Planning to guide concept understanding in the WWW

Seiji Yamada
CISS, IGSSE
Tokyo Institute of Technology
Midori-ku, Yokohama 226-8502, Japan
yamada@ymd.dis.titech.ac.jp
http://www.ymd.dis.titech.ac.jp/~yamada/

Yukio Osawa
Faculty of Engineering Science
Osaka University
Toyonaka, Osaka 560-8531, Japan
osawa@yachi-lab.sys.es.osaka-u.ac.jp
http://w3.sys.es.osaka-u.ac.jp/~osawa/

Abstract

This paper describes a novel navigation planning method that generates a plan (a sequence of Web pages) guiding concept understanding in the WWW (World Wide Web). It also has the ability to generate operators during planning from Web pages using keyword extraction methods. When a user wants to understand a concept, it is useful to browse for relevant Web pages in the WWW. However, in general, this task is very hard because the user does not know where such Web pages are, and has to search for them in the vast WWW search space. Even with a search engine, this consumes the user's energy. To deal with this problem, we propose navigation planning to automatically generate a sequence of Web pages by which a user systematically understand a target concept. First, with a planning framework, we formalize the browsing task in the WWW. Action is defined as the understanding of a Web page, and an operator for a Web page consists of conditional/effect knowledge. Unfortunately it is impossible to prepare operators for all the Web pages. Hence we develop a method to generate an operator from a Web page by extracting condition/effect terms with keyword extraction techniques. Then the whole planning procedure is describes. Finally we fully implement the navigation planning system and make experiments by comparing with methods using a search engine and link-tracing (like a Web robot). As results, we found out navigation planning is a promising approach to assist the concept understanding in the WWW.

Introduction

The accessible information through the Internet is increasing explosively as the WWW becomes widespread. In this situation, the WWW is very useful for a user who wants to understand a concept (called a *target concept*). He or she can browse helpful Web pages to understand a target concept. However, in general, this task is very hard because he or she may not know where such Web pages are, and has to search them over the vast WWW search space, and this consumes the user's energy.

A practical and simple solution of the problem is to use a search engine like MetaCrawler, AltaVista, YaHoo, with the target concept as a query. The engine provides a list of relevant Web pages. However, since the retrieved Web pages are not filtered sufficiently, a user has to select useful ones from them. Furthermore, since in most cases the retrieved Web pages include concepts that a user does not understand, he or she must search the useful Web pages for them using a search engine again. This task is repeated until a user understands the target concept, and wastes time.

We consider the task as planning, and propose *navigation planning* (*NaviPlanning* for short) to automatically generate a sequence of Web pages which can guide a user to understand a target concept. First, with an AI planning framework, we formalize the browsing in the WWW. Action is defined as the understanding of a Web page, and an operator consists of conditional knowledge and effect knowledge. Unfortunately it is impossible to give *NaviPlanning* the operators for all the Web pages. Thus we develop a method to generate operators from Web pages by extracting condition/effect terms with keyword extraction techniques. Finally we fully implement the *NaviPlanning* system and make experiments by comparing with methods using a search engine and link-tracing (like a Web robot).

There are studies on planning to generate procedures for gathering information through computer networks. The *Softbot* (Etzioni & Weld 1994) provides a framework and a interface for describing operators. A complete partial-ordering planner is used. *Occam* (Kwok & Weld 1996) is also a planner for gathering information. It is more efficient and able to reason about the capabilities of different information sources. *Sage* (Knoblock 1995) was developed for integrating planning, execution, replanning, and sensing to gathering information in distributed resources. The aim of these studies is to generate a plan as a procedure of gathering information, and a plan consists of UNIX commands, database operations. In contrast with these studies, the aim of *NaviPlanning* is to generate a plan which can guide a user to understand a concept in the WWW, and its plan is a sequence of Web pages.

Our research is concerned with the *Web Robot* (Jamsa, Lalani, & Weakley 1996) which is used to

gather Web pages for a search engine database. However it is not controlled and traces to only linked pages. *NaviPlanning* can trace to unlinked pages, and is controlled for generating a plan.

