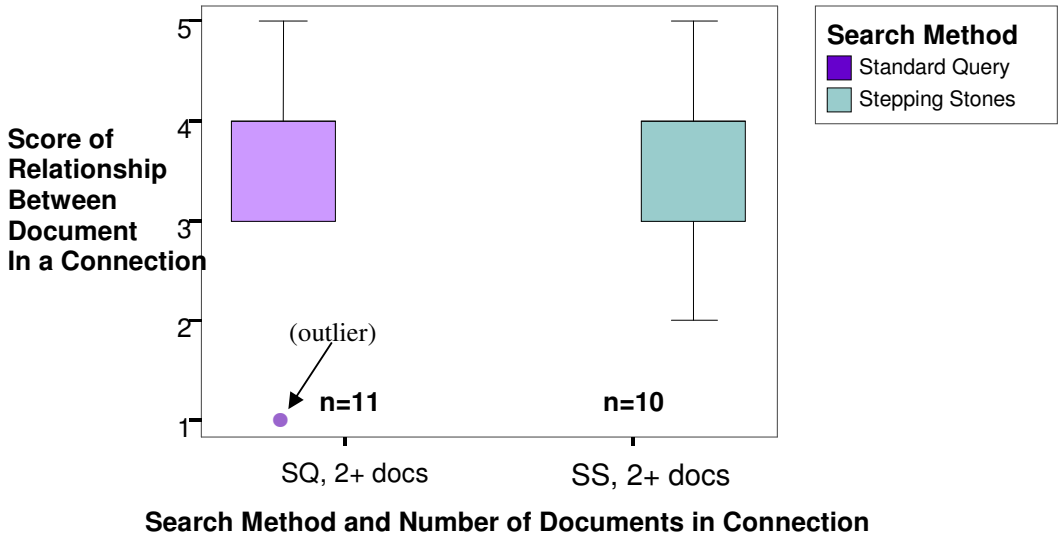


Figure 5.6. Box plot for scores of relationships between documents in a connection, for Standard Query and Stepping Stone search methods. Only connections involving 2 or more documents are counted; it does not make sense to score the relationship of 1 document to itself.

5.3.6 Support of Connections between Topics: Topic/Document Relationships

Every pathway returned by Stepping Stones is made of topics. Every connection between every pair of topics is supported by 1 or 2 documents. Even if a connection seems valid, according to the topic titles, it may not be valid from the point of view of the documents chosen by the system to support that connection. The purpose of question 5 in the questionnaire, filled by the participants for every connection, is to measure how well the documents can be associated by the participant to the topics in the connection.

We tested the scores given to Relation between Topic and Documents in a Connection in Standard Query and Stepping Stones by a Test for Equality of Means, comparing Standard Query and Stepping Stones with an Independent Samples Test. Since some of the data points are missing because of lack of answers from participants, equal variance was not assumed. The alternative hypothesis was "Average score for Stepping Stones is different from average score for Standard Query", tested at α=0.05.

Summary statistics for scores of relation between topic and documents in a connection

	N	Mean	Std. Deviation
Standard Query	43	4.21	0.600
Stepping Stones	44	3.82	0.843

T-Test of independent samples, equal variances not assumed

t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	95% Confidence Interval of the Difference	
2.498	77.749	0.015	0.39	0.157	0.079	0.703

Tables 5.11.a and 5.11.b. Summary statistics and results of hypothesis testing for scores of Topic/Document relationships.

The null hypothesis can be rejected at α=0.05 ($t_{observed}$=2.498, $t_{critical}$ (0.025,78)=2.048), and average scores for document/topic degree of relationship for Standard Query are higher than for Stepping Stones (4.21 versus 3.82).

Originally we thought this difference could be attributed to pathways made by the subjects containing mostly 2 topics while Stepping Stone pathways always consist of 3 topics or more. Since it is easier to connect documents to a smaller set of topics, they would accordingly have a higher score.

However, that is not the cause. The average scores for pathways in Standard Query is always higher than for Stepping Stones, regardless of whether they involve 2 or 3 topics, as seen in the following table:

Topic/Doc. Rel in SQ	N	Minimum Score	Maximum Score	Mean Score	Std. Deviation of Score
With 2 Topics	14	4	5	4.43	0.514
With 3 Topics	29	3	5	4.1	0.618

Table 5.12. Summary statistics for the score given to the Documents/Topics relationships in Standard Query, for pathways involving 2 and 3 topics.

Comparing the mean of topic/document relationships in Stepping Stones from Table 5.11.a with the means in Table 5.12, it is clear that the means are always higher for Standard Query, regardless of whether 2 or 3 topics are involved.

The only obvious explanation we can find is that users are better at labeling the topics for the pathways they found by themselves than the algorithm is at labeling its own topics, which coincides with the results of the cluster labeling evaluation.

An additional factor affecting the topic/document scores is that users may be scoring their own connections higher because they created them, and in some way it could be said that by scoring the topic/document relations, they are scoring their own work. There are 7 connections that were found as valid by the same subject for the same query pair both by using Standard Query (found by user) and Stepping Stones (found by the system). They are listed here:

User	Query Pair	Topic/Doc Relatedness Score for SQ	Topic/Doc Relatedness Score for SS
1	1	4	4
2	3	4	4
14	1	4	5
17	1	5	4
10	4	4	4
11	3	3	4
12	4	5	2

Table 5.13. Scores given in SQ and SS for the same document, in 1-document connections.

For perfectly fair subjects, the median for topic/document scores for Standard Query ($\hat{\mu}_{SQ}$) should be the same as for Stepping Stones ($\hat{\mu}_{SS}$). However, for these 7 cases, $\hat{\mu}_{SQ}$=4.1429 > $\hat{\mu}_{SS}$=3.85. While the difference is not statistically significant at α=0.05,

due almost surely to the small sample size, it shows that users may tend to assign a higher score to themselves than to SSPW, as did user 12, for example.

5.4 Comments of Experiment Participants about the Experience of Using the Stepping Stones Tool

After returning the results and being paid, participants in the experiment were asked to voluntarily comment about any aspects of using Stepping Stones that they liked or disliked. Here we transcribe all the comments sent by participants, only edited for syntax and grammar.

User Comment #1:

I spent a good bit of time playing with each of the methods for query, especially the tool that returned connections [referring to Stepping Stones]. *I thought it was quite an interesting way of doing things. Clearly, I did not use much of that tool in the work that I did and I think the reason after much thought is that I am so entrenched in doing online searches in one way that it is difficult to get out of that mode. I am simply used to laborious checks through document after document.*

It certainly was interesting to use.

User Comment #2:

I think the size of data set was too small to find good relations between two topics. And I believe we should treat topics as atomic concepts. I mean, for example, the topic "file system" is not related with the topic "system" or the topic "file". During most of my tries to trace the connections, I lost my way because of these "distracted topics".
Anyway, I liked the graphical interface showing paths between two concepts. It was a new experience.

User Comment #3:

Well, it would be great if all the connections were grouped together. For example, for each of the article listings, there were different associations (or connections) with each article. Many of the articles related to 'extensible operating systems' and 'virtual machines' had several connections associated w/ them (i.e., 'remote procedure',

'distributed operating system'). It would be great if there could be another grouping based on these connections (so all the articles for 'remote procedure' as they relate to 'extensible operating systems' and 'virtual machines') could be grouped together. (This is essentially the task we were required to perform for this experiment, so this may be outside the scope of (or unnecessary for) the Stepping Stones and Pathways system.

