# WebWatcher: Machine Learning and Hypertext

**Thorsten Joachims, Tom Mitchell, Dayne Freitag, and Robert Armstrong***
School of Computer Science
Carnegie Mellon University
May 29, 1995

## Abstract

This paper describes the first implementation of WebWatcher, a *Learning Apprentice* for the World Wide Web. We also explore the possibility of extracting information from the structure of hypertext. We introduce an algorithm which identifies pages that are related to a given page using only hypertext structure. We motivate the algorithm by using the *Minimum Description Length* principle.

## 1 Introduction

The World Wide Web is growing quickly and addresses more and more users. Although a lot of information is available in the World Wide Web (WWW), it is difficult to find particular pieces of information efficiently. Many have noted the need for software that helps the user search for information. This paper describes the design of WebWatcher [Armstrong et al., 1995], an agent which assists users in locating information on the WWW or searches autonomously on their behalf. In interactive mode WebWatcher acts as a *Learning Apprentice* [Mitchell et al., 1985] [Mitchell et. al., 1994]. It follows the user on his or her way through the World Wide Web and suggests hyperlinks whenever it is confident enough. WebWatcher learns by observing the user's reaction to this advice as well as by the eventual success or failure of the user's actions. The first implementation supports only this interactive mode. First we describe the design and the initial implementation of WebWatcher. We then introduce an algorithm which is used in WebWatcher to suggest pages related to the current page.

## 2 WebWatcher

This section describes the design of WebWatcher and how it assists users in their search for information. The system can be installed easily on any HTML-page by inserting a hyperlink to the WebWatcher server. This allows having multiple instances of WebWatcher which are experts for certain parts of the WWW. A user enters WebWatcher by clicking on a hyperlink to the WebWatcher server and can specify his or her interests by giving keywords. After that WebWatcher takes the user back to the page from which he or she entered the system. From now on WebWatcher follows the user's actions and suggests hyperlinks using its learned knowledge. It also offers other useful functions. The user can leave WebWatcher at any time by telling the system whether the search was successful or not. WebWatcher offers the following functionality:

1. highlighting hyperlinks on the current page, which WebWatcher deems useful according to the user's stated interests
2. adding new hyperlinks to the current page, based on the user's interests
3. suggesting pages related to the current page
4. sending email messages to the user whenever specified pages change

Figures 1 to 5 illustrate the sequence of web pages a user visits in a typical example. Figure 1 shows an HTML-page about machine learning in which we inserted a hyperlink to WebWatcher (line 6). The user follows this hyperlink and gets to a page which allows her to identify the type of information she seeks. In this scenario the user is looking for a publication and selects the category "paper". She is presented a form to elaborate the information request (figure 2). The user can fill in arbitrarily many keywords or leave fields blank. After that the user is sent back to the page from which she entered the WebWatcher system (figure 3).

But now WebWatcher is "looking over her shoulder" and modifies the page in three ways. (1) WebWatcher inserts a menubar on top of the original page. This menubar allows the user to invoke additional functions of WebWatcher or to terminate the search. (2) WebWatcher suggests additional hyperlinks above the menubar (figure 3, line 2). (3) WebWatcher highlights hyperlinks in the actual page which seem interesting according to the information seeking goal. The system highlights hyperlinks by putting "eyes" around them (figure 3, line 13). The size of the eye icon is a measure of WebWatcher's confidence in the advice. In our example the user follows WebWatcher's advice and takes the *"ILPNET"* hyperlink. She arrives at the page shown in figure 4. Until the user quits the search, WebWatcher will insert the menubar into the original page and give advice. While WebWatcher suggests which hyperlinks the user should take, the user remains firmly in control and may ignore the system's advice at any time. We think this is important because WebWatcher may provide imperfect advice, and because WebWatcher might not perfectly understand the user's information seeking