Some learning systems have been developed for information gathering and browsing in the WWW. *ShopBot* (Doorenbos, Etzioni, & Weld 1997) learns the text pattern indicating the price of CD-ROMs, and searches for the cheapest one more efficiently than a human. The purpose of *ShopBot* is different from our research. *WebWatcher* (Armstrong *et al.* 1995) and *Letizia* (Lieberman 1995) are able to indicate the Web pages which a user wants to see next. Using browsing history, they learn to predict useful Web pages for a user. Unfortunately they indicate only related Web pages to the current page, and do not generate systematic guidance for understanding a concept like *NaviPlanning*.

Navigation planning

In this research, *navigation* means a task that indicates useful Web pages to a user for guiding his concept understanding. A sequence of useful Web pages is called a *plan*, and *navigation planning* means the automatic generation of the plan.

By using an AI planning framework, we formalize the task that a user browses useful Web pages to understand a target concept. We can summarize the task in the following. This procedure is repeated until terminated by the user.

1. Search Web pages related to target concepts using a search engine.
2. Understand the Web pages retrieved by the search engine.
3. Select unknown concepts in the Web pages.
4. Go to *Step*1 with unknown concepts as target concepts.

The procedure above is considered *planning* (Fikes & Nilsson 1971)(Russell & Norvig 1995) using the following correspondence.

- *Action*: understanding concepts described in a Web page.
- *State*: A user's knowledge state: a set of words describing concepts which he knows.
- *Initial state*: A user's initial knowledge state.
- *Goal state*: a set of words which a user wants to understand.
- *Operator*: *U-Op(URL)* describing action is defined by the followings.
  - *Label*: URL of the Web page
  - *Condition*: $C = \{c_1, \cdots, c_i\}$, where C means the *condition knowledge* which is necessary to understand the Web pages, and c_k is an element (called a *condition word*).
  - *Effect*: $E = \{e_1, \cdots, e_j\}$, where E is *effect knowledge* which a user obtains by understanding the Web page, and e_l is an element (called an *effect word*).

The condition knowledge and effect knowledge are described with a set of words describing the knowledge. Using this formalization, we can apply a classical planning framework (Fikes & Nilsson 1971)(Russell & Norvig 1995) to navigation planning.

This *NaviPlanning* contains a significant problem which has not been in planning thus far. It is that the *U-Op(URL)* operators are not given in advance. This is because it is impossible for a human designer to generate the operators from all the Web pages in the WWW. Hence the operators need to be automatically generated from Web pages when they are necessary. In next section, we develop the method.

Generating operators from Web pages

We develop a method to extract condition and effect knowledge from a Web page in order to generate an operator. Since the condition/effect words are assumed to be written in the Web page, thus the problem is how to extract condition/effect words from a Web page (an html file).

Using TAG structure in a html-file

Various methods to extract keywords from text have been studied (Salton & Buckley 1997). Though most methods are based on frequency of words, one of the most effective methods is to utilize the structure in a text. Since a Web page is described in a HTML format (Berners-Lee & Connolly 1995), we can utilize TAG structures like `<TITLE>`, `<Hn>`, `<A HREF=...>`, for extracting condition/effect knowledge.

Extracting condition words The prime candidates for condition words are the words linked to other Web pages, i.e. the words between `<A HREF=URL>` and `</A>`, because an author explicitly indicates that the words are important for understanding the Web page. They have URL for a useful Web page to understand the words.

However all the linked words are not meaningful. For example, a commercial Web page may be linked to the Web page of it's sponsor company. Hence we filter the candidates from `<A HREF=URL>` tag using *KeyGraph* as mentioned later.

Extracting effect words Since the title of the Web page describes words which a user acquires by reading the page, the words between `<TITLE>` and `</TITLE>` are candidates for the effect words. In the same way, headings describe knowledge which a user obtains by reading the section. Thus the words between `<Hn>` and `</Hn>` are also candidates for effect words.

Input

- Input file (html-file or plain text file) and it's URL

Control parameters

- Weight for `<TITLE>`: α, Weight for `<Hn>`: β, Weight for `<A HREF=URL>`: γ
- The number of condition words: N_C
- The number of effect words: N_E

Output

- An operator *U-Op(URL)* = (URL, condition knowledge, effect knowledge).

Procedure

1. By *KeyGraph*, generate candidates C for condition knowledge and candidates E for effect knowledge from an input file. The candidates have weight within [0, 1].
2. Add the words between `<Hn>` and `</Hn>` with weight β to E.
3. Add the words between `<TITLE>` and `</TITLE>` with weight α to E.
4. Change the weight of the overlaping words between C and the words between each `<A HREF=URL>` and `</A>` to γ.
5. Select the N_c maximum weighted words in C for conditional knowledge and the N_E maximum weighted words in E for effect knowledge, and generate an operator *U-Op(URL)*. If a word in condition knowledge is extracted by `<A HREF=URL>` and has a URL, the URL is assigned to the condition word.