User Comment #4:

Regarding the experiment itself, since I had prior knowledge of OS and networking, I didn't have too much trouble with the connections in either tool. I found the normal search tool slightly more intuitive since I guess I'm used to that interface and it took a while for me to learn your tool. And it would have been helpful for the highlighted portions of the tool to be in maybe a slightly different color if an exact match of the whole phrase was found. For example, right now it would mark in yellow the term "systems" even if I was looking for the terms "distributed systems" or "file systems". If the complete phrase "distributed systems" was marked in some other color, it would make be easier to grasp the connection.

User Comment #5:

It's a good idea, but it is very difficult to generate the relationship maps on the fly and have them make sense. You might consider making a textbook-like thing out of it; it could be a good study aid, and you could construct it so that all the relations among the documents made sense.

User Comment #6:

Here are a few usability issues.

1) For the testing, I would have liked to view the connections I already submitted.
2) In the graph view I would have liked to zoom in all connections one or zoom back one.
3) I'd like I could search so it finds "web server" only, and not just occurrences of "web" or "server".

It's a really neat tool; I'd be interested to see how it performs with a whole internet search.

User Comment #7:

One problem with the instructions I am having is that you are asking for the connections between the two topics and when I use the normal method, I have to find them but your system basically lays them out in front of me. I agree with the breakdowns of your tool. The problem I have is whether or not this is the best evaluation of your system. I mean, if this test is to determine if your tool can discover links between topics, well yes, it does, and I agree with it.

With the Standard Query, I have to try and organize everything but with your tool, it lines everything up into topics. When you ask for information about a connection, there is nothing to look for as the information is just lined out in front of you.

I never really figured out what the "See all connections ordered by importance" meant so I didn't use it.

The images of connections can grow quickly so I copied them to Photoshop to shrink them down.

Your system seems very dependent upon the initial choice of topics. For Distributed Systems and IPC, it breaks it down by three topics and then if you open them up, they have many of the same subtopics. It might be useful to order by those subtopics instead of the first breakdown. How does your system manage this? One concept would be for a human to enter a connection and your system fill-in the documents that support it so for Distributed System and IPC, some topics would be performance, application and operating system.

I liked the breakdown of your tool for the first two topics.

User Comment # 8:

I am very impressed with the Stepping Stones and Pathways. It brought together topics in ways I would have not seen just using the regular search tool. Nice work!

There are some common themes along all the user's comments. All users that sent comments (8 out of 12) seem to have liked the tool, and there is an agreement on its usefulness at finding connections, although users find difficult to change the way they think of retrieval, from a ranked list to a list of connections. It is possible that their opinion was influenced by the desire to please the researcher, but we asked for the comments only after they finished the experiment and were paid, and the comments were

sent by email, not given in person, so as to diminish any pressure the participants could feel about having to give positive comments.

Many of the comments mention the problem of partial matching (for example comment #2), where they feel the system is not answering the query they posed, but instead matching individual words. In reality, as explaining in the "Query Pre-Processing" section in this chapter, the system was matching all words (all query words were considered as mandatory unless the result set was empty). However when a document was shown by the user interface, query words were individually highlighted in yellow along the document, whether they appeared one after another or not. Thus, participants had the impression that the system was not trying to match all words, but just the ones it could match. Also, coloring different topics in different colors would help.

Another problem mentioned is the expansion in size of the connection graph (comments #6 and #7). Due to the multiple views offered simultaneously by the Stepping Stones tool, screen real estate becomes scarce very quickly. In particular the connection graph is the area where this problem is more acute, since its size grows rapidly when adding more intermediate topics. Scrolling is not a good solution here, because the connection graph is meant to be an overview of all the topics and how they are related, and so it should be possible to see it all at the same time. A solution, as mentioned in the comments, is to add a zoom/pan capability to the connection graph, such that the user can control whether to see the whole graph, or to read individual parts of it.

Finally, comment #7 mentions that he/she would have found it useful to suggest intermediate topics to the tool, and make the tool find the connection for those intermediate topics, instead of suggesting some. We did not include this feature in the testing version since we wanted the participant to give us a judgment value on the topics and connections suggested by the system. From an implementation point of view, it is easier to build connections from topics the user suggests than to find them. We agree that adding the ability to suggest topics would enhance the utility of Stepping Stones as a way to explore collections, through a combination of some topics suggested by the system, and other hypothesized by the user, as she builds an understanding of the collection content.

6. Conclusions and Future Work

Summary

In this final chapter we summarize our findings in terms of the success of Stepping Stones as an alternative tool for information retrieval. We follow by describing some topics that we did not have the opportunity to explore as we would have wished, and that we think are worth pursuing as further research.

6.1 Conclusions

It is possible to automatically build specialized collections of connected topics from course syllabi. We have proven that is possible to build small, connected collections with a subset of labeled documents for the areas of Operating Systems, Information Retrieval, and Data Mining in computer science. These areas were selected after researching the syllabi availability over many areas of computer science. The method we used to harvest specialized collections of connected topics is not restricted to any particular area of science, but on the availability of syllabi based on research papers instead of textbooks, and the availability of those papers. Computer Science professors have always been adept at using technology in the classroom, and so we hypothesize they are more likely to publish their syllabi online, and the existence of public research paper databases like CiteSeer give a good chance that a syllabus can be based on online sources. It is certainly possible that other branches of science are not as adept at publishing their syllabi online, or building the syllabi around research papers instead of textbooks. A good candidate for an alternative branch of science to harvest papers and syllabi is Physics. The physics community has by necessity always explored new technologies: the World Wide Web started in CERN (a European physics lab) after all, and the ArXiv collection of physics preprints is one of the most successful examples of digital libraries.

We have developed a novel method to automatically find connections between topics without depending on keywords or classification systems. The closest system to ours in terms of purpose and functionality is Arrowsmith that also aims at connecting seemingly unconnected documents, and returns connections instead of a ranked list of documents. There are some important differences between Arrowsmith and Stepping Stones:

- Arrowsmith is a knowledge discovery tool about a precise kind of relationship in the realm of medicine: illness→ (symptom, effect)←treatment. Stepping Stones is a more generic tool to find relationships in any set of scientific papers, and so it trades flexibility with generality: relations found by Stepping Stones are going to be more generic than those found by Arrowsmith.
- To find a connection, Arrowsmith matches to 2 documents by common terms; Stepping Stones matches from 1 to 3 documents.

- Arrowsmith gives chains up to length 2. We can extend a chain.
- Arrowsmith connects terms; Stepping Stones connects topics.
- Arrowsmith uses common keywords and the PubMed classification system to make the match understandable. We match whole documents instead, but we give more importance to title matching by using Boolean matching over title, and a high threshold. This has the effect most of the time of connecting documents with related titles, which helps at identifying relations by title scanning, and is what we observed most users do.

This is not to say that our system is superior to Arrowsmith. The big advantage of Arrowsmith over Stepping Stones is that by relying on keywords and classification systems, it can give more precise answers about the particular connection between two terms. Stepping Stones performs matching by document sections, both in terms of document content and citations, and gives high importance to the relationship between document titles to make the connection clear to the user. The advantage of Stepping Stones over Arrowsmith is that it does not depend on an external classification system or keywords, and that since the answers are less specific than the ones by Arrowsmith, it fits with the needs of its intended user, a person with knowledge of the subject matter but not of the specifics in the collection.