*Email: <first name>.<last name>@cmu.edu

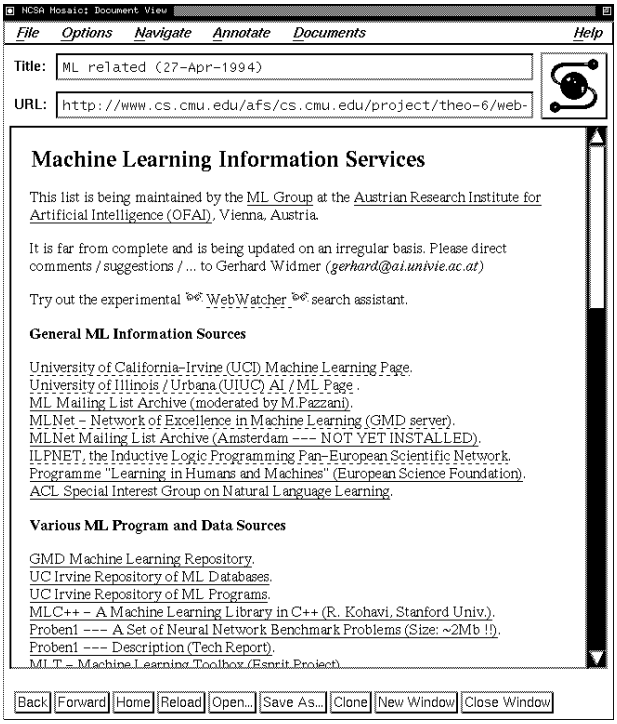


Figure 1: HTML-page with hyperlink to WebWatcher

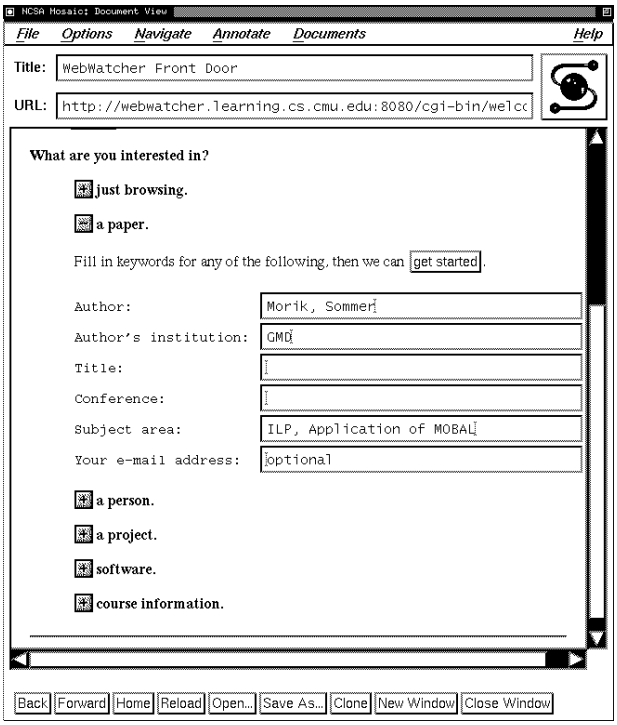


Figure 2: Paper search form

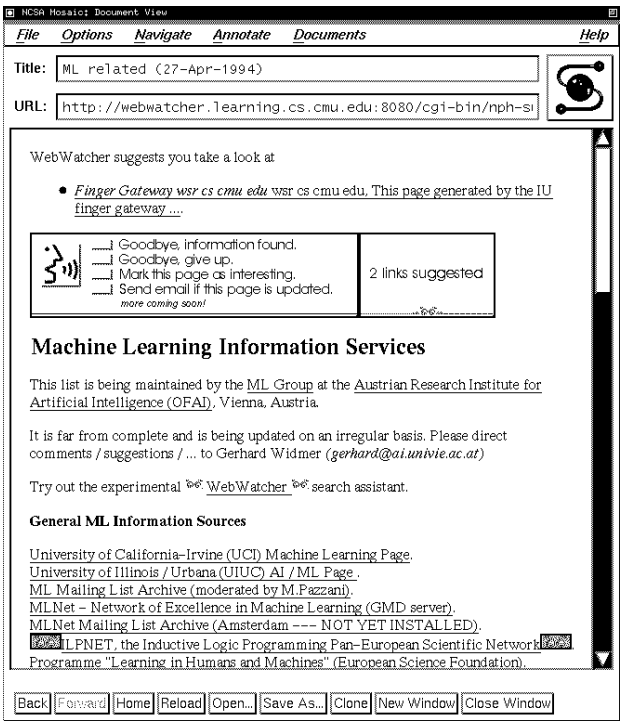


Figure 3: Original page (Abb. 1) with WebWatcher

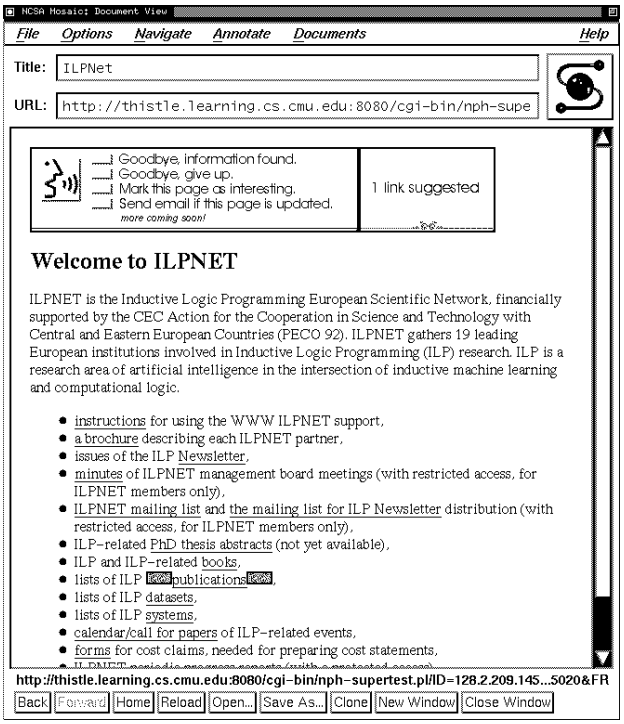


Figure 4: Next page (user has followed WebWatcher's advice)

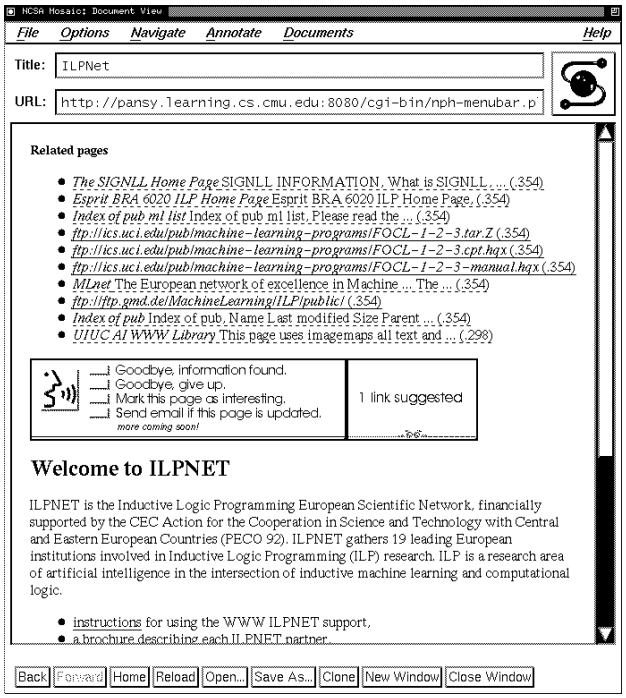


Figure 5: Related pages

goal. In our scenario the user is particularly interested in the "ILPNet" page. So she clicks on the button *"Mark this page as interesting"* in the menubar. WebWatcher stores this information and returns a list of 10 pages which WebWatcher estimates to be closely related (figure 5). The user can leave WebWatcher at any time by clicking on *"Goodbye. Information found."* or *"Goodbye. I give up."* in the menubar.

# 3 Machine Learning

The success of WebWatcher depends heavily on the quality and the quantity of its knowledge. Because of the dynamic nature of the World Wide Web, handcrafting and maintaining this knowledge seems difficult. Consequently, we are exploring methods for aquiring knowledge automatically.

The World Wide Web can be treated as a directed graph:

$$G_{WWW} = \{< P, Q > \mid P \, has \, hyperlink \, to \, Q, \; P, Q \in \mathcal{P}\}$$

$\mathcal{P}$ is the set of HTML-pages, the nodes of the graph. The hyperlinks define the edges. $G_{WWW}$ contains an edge from page $P$ to page $Q$ whenever there is a hyperlink in $P$ pointing to page $Q$.

Two kinds of knowledge are available through the WWW graph. There is knowledge in the nodes of the graph encoded as text. We have begun to explore ways of using this text for guiding search [Armstrong et al., 1995]. But the edges - the hyperlinks - also contain information. They show relations between nodes. The remainder of this paper describes how we can extract knowledge from the edges and how we can use this knowledge for guiding search.