Figure 1: **Ext-OP**: procedure of generating an operator

KeyGraph: a keyword extraction method

The extraction of condition/effect words using only the tag structure is not sufficient. All the linked words are not candidates for condition words, and all the condition words are linked. Thus we need to utilize another method to assist it, and *KeyGraph* is used.

KeyGraph is a fast method for extracting keywords representing the asserted core idea in a document (Ohsawa, Benson, & Yachida 1998). *KeyGraph* composes clusters of terms, based on co-occurrences between terms in a document. Each cluster represents a concept on which the document is based (i.e. condition words), and terms connecting clusters tightly are obtained as author's assertion (i.e. effect words). Furthermore the likelihood for condition/effect words can be computed by *KeyGraph*, and used for weight of an operator.

Another merit of *KeyGraph* is that it does not employ a corpus. Generally speaking, corpus-based methods such as TFIDF (Salton & McGill 1983) have to revise the corpus when new documents occur. Accordingly, keywords of all the documents already indexed have to be revised again – an enormous loss of time especially in the WWW, where many new and progressing topics appear in new Web pages everyday.

The extraction of condition/effect words using tag structure and *KeyGraph* are integrated, and the detailed procedure **Ext-OP** is shown in Fig.1.

Planning procedure

We now develop *NaviPlanning* procedure. Fig.2 shows the overview of *NaviPlanning*. It uses *backward beam search* (Russell & Norvig 1995) from a goal state (Fig.2(a)). The node expansion (Fig.2(b)) includes the search for related Web pages with a search engine and the generation of operators with **Ext-OP**.

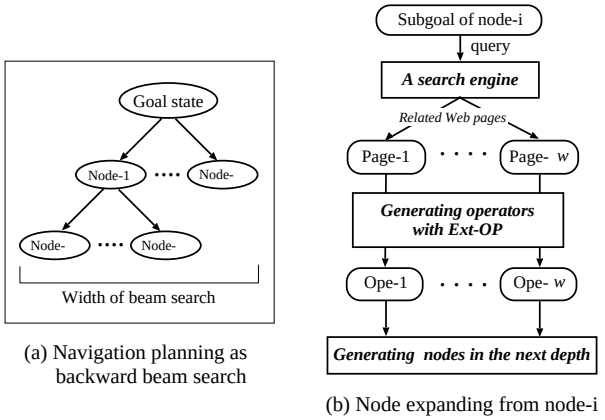
The detail of input, output and a procedure for the *NaviPlanning* is shown in Fig.3 and Fig.4. In this research, the evalution function H for a node is defined with the following formula.

$$H = \sum \textit{Weight of a satisfied subgoal} + \sum \textit{Weight of effect words satisfying subgoal}$$

Fig.5 shows a plan (limited depth $l = 4$) generated by *NaviPlanning* with the target concept "Turing test " and context "Turing". Four useful Web pages are indicated with hyperlinked titles and URLs.

Pruning and caching Web pages

Additional procedures: pruning and caching Web pages are introduced to accelerate *NaviPlanning*. We experimentally found that many Web pages have little information to understand something. For example, the Web page having only linked URLs or references, or extremely short or long pages. Since the operator



(a) Navigation planning as backward beam search

(b) Node expanding from node-i

Figure 2 Navigation planning (overview)

Input

- *Goal state* (= *target concepts*) G_0 : a set of the words which a user wants to understand.
- *Context GC* : a set of the words describing the background domain of the goal state.

Control parameters

- Initial state IS : a user's initial knowledge state.
- Limited depth l : planning stops at depth l.
- Limited number of Web page w : the number of the Web pages obtained from a search engine.
- Width b for *beam search*.
- Evaluation function H : evaluating the states in planning.

Output

- A plan P : a set of sequences of *U-Op(URL)*.

Figure 3 Input and output for Navigation planning

generated from a meaningless Web page is meaningless, such Web pages are pruned in *Step*3d of Fig.4.

Caching is also done by storing the Web pages obtained in *Step*3c of Fig.4. If a cached Web page is available in *Step*3c of Fig.4, it is used and planning is accelerated more than two times.