Stepping Stones and Pathways supports the user in finding valid answers to queries of the form "how is X related to Y?". Users of Stepping Stones had a positive response to the connections suggested by the tool, giving high scores to the quality of connection between topics. The median of scores for connections in Stepping Stones is a 5 ("topics are strongly related", the maximum value), with a 95% confidence interval for the mean between 4.06 and 4.38, which places the answers for Stepping Stones between "Related" and "Strongly Related". Of the 50 connections, only 7 had scores of 2 (vaguely related) and 3 (partially related). The high scores are not due to users repeatedly giving high scores to a small set of connections. As mentioned in the previous chapter, participants reported 61 different connections between the topics in Stepping Stones for the 4 queries.

Connections provided by Stepping Stones and Pathways are as good as connections found by users with more effort. The quality of connection between topics in Stepping Stones was statistically better, or at least as good as the connections that users found by themselves, both in terms of relationships between topics and between documents supporting and explaining the connections. Hand-made connections were only superior in terms of document/topic relationships, probably because users are better at labeling groups of documents than the labeling algorithm we used, so the human-assigned label is closer to the content of the documents. For every user, the average number of documents needed to find a valid connection by browsing a ranked list was almost twice the number needed to find a valid connection in the list of connections shown by Stepping Stones, indicating that finding a valid connection was easier with Stepping Stones. As we mentioned before, this is not due to one particular query in our experiments, or to the document set returned in Stepping Stones being smaller than with the ranked list.

What is a valid answer to "X related to Y?" depends on what is a valid connection for the user: Stepping Stones and Pathways suggests valid connections the user was not aware of. We expected users to restrict their set of connections to a very small group, finding a certain agreement on what connections were important. Our users in practice reported as valid a variety of connections both built by themselves and by the Stepping Stones tool. If we consider the Standard Query and Stepping Stones results as separate, disjoint sets, and all sets of connections reported by users for the same search method would all be subsets of a single set of answers, the users would have reported 37 different connections (since not all users returned the same number of connections). Instead they reported a total of 61 different connections. While the popularity of connections is not uniform, the difference in the variety of answers is not just chance. It reflects the fact that what constitutes a valid connection is subjective and is affected by the knowledge background and expectations of the user.

6.2 Future work

Use the syllabi and citation database to build a system that automatically suggests and expands bibliography: The combination of syllabi and citations can be used as a method to automatically suggest a bibliography. The user would give the system an area and topic within the area, such as Operating Systems as subject, and Distributed Systems as the area. A web crawler would find (using access to an existing web search engine, like the Google API) and identify syllabi that are based on papers. Then the system would try to find papers related to the area given by the user within the syllabi. The system can do this with high confidence because syllabi have the areas clearly delimited. If the area is found, then the system would use association rules [Agrawal93] to find tuples of citations in papers. These tuples should contain one or more papers that are included in the area the user is interested in within the syllabi. Finally the system would present the association rules to the user, in the form of "if you are interested in these papers, you should read these other papers too". The critical points for this system to succeed are the availability of a citation database, and the precision of the crawler at identifying relevant syllabi.

Separate the interface into casual and advanced user interfaces: During the user tests while designing the interface, and in our pilot tests, we found that users had trouble adapting to the idea of retrieving connections instead of documents. In almost all cases, the model of retrieval consisting of a query followed by a ranked list of results was for them almost second nature. Changing from obtaining a list to obtaining a graph as the result of a query took effort on their part, as can be seen in the user comments in the previous chapter. We partially addressed this problem by adding the list of all connections as part of the query result. Adding the list of connections, instead of just having the connection graph, had a very positive effect on users during our pilot study, since they found or felt it easier to transfer their knowledge of how to interpret a retrieval result that looks like a ranked list.

At the same time we realized that the classic retrieval model was too entrenched in the user's minds to take advantage or develop a plan using more sophisticated tools. We

originally had two extra options in the node menus: “Merge these topics” and “Split this topic into more specific topics”. These options corresponded to joining clusters into a super-cluster, or split a cluster into sub-clusters, with the purpose of refining and modifying the connections between topics. We found in our pilot tests that users did not understand when to use it, or simply ignored them completely. After interviewing them afterwards they recognized these options as potentially useful, but they did not need them as part of their retrieval plan; their retrieval plan was derived from the retrieval plan in a classic search engine. This is the reason we removed these options from the version of Stepping Stones used in our experiments. We firmly believe that it is possible to split the interface into two modes: casual and advanced, and prove that the advanced mode allows the user to perform a more sophisticated retrieval session than with a classic search engine for the particular task of finding connections between topics. The casual user interface would look like the current one, with the addition of a small graph overview, where the whole graph is shown in a small window, with a rectangle showing the area current visible in the main window. The advanced user interface would also include:

- The options to merge and split nodes.
- The capability of letting the user select the minimum similarity between documents to be two documents as related, in practice letting the user control the tradeoff in the number of connections reported versus the specificity of those connections. This could be controlled with a slider, with a “more connections” value on the side, and a “more specific connections” value on the other end.
- Let the user specify topics to add as stepping stones instead of the system giving them all. The user would be able to hypothesize on topics that connect the sub-queries, and the system would try to find a way to relate all those topics. From an implementation point of view, it is easier to find a connection from a topic given by the user than to try to find good topics to connect the sub-queries without aid.

Examination of Time-on-task: As mentioned in the previous chapter, the fact that participants performed the retrieval sessions without the experimenter being present had a serious effect on the accuracy of measuring tasks times. From the data gathered, it seemed that users were slower when using Stepping Stones than when using Standard Query. An aspect that merits further research is whether this is an effect of lack of familiarity with Stepping Stones, or an effect of the user interface design. A more accurate timing could be obtained by repeating the experiments in an environment where the researcher is present to control distractions, like a classroom setting. Nevertheless, that would not be enough to see if task time is a function of training, or if there is a defect in the design of the user interface that makes the user perform slower – perhaps the combination of the connection graph and connection list makes the interface intrinsically more complicated. The only way to accurately recognize the cause of why users are slower with Stepping Stones would be to have a subset of participants to use Stepping Stones repeatedly and see if timing improves significantly, while at the same time observing their actions to find critical incidents that delay their performance.

Calibration of Pathway scores: While we developed a way to calculate a ranking score for each pathway, during our pilot studies we found that users liked the ordering of

pathways by the number of documents involved (all pathways involving 1 document, then 2, and so on) more than ordering them by score. The reason is that it is easier to understand an order based on document count than an order based on a score, where the criterion ordering the pathways on screen is not apparent.

This fact, and that we needed to limit the number of connections reported by the user in order to allow for multiple retrieval sessions by the same user in a reasonable time frame, did not allow us to experimentally explore the quality of pathway scores. These scores are derived from a set of assumptions that are the same used to create the pathways, and since the pathways yielded good scores in our experiments, we know the scoring is not completely off-the-mark. However, we did not have the opportunity of comparing user scores to system scores to see if system scores match the user ranking of pathways. This experiment would proceed with the user performing few retrieval sessions only with Stepping Stones and Pathways. Participants would give a score to a long list of pathways in a 10-points-or-more scale, so as to allow enough differentiation between the scores. The ideal result would be that the ranking order given by the system scores matches the order given by the user. If connection A is given a system score higher than connection B, then the user score for A should be greater than or equal to the user score for connection B.

Another aspect we think merits further research is to assign subjective labels to values of the system score. Reporting the system scores to the user would be of little utility, since users have no context to interpret them. A more sensible approach would be to assign a label to at least some values in the scale used to obtain the user score for connections, and then find the threshold in system scores at which users assign them labels like “very good”, “good”, and “weak”. Once the system score is calibrated to the subjective scale, we could report scores to users using the subjective scale.