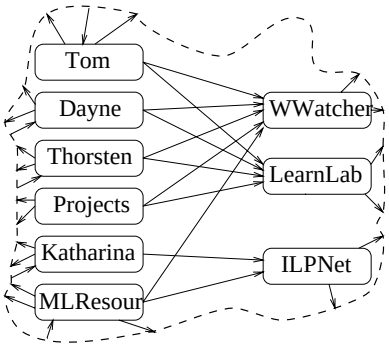


Figure 6: Simple example of WWW structure.

## 3.1 What should be learned?

Imagine we know the structure of the World Wide Web without being able to understand the text in the nodes. What kind of information could we get out of this? We describe below how WebWatcher implements a function to find related web pages using only this structural information about $G_{WWW}$. An example of the use of this functionality is given in figures 4 and 5. In figure 4 the user is at the "Welcome to ILPNET" page, clicks on the button *"Mark this page as interesting"*, and is next presented with the screen depicted in figure 5, which suggests related web pages.

## 3.2 Representation

Web pages are purposely designed to structure the user's search for information. Often web pages represent collections of hyperlinks about certain topics. An example is the "Machine Learning Resources Page" shown in figure 1. Other pages reflect the structure of organizations or the interests of users. Many people put hyperlinks they find interesting into their personal home page.

Ignoring its content, one way we can describe a page in the WWW is in terms of **which other pages contain hyperlinks to it**. In natural language such a description for the *"WWatcher"* page in figure 6 would look like this:

*The "WWatcher" page is a page which Tom thinks is interesting, because Tom has a hyperlink to it in his personal home page. The same is true of Dayne and Thorsten. Furthermore the "WWatcher" page is about a project at CMU, because it appears in the list of projects. It is about Machine Learning, because it is referred to from the "Machine Learning Resources" page.*

Below we describe an algorithm which works under the assumption that two pages are of similar interest if some third page points to them both. Similar ideas have previously been explored in Information Retrieval [Small, 1973]. The performance of query-based Information Retrieval systems has been improved using structural information from hypertext [Savoy, 1992].

### 3.3 Algorithm

The problem we are facing is similar to the problem of *Collaborative Filtering* [Resnick, 1994]. The target function we want to learn is a mapping from an arbitrary web *page* to a set of *related pages*:

$$Related : page \rightarrow \{related\,pages\}$$

WebWatcher uses a *nearest neighbor* approach to approximate this target function. To illustrate, consider the example web fragment from figure 6. The following matrix describes the hyperlinks in this fragment of the web.

| | hyperlink to | | | |
|---|---|---|---|---|
| page | $WWatcher$ | $LearnLab$ | $ILPNet$ | ... |
| $Tom$ | 1 | 1 | 0 | |
| $Dayne$ | 1 | 1 | 0 | |
| $Thorsten$ | 1 | 1 | 0 | |
| $Projects$ | 1 | 1 | 0 | |
| $Katharina$ | 0 | 0 | 1 | |
| $MLResour$ | 1 | 0 | 1 | |
| ⋮ | | | | ⋱ |

The 1 in the row of $Tom$ and in the column of $WWatcher$ says that the $Tom$ page contains a link to the page $WWatcher$. Imagine we want to find pages related to the $WWatcher$ page. As stated above, our assumption is that pages which are referred to from the same pages are related. This means that we have to look at the columns of the matrix and find the ones most similar to the $WWatcher$ column. The pages associated with the $n$ most similar columns are returned by *Related*.

We use *Mutual Information* [Quinlan, 1993] as a similarity measure for comparing columns, which we discuss in section 3.4. In our example *Mutual Information* measures—intuitively speaking—how well the occurrence of particular hyperlink in a page predicts the occurrence of another hyperlink in the same page (e. g. "How well does the occurrence of a hyperlink to the *LearnLab* page predict the occurrence of a hyperlink to the *WWatcher* page?"). Let $\mathcal{D}$ be a set of feature vectors. Each element of $\mathcal{D}$ corresponds to a row of the above matrix. Each feature vector represents a web page with associated attributes, each attribute corresponding to the existence or absence of an outgoing hyperlink to another particular page. We will use the naming convention that $l_i$ stands for any hyperlink to page $P_i$. For our problem the *mutual information* of hyperlink $l_i$ with respect to hyperlink $l_j$, over the set of web pages $\mathcal{D}$, is:

$$I(\mathcal{D}, l_i, l_j) = E(\mathcal{D}, l_i) \Leftrightarrow \frac{m_+}{m} * E(\mathcal{D}_+, l_i) \Leftrightarrow \frac{m_-}{m} * E(\mathcal{D}_-, l_i)$$

where $\mathcal{D}_+$ is the set of pages in $\mathcal{D}$ containing hyperlink $l_j$, and $\mathcal{D}_-$ is the set not containing $l_j$. $m = card(\mathcal{D})$ is the number of pages in $\mathcal{D}$. $m_+ = card(\mathcal{D}_+)$, $m_- = card(\mathcal{D}_-)$. $E(\mathcal{X}, l_i)$ is the entropy function with respect to attribute $l_i$.

In order to consider hyperlinks $l_j$ and $l_i$ similar in our problem, we require that they obey an additional condition. A binary attribute $l_j$ can predict the occurrence of another binary attribute $l_i$ in two ways. Either $l_j$ is true whenever $l_i$ is true, or $l_j$ is complementary to $l_i$. To avoid regarding attributes $l_i$ and $l_j$ as similar although they are complementary, they have to satisfy the condition, $NonComp(\mathcal{D}, l_i, l_j)$.

$$NonComp(\mathcal{D}, l_i, l_j) \Leftrightarrow \frac{p_+}{m_+} \geq \frac{p_-}{m_-}$$

$p_+$ is the number of pages in $\mathcal{D}_+$ which contain hyperlink $l_i$. $p_-$ is the number of pages in $\mathcal{D}_-$ which contain hyperlink $l_i$.

The following algorithm implements the target function *Related*. It returns the $n$ most related pages for a given page $P_{in}$:

`Input:` `page` $P_{in} \in \mathcal{P}$, `output length` $n$

- `for all hyperlinks` $l_i$ `which satisfy` $NonComp(\mathcal{D}, l_{in}, l_i)$:
  - `Calculate` $I(\mathcal{D}, l_{in}, l_i)$

`Output:` `the` $n$ `pages to which the hyperlinks with highest mutual information point`

The algorithm can intuitively be extended to use multiple input pages $P_{in_1}, \ldots P_{in_m}$. The evaluation of a hyperlink $l_i$ is then the sum of the mutual information of $l_i$ and each element of $\{l_{in_1}, \ldots l_{in_m}\}$.

### 3.4 Minimum Description Length Interpretation

In this section we motivate our choice of mutual information as a similarity function. Our argument is based on the *Minimum Description Length* (MDL) principle [Rissanen, 1978]. This principle says that in a machine learning problem one should prefer the hypothesis $H$ which minimizes the number of bits needed to encode the labelling of the training examples (given hypothesis $H$) plus the number of bits needed to encode the hypothesis $H$ itself. Unlike *Maximum Likelihood* methods, the MDL principle models the trade-off between training error and hypothesis complexity. More precisely the number of bits saved by having to encode fewer exceptions from $H$ is compared with the number of bits it costs to encode a more complex hypothesis. The *Minimum Description Length* principle can be derived from Bayes theorem (see, e.g., [Lang, 1995]).

To apply MDL to our problem, consider the target function

$$ContainsHyperlink\,l_{in} : page \rightarrow \{0, 1\}$$

This function predicts whether a page contains hyperlink $l_{in}$. The hypothesis language $\mathcal{L}_\mathcal{H}$ we use for the prediction task is very simple. It consists only of single attributes which are all the hyperlinks $l_i$ except $l_{in}$ itself.

$$\mathcal{L}_\mathcal{H} = \{l_i |\, P_i \in \mathcal{P}\} \Leftrightarrow \{l_{in}\}$$

This means that we can express hypotheses of the form "hyperlink $l_{in}$ occurs in page $\Leftrightarrow$ hyperlink $l_i$

occurs in page". According to the MDL principle the hypothesis $h \in \mathcal{L}_{\mathcal{H}}$, which minimizes the description length of the target values for the training examples of $ContainsHyperlink\, l_{in}$ given $h$, plus the description length of $h$, should be used for approximating $ContainsHyperlink\, l_{in}$. This hypothesis is most likely to predict best which pages contain a hyperlink $l_{in}$. Under the assumption that pages which are referred to from the same pages are similar the page associated with the best hypothesis for $ContainsHyperlink\, l_{in}$ has the highest probability of being most similar with $P_{in}$ in our model.