Experiments and Results

Accuracy of operators formation

In the first phase of experiments, we investigated how accurately the operators in the navigation planing were formed. The procedure in this experiment is as follows:

1) Take a document (a Web page) D, from a plan for a query (the goal state and context). Let $Cond$ and Eff denote the sets of words obtained from D, as the condition words and effect words respectively.

2) For $Cond$: Count C, the number of words in $Cond$, and c, the number of really necessary condition words of D, i.e. words in $Cond$ without understanding which the reader cannot understand document D. Also count C_e, the number of words in $Cond$ which seem to actually be effect words.

3)For Eff : Count E, the number of words in Eff, and e, the number of effect words understandable by reading D. Also count E_c, the number of terms in Eff which seem to actually be condition words.

The results from 23 Web pages are: $C = 494$, $c = 367$, $E = 331$, $e = 228$, $C_e = 15$, $E_c = 25$.

From these values, we obtain $c/C = 0.74$ and $e/E = 0.68$, which mean the accuracy of condition words and effect words respectively. We have no previous methods to compare with these values, because no previous indexing method obtains condition or effect words.

1. Initialize $N_0 = [n_0] = [(G_0, [])]$, $i = 0$. (N_i is a sequence of nodes at depth i. A node is described as $n_i = (G_i, P_i)$, where G_i is a set of the words for a subgoal, and a plan P_i is a sequence of operators.)
2. If $i = l$ or N_i is null then this procedure outputs P and stops.
3. Apply the following procedures to all the nodes n in N_i, and initialize $N_{i+1} = []$.
   (a) Extract D (called *difference*): a subset of G which is not included in IS.
   (b) Obtain w Web pages' URLs by inputing $D \cup GC$ as query to a search engine.
   (c) Get the Web pages (html files) of the obtained URL through a TCP/IP connection.
   (d) Generate operators from the obtained Web pages using the **Ext-OP** procedure. Then apply the following procedure to the operators which the intersection between its effect knowledge E and n's subgoal G is not empty. Get G' by eliminating the operator's E from n's G and adding the operator's C. Getting P' by adding the operator to the head of n's plan P. Then a new node (G', P') is added to N_{n+1}.
4. Evaluate all node in N_{n+1} using an evaluation function H, and update N_{n+1} with the best b nodes.
5. Set $i = i + 1$ and go to *Step*2.

Figure 4 Navigation planning procedure

However, we can at least say that words separated into $Cond$ and Eff as expected, because the overall rate of real conditional and effect terms are much less than 0.74 and 0.68, i.e. $(c + E_c)/(c + E_c + e + C_e) = 0.62$ and $(e + C_e)/(c + E_c + e + C_e) = 0.38$.

Operators are further evaluated in the next experiment, considering the practical utility of *NaviPlanning*.

Efficiency of understanding goal terms

In the second experiment, the utility of *NaviPlanning* was directly evaluated. The MetaCrawler[1](Selberg & Etzioni 1995)(Selberg & Etzioni 1997) search engine was employed for node expansion (Fig.2(b)) in *NaviPlanning*, and also used for the purpose of user's understanding of a target concept. The performances of these two methods, combined with link-tracing (like a Web robot) or not, were compared. In all experiments, the control parameters (Fig.3) in *NaviPlanning* were set as $IS = []$, $l = 4$, $w = 8$, $b = 4$, and the control

[1]http://www.metacrawler.com/

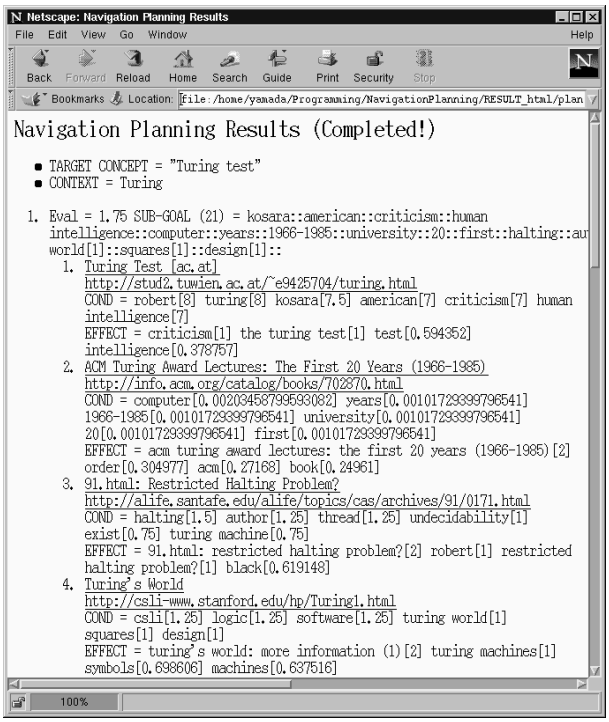


Figure 5 Plan for "Turing test"