Comparison of Arrowsmith vs. Stepping Stones using PubMed: As we have described many times, the closest system in terms of functionality to Stepping Stones is Arrowsmith. Given that both systems use different techniques and approaches, we think it would be of great interest to compare the results of both systems. All results published about Arrowsmith, and even its current public version, only work on PubMed. Therefore the only way to compare results is to use Stepping Stones on PubMed too. The biggest obstacle for a comparison is to find a suitable pool of subjects with the knowledge in medicine required to understand and qualify the results. Relationships among topics found by Arrowsmith are going to be more generic than those found by Stepping Stones. The questions to answer are: How much more generic? Are the relationships found by Stepping Stones different yet valid to explore other connections in medicine, beside those between symptom and treatment?

Improve cluster labeling by only clustering the critical paragraphs of each document: As we mentioned at the end of Chapter 4, cluster labeling worked, but could be improved. The two main factors degrading the quality of cluster labeling with STC are: noise phrases, and extent of document. We discussed noise phrases before; they are phrases with no semantic meaning from a clustering point of view which appear

repeatedly as a result of language usage (papers are usually written in "technical English", a subset of the English, and these phrases appear often).

Document extent affects labeling because Suffix-Tree Clustering (STC) is much better at labeling sentences than whole documents. Sentences are usually more coherent than whole documents in terms of topics, addressing only one or two topics at a time. Finding sentences that address the same topics the document addresses as a whole, and using those sentences to find labels for the clusters, would lead to a more accurate labeling. Zamir's STC paper [Zamir98] does exactly that, by using the text snippets returned by the search engines to cluster the web pages. There is an extensive corpus of research in the Information Retrieval community on finding the most content-bearing paragraphs in a document, under the name of "Text Summarization", like [Salton96], [Hearst97b], [Goldstein99], [Lin2000], [Mani2001] [Moens2001] and [Teufel2002] to name a few representative papers. [Mani99] and [Radev2002] are good surveys of the modern state of text summarization. We strongly believe that intelligently summarizing the paper instead of doing it by using the first 4 kilobytes of text, as we did, would lead to significantly more accurate labels for clusters.

Study the effect of noise on finding pathways over bigger collections: The collection of papers we used for the experiments is a small one. While its size did not guarantee a successful outcome of our experiments, we certainly admit that a small collection of papers on a specific area of computer science has a higher chance of allowing the system to find related papers than a wider collection, with more topics and more documents that can be misclassified by the system. The problem has always been, as we mentioned in Chapter 3, that we have specific requirements for a collection to be useful as a source for experiments with Stepping Stones. Thus, we took the effort of building our own. There is a new text collection called INEX [Fuhr2003] [Fuhr2004], made available to the public (with certain restrictions) in 2003, that can be a good candidate for Stepping Stones and Pathways. INEX is a collection of over twelve thousand articles from 18 IEEE magazines and transactions from 1995 to 2002. All articles are marked-up in XML, denoting not only document text structure (title, body) but also references. Although we do not know of any study about the specificity of content in INEX, given the limited scope of IEEE, it is a likely candidate worth exploring. In particular it is important to see if it is possible to build a set of queries in INEX about related topics, that can take the place of the ones we built from relationships inside and between syllabi.

Frequency of occurrences of queries that are candidates for better answers with Stepping Stones: We centered our study in scientific literature collections, because that is where we see our approach as having the biggest payoff, and because of their rich structure of topics, citations, and authors. A question that we could not answer for lack of resources is the frequency of queries that are likely to get a better answer by splitting the query and using Stepping Stones to retrieve connections between the split sub-queries. One possible way to perform this study would be to analyze the query log of a public scientific literature collection, such as CITIDEL. Under the assumption that a succession of queries from the same user in a short amount of time using variations of similar topics indicates poor relevance results, then it is possible to identify a set of queries that are

candidates for more detailed analysis as candidates for splitting. By comparing the number of semantically similar words in two successive queries, the researcher can identify variations of a query. Netscape's Open Directory RDF description can be used as a taxonomy to identify variations of related queries with different words referring to the same concepts. Experiences in the past [Labrou99] indicate that this is a feasible approach. Once the set of candidate queries is determined, we will evaluate how recall can be improved by query splitting. Query splitting has been studied in the past [Borodin68] [Kane-Esrig91] but not in much detail, mostly because of the high computational cost of optimal unrestricted splitting in an interactive search engine. We argue that nowadays it is possible to perform query splitting because of three factors: *a*) User studies [Moricz98] [Spink2001] indicate that query length in real-life queries is limited; *b*) even in the worst case, query words are not issued in a completely random order by the user, and so the combinatorial problem is reduced to a displacement of all query words up to N places, where N < length of query and N is an experimental parameter; and *c*) the cluster labeling method we use produces valid phrases, and we can use these phrases as topics to match against query parts.

Appendix A. Example of instructions sent to users that participated in the evaluation of Stepping Stones and Pathways

What follows is the text of the instructions sent to one of the users. All users received the same instructions, except for the URLs to use to perform the tasks, and the topics involved.

Instructions for Participant:

You may remember that I asked for participants for an experiment, and you volunteered to participate. Thank you again for agreeing to participate in this paid experiment. The whole experiment should take from 1 hour to an hour 15 minutes. If you are going to participate, or you are not, please send me an email to [`email address`] telling me so.

Any questions that you have about the experiment, do not hesitate to write me to [`email address`]. It would be convenient for you to print these instructions before starting. We would like the results back by [`deadline`]. If you need more time, please tell me so.

Here's what we want you to do:

What we want you to do is to find how pairs of topics in the field of Operating Systems are related. Think of a topic as a concept, like "file system", or "virtual memory". Topics are related by one or more documents, in this case from a group of research papers about Operating Systems. For example, "Java" and "Virtual Machine" can be related through "garbage collection", because one paper would talk about garbage collection in Java, and another about the implementation of garbage collection in virtual machines. We are going to give you 2 pairs of concepts, and we are going to ask you to find how they are related using two different methods:

Method A) a search tool that returns a list of documents, ordered from an estimation of most relevant to less relevant. Let's call it the "Everyday Search Tool". It works like many other search tools you have used on the web. Here you will type the words for the two topics, and then read the documents (no more than 25, not all relevant) trying to find how their content relate the two topics. For example, if the topics are "Java" and "Operating Systems", you will type the words "Java Operating Systems" in the text field at the top. We have a quick description of this tool here: [`url of tutorial for Standard Query`].
You can get used to it here: [`url of training version of Standard Query`].

Method B) A different tool we designed, that instead of returning a list of documents, returns connections between the topics. This tool, that we call "Stepping Stones and Pathways" is different from what you are used to, so we have written a not-too-long

tutorial for it. Take a look at [`url of tutorial for Stepping Stones`]. Please read it, and then play with this tool for a while here: [`url of training version of Stepping Stones`]. Try with pairs of topics like "web server" and "java", or "file system" and "java", to get used to this tool.

After you get acquainted to the tools (we estimate a minimum of 20-30 minutes), then we are going to ask you to find the connections between these 2 pairs of topics (definitions at the end of the email):

"*web server*" and "*fault tolerance*"
"*extensible operating system*" and "*virtual machine*"

For each pair, we are ask you to find up to 3 connections between every pair of topics, telling how those topics are related to each other. ***Every time*** you find a good relation between the topics, you will click on the button labeled "I found a connection" on the top right. A short form with 6 simple questions will pop-up so you can tell us about the connection between the topics. Remember to click on the "I found a connection" button every time you find a good connection, up to 3 of them. If you can't find 3, give us as many as you can.

