Interactive Web Page Filtering with Relational Learning

Masayuki Okabe[*] and Seiji Yamada

CISS, IGSSE, Tokyo Institute of Technology
4259 Nagatuta-Cho, Midori-ku, Yokohama 226-8502, JAPAN
{okabe, yamada}@ymd.dis.titech.ac.jp

Abstract. This paper describes a system for collecting Web pages that are relevant to a particular topic through an interactive approach. Indicated some relevant pages by a user, this system automatically constructs a set of rules to find new relevant pages. The purpose of the system is to reduce users' browsing cost by filtering non-relevant pages automatically. Such an approach can be useful when users do not know how to describe their requirements to search engines. We describe the representation and the learning algorithm, and also show the experiments comparing its performance with a search engine.

1 Introduction

Search engines are indispensable tools to access useful information which might exist somewhere on the Internet. While they have been getting higher capability to meet various information needs and large amounts of transactions, they are still insufficient in the ability to support the users who need to collect a certain number of Web pages relevant to their purpose. Based on a query(usually composed of a few words[1]) inputted by a user, search engines return a "hit list" in which so many Web pages are presented in a certain order. However it does not often reflect the user's intent, and thus the user would waste much time and energy on judging the Web pages. To resolve this problem and to provide efficient retrieval process, we propose a system which mediates between users and search engines in order to select only relevant Web pages out of hit list through the interactive process called "relevance feedback"[5]. Given some Web pages marked with their relevancy(relevant or non-relevant) by a user, this system generates a set of rules, called *decision rules*, each of which is a logical rule to decide whether the user should look a Web page or not. The system constructs decision rules from the combination of keywords, relational operators and tags with a learning algorithm which is superior to learn structural patterns. We have developed this basic framework in document retrieval[3] and found our approach was promising. In this paper, we applied this method to the intelligent interface which coordinates the hit lists of search engines in order for individual user to find their wanted information easily.

[*] Japan Science and Technology Corporation

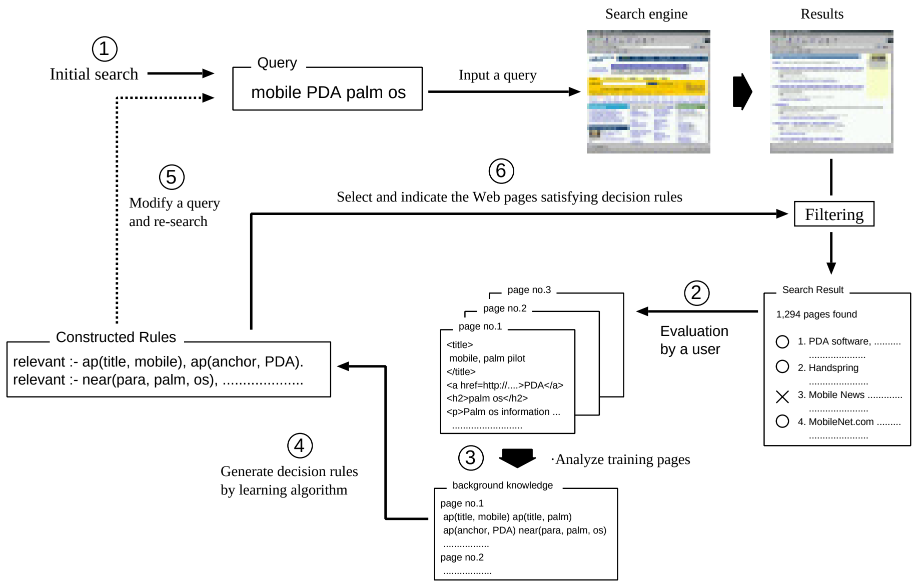
Search engine
Results
① Initial search
Query
mobile PDA palm os
Input a query
⑤ Modify a query and re-search
⑥ Select and indicate the Web pages satisfying decision rules
Filtering
Search Result
1,294 pages found
1. PDA software, ...
2. Handspring
3. Mobile News ....
4. MobileNet.com ....
② Evaluation by a user
page no.3
page no.2
page no.1
<title> mobile, palm pilot </title>
<a href=http://....>PDA</a>
<h2>palm os</h2>
<p>Palm os information ...
Constructed Rules
relevant :- ap(title, mobile), ap(anchor, PDA).
relevant :- near(para, palm, os), ...
④ Generate decision rules by learning algorithm
③ ·Analyze training pages
background knowledge
page no.1
ap(title, mobile) ap(title, palm)
ap(anchor, PDA) near(para, palm, os)
page no.2

Fig. 1. Interactive Web search

In the remainder of the paper, we describe the interactive process, decision rules, and the experiments.