parameters (Fig.1) in **Ext-OP** were set as weights $\alpha = 2$, $\beta = 1$, $\gamma = 1$, $N_C = 6$, $N_E = 4$. Since IS is null, *NaviPlanning* stopped at depth = 4, and only 4-step plans were generated. The time taken to generate a plan by *NaviPlanning* was $2 \sim 3$ hours (the most part was taken for getting Web pages). Additionally experiments with depth = 8 were made, however planning was not completed in most cases.

In more concrete words, we compared the following five methods.

Method-1: Search engine Goal state and context terms were given as a query, constraining that the goal *must* be included in each page to be retrieved. The user read only the retrieved pages in the order as sorted by MetaCrawler, until he or she understood the goal term clearly.

Method-2: Search engine + Tracing links Queries were given as in Method-1. Then the user read the retrieved pages in the order as sorted by MetaCrawler, being allowed to trace links from the pages being read, until he or she understood the goal term explicitly.

Method-3: NaviPlanning The user gave the goal term and context terms separately to *NaviPlanning*. Then, he or she read the retrieved pages in the order as sorted by *NaviPlanning*.

Method-4: NaviPlanning + Tracing links Queries were given as in Method-3. Then, he or she read the retrieved pages in the order as sorted by *NaviPlanning*, being allowed to trace links from the pages being read, until he or she understood the goal term explicitly.

Method-5: The user traced links only from the top-page of MetaCrawler. The user was not allowed to start from other retrieved pages of MetaCrawler.

43 queries were tested for each method. The number of clearly understood goals by Method-1 and Method-3 was both 26 goals. Method-5 did the far poorest, making the user understand only 13 goals. These values may sound as though MetaCrawler and *NaviPlanning* performed equally fine, and tracing links was of no use.

However, let us consider one more significant factor – the effort of the user in each procedure. This factor may clarify the difference of tested methods. The first effect we observed was that *NaviPlanning* remarkably helps the user in understanding goal terms efficiently. For 9 queries, the user of Method-3 was satisfied with his or her own understanding by the first page read, while Method-1 users were in only 6 cases. This fact implies that operators were obtained quite well, i.e. effect words of the start (top) page of *NaviPlanning* corresponded to the assigned goal.

The second effect observed was that tracing links helped in understanding pages which were not understood without tracing links. In total, 13 goals were newly understood by adding link-tracing (i.e., in Methods 2 and 4), among 34 goals which failed to be understood without tracing links.

Due to all these effects, Method-4 was be the most effort-saving approach for the user. Fig.6 compares user effort in the two most effective methods, i.e., Method-2 and 4. Two features are remarkable in Fig.6. One is that Method-4 (*NaviPlanning* + Tracing Links) enabled a user to understand goals while reading many fewer pages than Method-2 (Search engine + Tracing Links). The other feature is that Method-2 requires greater effort of the user for a larger number of pages retrieved by MetaCrawler. This means that, for a popular term (e.g. "MIDI" in digital music, or "genetic algorithm" in the AI area) which is already prevalent in the WWW, it is relatively rare that a page is devoted for defining the term. In other words, common-sense terms are not desired or intended to be defined.

Discussion

In the experiments above, *NaviPlanning* was verified to help a user in efficiently understanding a target concept. The only observation above which may look discouraging for *NaviPlanning* is that the number of successfully understood pages were almost equal for *NaviPlanning* and MetaCrawler.

However, the content of the pages which failed to be understood were quite different between the two methods. This implies that there are both advantages and disadvantages in the current version of *NaviPlanning*.