IMPORTANT NOTES:
* Every time, IGNORE what you have found in the previous step. We want what you have found using each tool, not what you have learned.
* WE WANT RELATIONSHIPS FOUND IN THE GROUP OF PAPERS, not relationships that you knew about beforehand.
* In the case of Stepping Stones and Pathways, please INCLUDE AT LEAST ONE CONNECTION WITH multiple documents, if available and possible. The more, the better.

Your tasks:

1) We ask you to find using "Stepping Stones and Pathways" how the topics "`extensible operating system`" and "`virtual machine`" are related, using the following URL:
[`url of Stepping Stones tool that includes user and query pair ids`]

2) Finally, We ask you to find with the "Everyday Search Tool" how the topics "`extensible operating system`" and "`virtual machine`" are related, using the following URL:
[`url of Standard Query tool that includes user and query pair ids`]

3) We ask you to find with the "Stepping Stones and Pathways" tool how the topics "web server" and "fault tolerance" are related, using the following URL:
[`url of Stepping Stones tool that includes user and query pair ids`]

4) We ask you to find with the “Everyday Search Tool” how the topics “web server” and “fault tolerance” are related, using the following URL:
[`url of Standard Query tool that includes user and query pair ids`]

Once you are done please send me an email at [`email address`] titled “Paid Experiment Done” telling me:

* In a scale from 1 to 5, your confidence in your knowledge of Operating Systems (1=very basic, 2=limited 3=average, 4=good 5=expert)
* A sentence telling that you participated in this experiment, with your complete name.

Thanks again!

Fernando

DEFINITIONS:
If you don’t know what any of the topics are about, you can find the definitions of the topics here:
[`url of a web page with definitions of all topics in the query pairs`]

Bibliography

[Anick90] Anick, P., Brennan, J., Flynn, R. et al. *A Direct Manipulation Interface for Boolean Information Retrieval via Natural Language Query*. In Proceedings of SIGIR'90, pp. 135-150, 1990.

[Agrawal93] Agrawal, R., Imielinski, T., Swami, A. *Mining Associations between Sets of Items in Massive Databases*. Proceedings of SIGMOD'93, pp. 207-216, 1993.

[Baeza92] Baeza-Yates, R. *Information Retrieval: Data Structures and Algorithms*. Pages 13-27. Prentice Hall, 1992.

[Baker90] Baker, M., Outherhost, J. *Availability in the Sprite distributed file system*. Proceedings of the 4th ACM SIGOPS European Workshop, pp. 1-4. ACM Press, 1990.

[Barr2002] Barr, R. *Spyce: Python Server Pages*. Available at http://spyce.sourceforge.net/ .

[Bartell94] Bartell, B. T., Cottrell, G. W., & Belew, R. K. *Automatic combination of multiple ranked retrieval systems*. Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval., pp. 173-181, 1994.

[Belkin95] Belkin, N, Kantor, P., Fox, E., and Shaw J. *Combining the Evidence of Multiple Query Representations for Information Retrieval*. Information Processing and Management, Vol. 31, Issue 3, pp. 431-448, 1995.

[Benford95] Benford, S., Snowdown, D., Greenhalgh, C. Ingram, R. et al. *VR-VIBE: A Virtual Environment for Co-operative Information Retrieval*. Proceedings of Eurographics*'95*, pp. 349-360, 1995.

[Botafogo91] Botafogo, R., Shneiderman, B. *Identifying aggregates in hypertext structures*. In Proceedings of Hypertext'91, pp. 63-74, 1991

[Bourdoncle97] Bourdoncle, F. 1997 "*LiveTopics: Recherche Visuelle d'Information sur l'Internet*." (LiveTopics: visual search for information on the Internet) Dossiers de l'Audiovisuel, La Documentation Francaise No. 74 (July-Aug 1997), pp. 36-38, 1997.

[Brin98] Brin, S., Page L. *The Anatomy of a Large-Scale Hypertextual Web Search Engine*. Proceedings of the 7th International World Wide Web Conference. 1998. Online proceedings, publication available at http://decweb.ethz.ch/WWW7/1921/com1921.htm

[Buckley2000] Buckley,C. and Walz, J. *The TREC-8 Query Track*. In Proceedings of the 8th Text Retrieval Conference (TREC) – NIST Special Publication 500-246, 2000. Online proceedings, publication available at http://trec.nist.gov/pubs/trec8/papers/qtrack.pdf

[Buntine94] Buntine, W. *Operations for Learning with Graphical Models*. Journal of Artificial Intelligence Research, 1994.

[Casanova91] Casanova, M., Tucherman, L., Lima, M., et al. *The Nested Context Model for Hyperdocuments*. In Proceedings of Hypertext'91, pp. 193-201, 1991.

[Chalmers92] Chalmers, M., Chitson, P. *Bead: Explorations in Information Visualization*. Proceedings of SIGIR'92, pp. 330-337, 1992.

[Chatfield80] Chatfield, C., Collins, A. *Introduction to Multivariate Analysis*. Chapman & Hall, London, 1980.

[Cooper92] Cooper, W., Gey, F., Dabney, D. *Probabilistic Retrieval Based on Staged Logistic Regression*. In Proceedings of SIGIR'92, pp. 198-210, 1992.

[Cutting92] Cutting, D., Karger, D., Pedersen, J. and Tukey, J. *Scatter/Gather: A Cluster-based Approach to Browsing Large Document Collections*. In Proceedings of SIGIR'92, pp. 318-329. ACM Press, 1992.

[Dasigi98] Dasigi, V. *Information Fusion Experiments for Text Classification*. In Proceedings of the 1998 IEEE Information Technology Conference, pp. 23-26, 1998.

[Das-Neves97] Das-Neves, F. *The Aleph: a tool to Spatially Represent user Knowledge about the WWW docuverse*. In Proceedings of Hypertext'97, pp. 197-207, 1997.

[Das-Neves2000] Das-Neves, F, Fox. E. *A Study of User Behavior in an Immersive Virtual Environment for Digital Libraries*. Proceedings of ACM Digital Libraries 2000, pp. 103-111, 2000.

[Dean99] Dean, J., Henzinger, M. R. *Finding related pages in the World Wide Web*. In Proceedings of the Eighth International World Wide Web Conference, 1999.

[Dumais93] Dumais, S. Latent semantic indexing (LSI) and TREC 2. In Proceedings of the Second Text Retrieval Conference (TREC-2), pp. 105-115, 1993.

[Dunn73] Dunn, J. *A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters*. Journal of Cybernetics, Issue 3, pp. 32-57, 1973.

[Fajtlowicz88] Fajtlowicz, S. On Conjectures of Graffiti. Discrete Mathematics, Issue 72, pp. 113-118, 1988.

[Fox83] Fox, E. *Extending the Boolean and Vector Space Models of Information Retrieval with P-Norm Queries and Multiple Concept Types*. Cornell University Department of Computer Science dissertation, 1983.

[Fox85] Fox, E. *Composite Document Extended Retrieval: an Overview*. In Proceedings of SIGIR 95, pp. 42-53, 1995.

[Fox88] Fox, E., and Koll, M. *Practical Enhanced Boolean Retrieval: Experiences with the SMART and SIRE Systems*. Information Processing and Management, Vol. 24, Issue 3, pp. 257-267, 1988.