2 Interactive Web search with relevance feedback

Figure 1 shows the interactive Web search process we propose. This process consists of six steps, each of which corresponds to the number in Figure 1. In step 1 a user starts to search by inputting a query to the system, and then receives the hit list. In step 2 the user evaluates and marks the relevancy(relevant or non-relevant) of its upper (more or less)10 pages in order to teach the system what kind of pages are needed. In step 3 the system makes an analysis of the marked pages by extracting extended keywords and literals which are used to construct decision rules. Based on the literals and a learning algorithm, in step 4 the system generates decision rules to distinguish relevant pages and non-relevant ones. We give the detailed description of decision rules in the next section. Step 5 is prepared for the case that the user noticed the initially or previously inputted query was not proper or sufficient and thinks it's better to do re-search. This step is not always done, thus it is indicated by dashed line. Step 6 is the revision procedure in which the system selects(re-selects) relevant pages based on the newly constructed decision rules to provide the user with the better results. These procedures follow the general relevance feedback process, and the steps from 2 to 6 repeat until the user would collect enough relevant pages.

3 Decision rules

3.1 Rule representation

The bodies of decision rules consist of the following literals standing for relations between terms and tags.

- $ap(region\_type, word)$: This literal is true *iff* a word *word* appears within a region of *region_type* in a Web page.
- $near(region\_type, word1, word2)$: This literal is true *iff* both of words w_i and w_j appear within a sequence of 10 words somewhere in a region of *region_type* of a Web page. The ordering of the two words is not considered.

We can easily consider that the importance of words significantly depends on tags of HTML. For example, the words within `<TITLE>` seem to have significant meaning because they indicate the theme of the Web page. Hence we use the *region_type* to restrict a tag with which words are surrounded. We prepare the *region_type* in the followings.

- *title* : The region surrounded with title tags `<TITLE>`.
- *anchor* : The region surrounded with anchor tags `<A>`.
- *head* : The region surrounded with heading tags `<H1∼4>`.
- *para* : The region surrounded with paragraph tags `<P>`.

3.2 Learning algorithm

Figure 2 shows the learning algorithm for decision rules. Under the separate-and-conquer strategy[2] this algorithm repeats mainking a new single rule until each of E^+ is covered by some rules. A single rule starts with empty body, and repeats adding a literal until it does not cover any of E^-. The literal is selected from a condition candidate set C, all of which are the possible combinations among *region_type*s and keywords in K as its arguments. The criteria for selecting a literal which should be added to the body is based on *information gain*[4]. Using information gain, this algorithm searches a good combination greedily and efficiently. However it sometimes selects a bad literal and stops before completion. In such a case, if a current rule has some literals in its body, the rule restarts from an empty body and resumes adding a literal from C except for the literal l_1 which was firstly added in the previous cycle. If the body of a current rule has no literal, a new keyword is added to K and C is updated. The added keyword is selected from terms in positive training pages E^+.

4 Experiments and Results

In order to answer the question of how many relevant pages we can find more with the proposed system in the condition of looking over a certain pages, we conducted two retrieval experiments. The one is a retrieval *not using* our system(*retrieval1*). In this retrieval, we judged 50 pages from the top of the hit list

```
Input: $E^+$ : a set of positive training pages,   $E^-$ : a set of negative training pages
       $C$ : a condition candidate set,  $K$ : a set of extended keywords
Output: $R$ : a set of decision rules.
Variables: $rule$ : a decision rule,  $S$ : a set of exception literals,  $l_1$ : an exception literal
Initialize: $K \leftarrow$ a set of words in a query. $R,\ S,\ l_1 \leftarrow empty.\ rule \leftarrow relevant$:-.
Repeat
 1:  · Investigate the number $p$ of positive training pages satisfying the $rule$
       and the number $n$ of negative training pages satisfying the $rule$.
 2:  if $n = 0$ then
 3:    · Add $rule$ to $R$.
 4:    · Remove a positive training page satisfying the $rule$ from $E^+$.
 5:    if $E^+$ is empty then Finish
 6:    else Initialize $rule$, $S$, $l_1$.
 7:  else
 8:    · For all literals in $C \cap \overline{S}$, compute the information gain $G$.
 9:    if No literal with $G > 0$ then
10:      if the body of the $rule$ is empty then
11:          · Add a keyword to $K$.
12:          · Update $C$.
13:      else
14:         · Initialize $S$ and $rule$.
15:         · Add $l_1$ to $S$, and initialize $l_1$.
16:    else
17:· Select $l_{max}$ having the maximum $G$.
18:      if the body of the $rule$ is empty. then $l_1 := l_{max}$
19:      · Add $l_{max}$ to $rule$ and $S$.
```