The algorithm described in the previous section corresponds to applying the MDL principle to $\mathcal{L}_{\mathcal{H}}$, under some simplifying assumptions. In particular, if one assumes that the prior probability distribution over hypotheses in $\mathcal{L}_{\mathcal{H}}$ is uniform, then the description length is the same for each hypothesis. This allows ignoring the description length of the hypothesis, and focusing only on the description of target values given the hypothesis. Furthermore we can maximize the reduction in description length instead of minimizing the absolute number of bits. *Mutual information* is proportional to this reduction.

The MDL analysis suggests that a better algorithm can be derived if these simplifying assumptions were not made. For predicting how interesting another page is given that the user likes the current page a prior distribution over hypotheses in $\mathcal{L}_{\mathcal{H}}$ derived from the frequencies of hyperlinks occuring in the World Wide Web would fully exploit the MDL framework.

# 4 Results

This paper describes work in progress. At this point we do not have final evaluations of the effectiveness of this algorithm in suggesting interesting related pages, although preliminary tests are encouraging. Nevertheless it is possible for the reader to try the algorithm as part of WebWatcher. The system is accessible to the public from the WebWatcher Home Page at URL `http://www.cs.cmu.edu/afs/cs/project/theo-6/web-agent/www/project-home.html`.

# 5 Conclusion

WebWatcher is a flexible interface which allows learning from users' actions and assists them interactively while they are browsing the World Wide Web. We have shown here an algorithm that extracts and uses information stored in the structure of hypertext, without considering the text itself. We conjecture that the use of structural information can improve text based methods in many cases.

# 6 Future Work

Our next goal is to find adequate measures for testing the performance of the algorithm. We also plan to explore how to use other structural information such as outgoing links of a page. Furthermore, we plan to apply the algorithm to problems of collaborative filtering.

# 7 Acknowledgements

We thank Ken Lang, Sean Slattery, Phoebe Sengers and Adrian Perrig for helpful comments on this paper. This research is supported by a Rotary International fellowship grant, an NSF graduate fellowship, and by Arpa under grant number F33615-93-1-1330.

# References

[Armstrong et al., 1995] R. Armstrong, D. Freitag, T. Joachims, T. Mitchell, *"WebWatcher: A Learning Apprentice for the World Wide Web"*, 1995 AAAI Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments, March 1995. URL http://www.cs.cmu.edu/afs/cs/project/theo-6/web-agent/www/webagent-plus.ps.Z

[Lang, 1995] K. Lang, *"NewsWeeder: Learning to Filter Netnews"*, International Conference on Machine Learning, 1995. URL http://anther.learning.cs.cmu.edu/ml95.ps

[Mitchell et al., 1985] T. Mitchell, S. Mahadevan, and L. Steinberg, *"LEAP: A Learning Apprentice for VLSI Design"*, Ninth International Joint Conference on Artificial Intelligence, August 1985.

[Mitchell et. al., 1994] T.M. Mitchell, R. Caruana, D. Freitag, J. McDermott, and D. Zabowski, *"Experience with a Learning Personal Assistant"*, Communications of the ACM, Vol. 37, No. 7, pp. 81-91, July 1994. URL http://www.cs.cmu.edu/afs/cs/user/mitchell/ftp/cacm.ps.Z

[Quinlan, 1993] J.R. Quinlan, *"C4.5: Programs for Machine Learning"*, Morgan Kaufmann, 1993.

[Resnick, 1994] Paul Resnick et al. *"GroupLens: An Open Architecture for Collaborative Filtering of Netnews"*, Internal Research Report, MIT Center for Coordination Science, March 1994. URL http://www-sloan.mit.edu/ccs/1994wp.html

[Rissanen, 1978] J. Rissanen, *"Modelling by Shortest Data Description"*, Automatica, 14, 1978, 465-471.

[Savoy, 1992] Jacques Savoy, *"Ranking Schemes in a Hypertext Retrieval System"*, Universite de Montreal Departement d'Informatique et de Recherche Operationnelle Publication, Publication #811, February 1992.

[Small, 1973] H. Small, *"Co-Citation in the Scientific Literature: A New Measure of the Relationship Between Two Documents"*, Journal of the American Society for Information Science, 24(4), 1973, 265-269.