[Fox92] Fox, E., Koushik, M., Shaw, J., Modlin, R., Rao, D. *Combining Evidence from Multiple Searches***.** In Proceedings of the First Text Retrieval Conference (TREC-1), pp. 319-338, 1992.

[Fox93b] Fox, E. A. (1983). *Characterization of Two New Experimental Collections in Computer and Information Science containing Textual and Bibliographic Concepts*. Technical Report 83-561, Cornell University, Ithaca, NY, 1993.

[Fox94] Fox, E., Shaw, J. *Combination of Multiple Searches*. Proceedings of the 2nd Text Retrieval Conference (TREC-2), pp. 243-252, 1994.

[Fox2003] Fox, E. Case Studies in the US National Science Digital Library: DL-in-a-Box, CITIDEL, and OCKHAM. Proceedings of ICADL'2003, pp. 17-25, 2003.

[Fowler91] Fowler, R., Fowler, W., Wilsion, B. *Integrating Query Thesaurus, and Documents through a Common Visual Representation*. In Proceedings of SIGIR'91, pp. 142-151, 1991.

[Frantz2002] Frantz, J., Tapus, C., Smith,J. and Hickey, J. *MojaveFS: A Transactional Distributed File System (whitepaper).*
Available at http://mojave.caltech.edu/papers/submitted/icdcs03.pdf .

[Fung95] Fung, R., and Favero, B. *Applying Bayesian Networks to Information Retrieval*. Communications of the ACM, Vol. 38, No 3, pp. 42-48, 1995.

[Fuhr2003] Fuhr, N., Gövert N., Malik, S., et al. *INEX Working Group*. Available at http://www.is.informati.uni-duisburg.de/projects/inex/

[Fuhr2004] Fuhr, N., Lamas. M. (Eds.). *Proceedings of the INEX 2003 Workshop*. SIGIR Forum, Vol. 38, Issue 1, pp. 42-47, 2004.

[Gansner99] Gansner, E., North, S. *An Open Graph Visualization System and its Applications to Software Engineering*. Software Practice and Experience*,* Issue 1, pp. 1–5, 1999.

[Garfield65] Garfield, E. *Essays of an Information Scientist*, Vol. 1, pp.84-90, 1965.

[Garfield72] Garfield E. *Citation Analysis as a Tool in Journal Evaluation*, Science, Issue 178, pp. 471- 479. 1972.

[Garfield88] Garfield, E. *Essays of an Information Scientist*. Volume 11, pp. 369-374, 1988.

[Gauch99] Gauch,S., Wang,J., Rachakonda, S. *A Corpus Analysis Approach for Automatic Query Expansion and its Extension to Multiple Databases*. ACM TOIS Vol. 17, No. 3, pp. 250-269, 1999.

[Gey94] Gey, F. *Inferring Probability of Relevance using the Method of Logistic Regression*. In Proceedings of SIGIR'94, pp. 222-231, 1994.

[Giles98] Giles, L., Bollaker, K., and Lawrence, S. *Citeseer: An Automatic Citation Indexing System*. Proceedings of Digital Libraries'98, pp. 89-98. ACM Press, 1998.

[Ginsparg96] Ginsparg, P., *Los Alamos XXX*. APS News. APS Online (insert), November 8, 1996.

[Goldstein99] Goldstein, J., Kantrowitz, M., Mittal, V. *Summarizing Text Documents: Sentence Selection and Evaluation Metrics*. Proceedings of SIGIR'99, pp. 121-128, 1999.

[Gordon1998] Gordon, M., Dumais, S. *Using Latent Semantic Indexing for Literature Based Discovery*. Journal of the American Society for Information Science and Technology, Vol. 49, Issue 8, pp. 674-685, 1998.

[Guha99] Guha S., Rastogi R. and Shim K. *ROCK: A Robust Clustering Algorithm for Categorical Attributes*. In 15th ICDE International Conference on Data Engineering, pp. 512-521, 1999.

[Halasz88] HALASZ, F. *Reflections on Notecards: Seven Issues for the Next Generation of Hypermedia Systems*. Communications of the ACM Vol. 31, No. 7, pp. 836–852.

[Hara91] Hara, Y., Keller, A., Wiederhold G. *Implementing hypertext database relationships through aggregations and exceptions*. In Proceedings of Hypertext'91, pp. 75-90, 1991.

[Hearst95] Hearst, M., Karger D., and . Pedersen J. *Scatter/Gather as a Tool for the Navigation of Retrieval Results.* Proceedings of the 1995 AAAI Fall Symposium on Knowledge Navigation, pp. 65-71, 1995.

[Hearst96] M. A. Hearst and J. 0. Pedersen. *Reexamining the cluster hypothesis: Scatter/Gather on retrieval results*. In Proceedings of ACM SIGIR'96, pp. 76-84, Aug. 1996.

[Hearst97] Hearst, M., and Karadi, C. *Cat-a-Cone: An Interactive Interface for Specifying Searches and Viewing Retrieval*. In Proceedings of SIGIR'97, pp. 246-255, 1997.

[Hearst97b] Hearst, M. *TextTiling: Segmenting Text into Multi-paragraph, Subtopic Passages*. Computational Linguistics, Vol. 3, Issue 1, pp. 33-64, 1997.

[Heckerman96] Heckerman, D. *A Tutorial on Learning with Bayesian Networks*. Microsoft Technical Report MSR-TR-95-06, 1995. Available at http://research.microsoft.com/research/pubs/view.aspx?msr_tr_id=MSR-TR-95-06 .

[Hemmje94] Hemmje, M., Kunkel, C., Willet, A. *LyberWorld - A Visualization User Interface Supporting Full Text Retrieval*. In Proceedings of SIGIR'94, pp. 249-259, 1994.

[Hetzler98] Hetzler, B., Harris, M., Havre, S., Whitney, P. *Visualizing the Full Spectrum of Document Relationships*. In: Structures and Relations in Knowledge Organization. Proc. 5th Int. ISKO Conf. Wurzburg: ERGON Verlag, 1998. pp. 168-175, 1998.

[Jardine71] Jardine, N, and Van Risjbergen, J.C. *The Use of Hierarchic Clustering in Information Retrieval*. Information. Storage and Retrieval., Vol. 7, pages 217-240, 1971.

[Jensen96] Jensen, F. *An Introduction to Bayesian Networks*. Springer-Verlag, 1996.

[Jensen2001] Jensen, F. *Bayesian Networks and Decision Graphs*. Springer-Verlag, 2001.

[Jing94] Jing Y., Croft, W. B. *An Association Thesaurus for Information Retrieval*. In Proceedings of the Intelligent Multimedia Information Retrieval Systems (RIAO '94, New York, NY), pp. 146–160, 1994.

[Jones98] Jones, S. *Graphical Query Clustering and Dynamic Result Previews for a Digital Library*. In Proceedings of UIST'98, pp. 143-151, 1998.

[Karypis98] Karypis, G., Kumar, V. *Multilevel k-way Partitioning Scheme for Irregular Graphs*. Journal of Parallel and Distributed Computing, vol. 48, pp. 86-129, 1998.

[Karpis99] Karpis, G, Han, E., Kumar, V. Chameleon: *Hierarchical Clustering Using Dynamic Modeling*. Computer, Vol. 32, Issue 8, pp. 68-75, 1999.

[Kessler63] Kessler, M.M. *Bibliographic Coupling Between Scientific Papers*. American Documentation, Vol. 14, pp. 10-25, 1963.