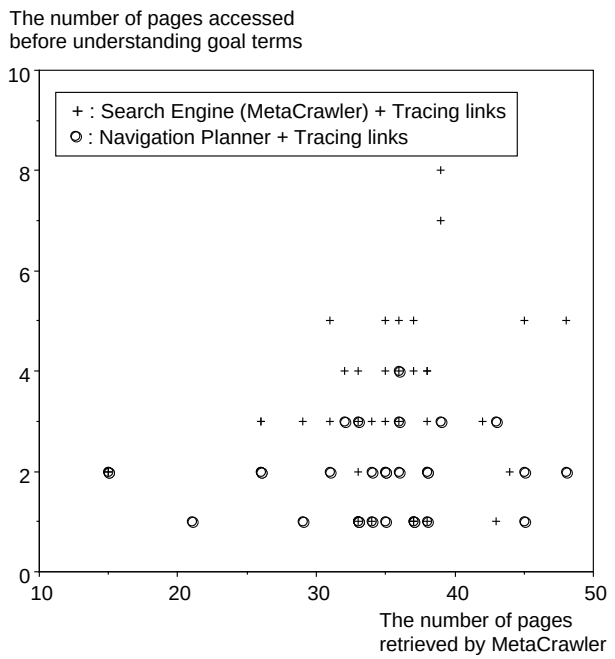


Figure 6: Comparison of the performance of two best methods.

The major advantage is that *NaviPlanning* enables efficient planning for understanding a target concept, by finely formed operators. On the other hand, according to more detailed data, the disadvantage was that because of the width of beam search, it does not retrieve so many pages directly concerned with the target concept as search engines do. Here is the risk of leading to a misguided plan beginning from a badly retrieved start page (the top page of *NaviPlanning*). Currently we are thinking of letting a user select the start page, as the next stage of this research.

Conclusion

We proposed a novel navigation planning method that generates a plan guiding understanding of a concept in the WWW. It also has the ability to generate operators during planning from Web pages using keyword extraction methods. The search for useful Web pages for a user to understand goal concepts was formalized using a planning framework, and an operator corresponding to the understanding of a Web page was defined with conditional/effect knowledge. Then we described the whole planning procedure. Finally we fully implemented the navigation planning system and made experiments by comparing with methods using a search engine and link-tracing (like a Web robot). As results, we found *NaviPlanning* a promising approach to assist information retrieval in the WWW.

References

Armstrong, R.; Freitag, D.; Joachims, T.; and Mitchell, T. 1995. WabWatcher: A learning apprentice for the World Wide Web. In *The 1995 AAAI Spring Symposium on Information Gathering from Heterogeneous, Distributed Environment*. AAAI.

Berners-Lee, T., and Connolly, D. 1995. *Hypertext Markup Language – 2.0*. RFC 1866.

Doorenbos, R. B.; Etzioni, O.; and Weld, D. S. 1997. A scalable comparison-shopping agent for the World-Wide Web. In *Proceedings of the First International Conference on Autonomous Agent*, 39–48.

Etzioni, O., and Weld, D. 1994. A SoftBot-based interface to the Internet. *Communication of the ACM* 37(7):72–76.

Fikes, R. E., and Nilsson, N. J. 1971. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence* 2:189–208.

Jamsa, K.; Lalani, S.; and Weakley, S. 1996. *Web Programming*. Jamsa Press.

Knoblock, C. A. 1995. Planning, executing, sensing, and replanning for information gathering. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, 1686–1693.

Kwok, C. T., and Weld, D. S. 1996. Planning to gather information. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, 32–39.

Lieberman, H. 1995. Letizia: A agent that assists Web browsing. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, 924–929.

Ohsawa, Y.; Benson, N. E.; and Yachida, M. 1998. *KeyGraph*: Automatic indexing by co-occurrence graph based on building construction metaphor. In *Advanced Digital Library Conference*. to appear.

Russell, S., and Norvig, P. 1995. *Artificial Intelligence –A Modern Approach–*. Prentice-Hall.

Salton, G., and Buckley, C. 1997. Term-weighting approaches in automatic text retrieval. In Jones, K. S., and Willet, P., eds., *Readings in Information Retrieval edt.), Morgan Kaufmann*. Morgan Kaufmann. 323–328.

Salton, G., and McGill, M. J. 1983. *Introduction to modern information retrieval*. McGraw-Hill.

Selberg, E., and Etzioni, O. 1995. Multi-service search and comparison using the metacrawler. In *the 1995 World Wide Web Conference*.

Selberg, E., and Etzioni, O. 1997. The metacrawler architecture for resource aggregation on the Web. In *IEEE Expert*, volume January-February. 11–14.