Fig. 2. Learning Algorithm

returned by a search engine. The other is a retrieval *using* our system(*retrieval2*). In this retrieval, we made feedbacks every after we judged 10 pages according to the procedure described in Section 2(excluding the pages which are already judged and decision rules don't satisfy). We made total four feedbacks. In both retrieval, we judged total 50 pages from the same hit list. We used the Google as a test WWW search engine, which is recognized as one of the most powerful search engines. For test questions, we used 20 *topics*(No. 401∼420) provided by the small web track in TREC-8(see http://trec.nist.gov). This test collection is often used for evaluating the performance of retrieval systems in Information Retrieval community. We picked up a few words for the initial query to the search engine. Relevance judgment for each page is conducted by the same searcher according to the account written in each topic.

Figure 3 shows the relation between judged pages and relevant pages found in the judged pages. The number of relevant pages is average value per 20 topics. About first 10 pages, there is no difference because two methods judge the same pages. The difference of found relevant pages increases after the first feedback. As a result, retrieval2 got about 5 relevant pages more than retrieval1 after the fourth feedback was done. However the number of found relevant pages differs in every topics. Figure 4 shows the differences of the topics after the fourth feedback. Let A be the number of relevant pages in retrieval1 and B be the one in retrieval2, the difference D is calculated by $D = B - A$. The effect gradually

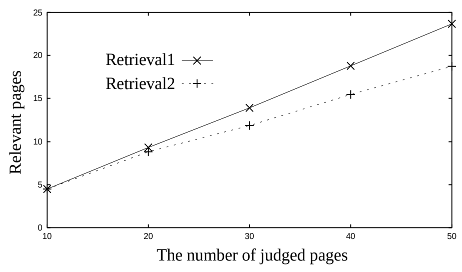

Fig. 3. The average number of relevant pages

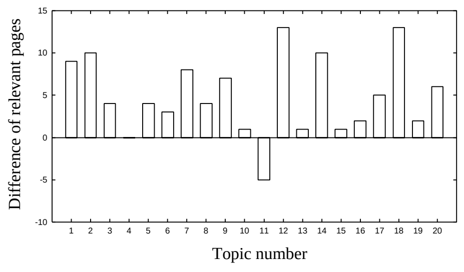

Fig. 4. Difference after the fourth feedback (total 50 pages judged)

increases as the feedback repeats. In figure 4, we can see the effect clearly. Our system produces good results for most of the topics.

5 Conclusion

We described a system which enhances the effectiveness of the WWW Search Engines by using relevance feedback and relational learning. The main function of our system is the application of decision rules which is constructed by relational learning technique. We presented its representation and learning algorithm. Then we evaluated their effectiveness through retrieval experiments. The results showed that our system enables us to find more relevant pages though the effect differs in every questions. Our system needs quick response and moderate machine power. Thus it should be a user side application because search engines cannot afford to attach such a function. One of the future problem is to reduce the cost which users need to judge pages. We plan to apply clustering methods for this problem.

References

1. Baeza-Yates, R. and Ribeiro-Neto, B.: Modern Information Retrieval: Addison-Wesley, Wokingham, UK, (1999)
2. Furnkranz, J.: Separate-and-Conquer Rule Learning, *Artificial Intelligence Review*, Vol.13, No.1 (1999)
3. M. Okabe and S. Yamada: Interactive Document Retrieval with Relational Learning, *Proc. of the 16th ACM Symposium on Applied Computing*, pp.27-31 (2001)
4. Quinlan, J.R., and Cameron-Jones, R.M.: Induction of Logic Programs: FOIL and Related Systems, *New Generation Computing*, Vol.13, Nos.3,4, pp.287-312 (1995)
5. Salton, G. and Buckley, C.: Improving Retrieval Performance by Relevance Feedback, *Journal of the American Society for Information Science*, Vol.41, No.4, pp.288-297 (1990)