[King67] King, B. *Step-wise Clustering Procedures*. Journal of the American Statistical Association, Issue 69, pp. 86-101.

[Kleinberg98] Gibson, D, Kleinberg, J., Raghavan P., *Clustering Categorical Data: An Approach Based on Dynamical Systems*. Proceedings of the 24th International Conference on Very Large Data Bases, pp. 311 - 322, 1998.

[Kleinberg99] Kleinberg J. *Authoritative Sources in a Hyperlinked Environment*. Journal of the ACM, Issue 48, pp. 604-632, 1999.

[Kohonen95] Kohonen T. *Self-Organizing Maps*, Volume 30 of Springer Series in Information Sciences. Springer-Verlag, Berlin, 1995.

[Krowne2001] Krowne, A. The ESSEX Search Engine. Available at http://br.endernet.org/~akrowne/elaine/essex/ (Note: The version Essex used in this dissertation differs considerably from the one available in the website).

[Lagus96] Lagus, K. T. Honkela, T., Kaski, S., & Kohonen, T. *Self-Organizing Maps of Document Collections: A New Approach to Interactive Exploration*. In Proceedings of the International Conference on Knowledge Discovery and Data Mining, 1996.

[Lance67] Lance, G, and Williams, W. *Mixed-Data Classificatory Programs I - Agglomerative Systems*. Australian Computer Journal, Vol. 1, Issue 1, pp. 15-20, 1967.

[Lance67b] Lance, G. N. and Williams, W. T. A general theory of classificatory sorting strategies: 1. hierarchical systems. *Computer Journal, 9:373-380,* 1967.

[Lamping95] Lamping, J., Rao, R., Pirolli, P. *A focus+context technique based on hyperbolic geometry for visualizing large hierarchies*. In Proceedings of CHI'95, pp. 401-408, 1995.

[Larsen2004] Larsen, B. *References and Citations in Automatic Indexing and Retrieval Systems – experiments in the Boomerang Effect. Doctoral Dissertation*, Department of Information Studies, Royal School of Library and Information Science, 2004. Available at http://www.db.dk/blar/dissertation.

[Lawrence99] Lawerence, S, Giles, L., Bollaker, K. *Autonomous Citation Matching. Proceedings of the Third Conference on Autonomous Agents*, pp. 392-393, 1999.

[Lee97] Lee, J. H.. *Analyses of Multiple Evidence Combination*. In Proceedings of SIGIR'97, pp. 267-276, 1997.

[Leuski2001] Leuski, A. *Evaluating Document Clustering for Interactive Information Retrieval.* Proceedings of the 10th International ACM Conference on Information and Knowledge Management. New York, ACM. pp. 33-40, 2001.

[Ley95] Ley, M. DB&LP: *A WWW Bibliography on Databases and Logic Programming*. Compulog Newsletter, 1995.

[Ley2002] Ley, M**.** The DBLP Computer Science Bibliography: Evolution, Research Issues, Perspectives. Proceedings of the 9th International Symposium on String Processing and Information Retrieval, pp. 1-10, 2002.

[Lewis96] D. D. Lewis. *Reuters-21578 text categorization test collection distribution* 1.0. http://www.research.att.com/#lewis, 1996.

[Lin2000] Lin, C., Hovy, E. *The Automated Acquisition of Topic Signatures for Text Summarization*. Proceedings of the 17th Conference on Computational Linguistics. Vol. 1, pp. 495-501, 2000.

[Lindsay99] Kindsay, R., Gordon, M. *Literature-Based Discovery by Lexical Statistics*. Journal of the American Society for Information Science and Technology, Vol. 50, Issue 7, pp. 574-587, 1999.

[Mani99] Mani, I., Maybury, M. *Advances in Text Summarization*. MIT Press, 1999.

[Mani2001] Mani, I. *Text Summarization and Question Answering: Recent Developments in Text Summarization*. Proceedings of the 10th International Conference on Information and Knowledge Management, pp. 529-531, 2001.

[Cleverdon67] Cleverdon, C. The Cranfield Tests on Index Language Devices. Proceedings of ASLIB, Vol 19, Issue 6, pp. 173-194, 1967.

[McCreight76]. McCreight, E. *A Space-Economical Suffix Tree Construction Algorithm*. Journal of the ACM, Issue 23, pp. 262-272, 1976.

[Miller2000] Miller, D., Leek, T., Schwarts, R. *A Hidden Markov Model Information retrieval system*. In Proceedings of SIGIR 2000.

[Moens2001] Moens, F., De Busser, R. *Generic Topic Segmentation of Document Texts*. Proceedings of SIGIR'2001, pp. 418-419, 2001.

[Morickz98] Morickz M., Marais J., Hensinger M., Silverstein C. *Analysis of a Very Large Alta Vista Query Log*. Technical Report 1998-014 COMPAQ Systems Research, 1998.

[Mukherjea95] Mukherjea, S., Foley, J.D., and Hudson, S. *Visualizing Complex Hypermedia Networks through Multiple Hierarchical Views*. Proceedings of CHI'95, pp. 331–337, 1995.

[Murtagh84] Murtagh, F. *A Survey of Recent Advances in Hierarchical Clustering Algorithms which use Cluster Centers.* Computer Journal, Issue 26, pp. 354-359, 1984.

[Nagy68] Nagy, G. *State of the Art in Pattern Recognition*. Proceedings of IEEE. Issue 56, pp. 836-862, 1968.

[Nelson96] Nelson, M. *Fast String Searching With Suffix Trees*. Dr Dobb's Journal, August 1996 issue.

[Noik93] Noik, E. *Exploring large hyperdocuments: Fisheye Views of Nested Networks. In Proceedings of Hypertext'93,* pp. 192-205, 1993.

[Nowell93] Nowell, L., Hix, D. *Visualizing search results: User Interface Development for the Project Envision Database of Computer Science Literature.* In Advances in Human Factors/Ergonomics, Proceedings of HCI International '93, 5th International Conference on Human Computer Interaction, vol. 19B, Human-Computer Interaction: Software and Hardware Interfaces: pp. 56-61, Elsevier, 1993.

[Nowell96] Nowell, L., Hix, D., France, R., Heath, L., Fox, E. A. *Visualizing Search Results: Some Alternatives to Query-Document Similarity*. In Proceedings ACM SIGIR '96. pp. 67-75, 1996.

[Pearl88] Pearl, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA, 1988.

[Pirolli96] Pirolli, P., Schank, P., Hearst, M., Diehl, C. *Scatter/Gather Browsing Communicates the Topic Structure of a Very Large Text Collection*. Proceedings of SIGCHI'96, pp. 213-220, 1996.

[Porter80] Porter, M. An Algorithm for Suffix Stripping, Program, Vol. 14, Issue 3, pp. 130-137, 1980.

[Radev2002] Radev, D., Hovy, E., McKeown, K. (Eds.). *Introduction to the Special Issue on Summarization*. Computational Linguistics, Vol. 28, Issue 4, pp. 399-408, 2002.

[Ramakrishnan2004] Ramakrishan, N., Kumar, D., Mishra, B., et al. *Turning CARTwheels: An Alternating Algorithm for Mining Redescriptions*. Proceedings of KDD 2004, pp. 266-275, 2004.

[Rocchio71] Rocchio, J. *Relevance Feedback in Information Retrieval.* In The Smart Retrieval System—Experiments in Automatic Document Processing, G. Salton, Ed. Prentice-Hall, pp. 313–323, 1971.

[Rorvig99] Rorvig, M. E. *A Visual Exploration of the Orderliness of TREC Relevance Judgments*. JASIS, Vol. 50, Issue 8, pp. 652-660, 1999.

[Salton75] Salton, G., Wong, A., and Yang C. *A Vector Space Model for Information Retrieval*. Communications of the ACM, Vol. 18, Issue 11, pp. 613-620, 1975.

[Salton96] Salton, G., Singhal, A., Buckley, C., Mitra. M. *Automatic Text Decomposition using Text Segments and Text Themes*. Proceedings of Hypertext'96, pp. 53-65, 1996.

[Sebrechts99] Sebrechts, M., Cugini, J., Laskowski, S., et al. *Visualization of search results: a comparative evaluation of text, 2D, and 3D interfaces*. In Proceedings of SIGIR'99, pp. 3-10, 1999.

[Sengupta92] Sengupta, I. N. (1992). *Bibliometrics, Informetrics, Scientometrics and Librametrics: An Overview*. Libri, Vol. 42, Issue 2, pp. 75-98, 1992.

[Shaw97] Shaw, W., Burgin, R., Howell, P. *Performance Standards and Evaluations in IR test collections: Cluster-based Retrieval Models*. Information Processing and Management, Vol. 33, Issue 1, pp. 1-14, 1997.

[Shneiderman98] Shneiderman, B. *Designing User Interfaces*, p. 523. Addison Wesley, 1998.

[Simkin2003] Simkin, M., Roychowdhury, V. *Read before you Cite!* Complex Systems, Issue 14, pp. 269-274, 2003.

[Small73] Small, H. *Co-citation in the Scientific Literature: A New Measure of the Relationship between Two Documents*, Journal of the American Society for Information Science and Technology, Vol. 24, pp. 265-269, 1973.

[Sneath73] Sneath, P.H, and Sokal R. *Numerical Taxonomy*: The Principles and Practice of Numerical Classification. Freeman, UK, 1973.

[Spink2001] Spink, A., Wolfram, D. Jansen, M, and Saracevik T. *Searching the Web: the Public and their Queries*. Journal of the American Society for Information Science and Technology, Vol. 52, Issue 3, pp. 226-234, 2001.

[Spasser97] Spasser, M. *The Enacted Fate of Undiscovered Public Knowledge*. Journal of the American Society for Information Science and Technology, Vol. 48, Issue 8, pp. 707-717, 1997.

[Swanson86] Swanson, D. *Undiscovered Public Knowledge*. Library Quarterly, Vol.56, No.2, pp. 103-118, 1986.

[Swanson87] Swanson, D. *Two Medical Literatures that are logically but not Bibliographically Connected*. Journal of the American Society for Information Science and Technology, Vol. 38, Issue 4, pp. 228-233, 1987.

[Swanson89] Swanson, D. *A Second Example of Mutually Isolated Medical Literatures Related by Implicit, Unnoticed Connections*. Journal of the American Society for Information Science and Technology, Vol. 40, Issue 6, pp. 432-435, 1989.

[Swanson91] Swanson, D. *Complementary Structures in Disjoint Scientific Literatures*. In Proceedings of SIGIR'91, pp. 280-289, ACM Press, 1991.

[Swanson2001] Swanson, R., Smalheiser, N, Bookstein, A. *Information Discovery from Complementary Literatures: Categorizing Viruses as Potential Weapons*. Journal of the American Society for Information Science and Technology, Vol. 52, Issue 10, pp. 797-812, 2001.

[Teufel2002] Teufel, S., Moens, M. *Summarizing Scientific Articles: Experiments with Relevance and Rhetorical Status*. Computational Linguistics, Vol. 28, Issue 4, pp. 409-445, 2002.

[Thorne77] Thorne, F. *The Citation Index: Another Case of Spurious Validity*. Journal of Clinical Psychology, Vol. 33, issue 3, pp. 1157-1161, 1977.

[Tunkelang97] Tunkelang, D., Byrd, R., and Cooper, J. *Lexical Navigation: Using Incremental Graph Drawing for Query Refinement.* Proceedings of GD'97, Graph Drawing, 5th International Symposium, pp. 316-321, 1997.

[Turtle90] Turtle, H., Croft, W. *Inference networks for document retrieval*. In Proceedings of SIGIR'90, pp. 1-24, 1990.

[Turtle91] Turtle H., & Croft, W. B. (1991). *Evaluation of an Inference Network-based Retrieval Model.* ACM Transactions on Information Systems, 9, pp. 187-222, 1991.

[Ukkonen95] Ukkonen, E. *On-line Construction of Suffix Trees*. Algorithmica, Vol. 14, Issue 3, pp. 249-260, 1995.

[Valdes-Perez94] Valdes-Perez, R. Conjecturing Hidden Entities via Simplicity and Conservation Laws: Machine Discovery in Chemistry. Artificial Intelligence, Vol. 65, Issue 2, pp. 247-280, 1994.

[Valdes-Perez99] Valdes-Perez, R. Discovery Tools for Science Apps. Communications of the ACM, Vol 42, Issue 11, pp. 37-41, 1999.

[vanRijsbergen75] van Rijsbergen, C.J. and Croft, W.B. (1975). *Document Clustering: An Evaluation of some Experiments with the Cranfield 1400 collection*. Information Processing and Management, Issue 11, pp. 171-182, 1975.

[VanRijbergen79] van Rijsbergen, C. J. *Information Retrieval - 2nd Edition*. Butterworths, London, 1979.

[Vogt98] Vogt, C. C. & Cottrell, G. W. (1998). *Predicting the Performance of Linearly Combined IR Systems*. In Proceedings of SIGIR'98, pp. 190-196, 1998.

[Voorhees85] Voorhees, E. M. The cluster hypothesis revisited. In *Proceedings of ACM SIGIR,* pp. 188-196, 1985.

[Wang92] Wang,Y., and Kitsuregawa M. *Evaluating Contents-Link Coupled Web Page Clustering for Web Search Results*. Proceedings of the eleventh international conference on Information and knowledge management, pp. 499-506, ACM Press, 1992.

[Ward67] Ward, J. *Hierarchical Clustering to Optimize an Objective Function*. Journal of the American Statistical Association, Issue 58, pp. 236-244, 1967.

[Webber2001] Webber, M. Vos, R, Klein, M., and de Jong-van den Berg. *Using Concepts in literature-based Discovery: Simulating Swanson's Raynaud-fish-oil and migraine-magnesium Discoveries*. Journal of the American Society for Information Science and Technology, Vol. 52, Issue 7, pp. 548-557, 2001.

[Williams84] Williams, M. *What makes RABBIT run?* Journal of Man-Machine Studies, Vol. 21, pp. 333- 352, 1984.

[Williamson2000] Williamson, R. *Relevance and Reinforcement in Interactive Browsing*. In Proceedings of SIGIR 2000, pp. 151-170, 2000.

[White81] White, H. *Cocited Author Retrieval Online: An Experiment with the Social Indicators Literature*. Journal of the American Society for Information Science, Issue 32, pp. 16-22, 1981.

[Xu96] Xu, J, Croft, B. *Query Expansion using Local and Global Document Analysis*. In Proceedings of SIGIR'96, pp. 4-11, 1996.

[Zamir98] Zamir, O., Etzioni, O. *Web document clustering: a Feasibility Demonstration*. Proceedings of SIGIR'98, pp. 45-54. ACM Press, 1998.