Available online at www.sciencedirect.com
SCIENCE DIRECT®
Knowledge-Based Systems 18 (2005) 117–124
Knowledge-Based SYSTEMS
www.elsevier.com/locate/knosys

Learning filtering rulesets for ranking refinement in relevance feedback

Masayuki Okabe[a], Seiji Yamada[b,*]

[a]*Toyohashi University of Technology, 1-1 Tempaku, Toyohashi, Aichi 441-8580, Japan*
[b]*National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda, Tokyo 101-8430, Japan*

Received 14 November 2002; accepted 6 April 2004
Available online 18 January 2005

Abstract

In this paper we propose an approach for refining a document ranking by learning filtering rulesets through relevance feedback. This approach includes two important procedures. One is a filtering method, which can be incorporated into any kinds of information retrieval systems. The other is a learning algorithm to make a set of filtering rules, each of which specifies a condition to identify relevant documents using combinations of characteristic words. Our approach is useful not only to overcome the limitation of the vector space model, but also to utilize tags of semi-structured documents like Web pages. Through experiments we show our approach improves the performance of relevance feedback in two types of IR systems adopting the vector space model and a Web search engine, respectively.

Keywords: Filtering rulesets; Ranking refinement; Relevance feedback

1. Introduction

As seen in the development of knowledge management systems or WWW search engines, Information Retrieval (IR) is one of the core technologies to exploit valuable knowledge from huge electric documents stored in digital libraries or the Internet. Since users have a wide range of objectives, IR systems need to be able to change their behaviors according to personal requests, goals and interests. The most difficult problem to achieve such adaptive behavior is to translate a user's ambiguous requirements into a concrete query. As many researches have pointed out so far, it is usually difficult for novices to input a set of keywords, which focuses on a specific topic they want to find—selecting good keywords or finding effective keyword combinations is not an easy task. Some of general search engines recently support for users to input additional keywords by showing several relational words, which are prepared by analyzing the history of transactions stored in their databases. This function, however, relatively works well if a user is searching a major topic like 'Linux for macintosh'. However, for more specific topic like 'Linux drivers for the wireless LAN card on macintosh', they do not suggest any alternative or additional keywords.

Expressing search intents are a problem of how to characterize documents a user wants to find. Such a problem has been considered mainly under the research of relevance feedback [1], which is the most successful framework of improving retrieval effectiveness with iterative human help. Since a user has only to input an initial query and mark some relevant documents during a retrieval process, this technique can reduce users effort and enables them to find a cluster of topically related documents efficiently. In most cases, relevance feedback is realized by using the vector space model because of its simplicity and effectiveness. However, it has been pointed out that the vector space model cannot represent logical and proximity word relations which are useful to identify a document precisely [2]. Although there are several retrieval models which can adopt word relations [3,4], they are not so powerful that they can become alternative ways to the vector space model.

Meanwhile, recently many of machine learning techniques have been tested in document categorization problems where each document is automatically

* Corresponding author. Fax: +81 342 122 562.
E-mail addresses: okabe@mc.tut.ac.jp (M. Okabe), seiji@nii.ac.jp (S. Yamada).

0950-7051/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.knosys.2004.04.011

categorized into pre-defined classes, and revealed to be superior to classify documents. In spite of their potential, none of them has been applied to relevance feedback so far. It is not clear that those machine-learning techniques can improve the effectiveness of traditional relevance feedback based on the vector space model. In order to investigate this point, we extend a process of relevance feedback by incorporating a rule-based document filtering. Document filtering is based on a set of rules which describes the effective propositional and proximity word relations to identify documents a user wants to find.

In short, the aim of this paper is to propose a method for improving the performance of a traditional method of relevance feedback by applying a document filtering method. Through experiments we show our approach improves the performance of relevance feedback in two types of IR systems which adopt the vector space model and a Web search engine.

The reminder of the paper is organized as follows. In Section 2 we first describe a traditional method of relevance feedback which adopts the vector space model as an IR model. Then, we incorporate additional procedures of document filtering into the process of a traditional method. Section 3 explains a ruleset—representation and its learning algorithm. Section 4 presents experiments to evaluate the effectiveness of our method. In Section 5 we analyze several rulesets. Finally, Section 6 concludes our work.

2. Relevance feedback with rule-based ranking refinement

2.1. To overcome the limitation of standard relevance feedback

Relevance feedback [5] is a framework for efficiently collecting relevant documents by repeating human evaluation and query modification. Though several retrieval models have been applied to this framework, the vector space model is one of the best models to make a practical system because of its simplicity and effectiveness. In this model, a query is modified based on the following formula

$$Q_{i+1} = Q_i + \frac{1}{N^+}\sum_{j=1}^{N^+} D_j^+ - \frac{1}{N^-}\sum_{j=1}^{N-} D_j^- .$$

Once a retrieval system returns a document ranking, a user evaluates some part of the result and can find N^+ numbers of relevant documents and N^- numbers of non-relevant ones. An IR system automatically updates a query vector Q_i using those documents—each document is represented by D_j^+ (if relevant) and D_j^- (if non-relevant) in the above formula. $D_j^+(D_j^-)$ is a vector whose components are the weight (such as $tf{\cdot}idf$ value) of all the words included in a document.

Many researches have tried to improve the performance of this relevance feedback so far by proposing methods for calculating word weights [6] and selecting good training examples [7]. Although some good weighting methods have been proposed [8], the vector space model has a critical limitation that it cannot utilize any word relation. To overcome this limitation, we considered exploiting useful word relations.

Word relations such as simple and–or relation, order in a document, proximity between words and so on are very useful to characterize a document. Several researches in the field of machine learning have recently demonstrated their usefulness in the task of document filtering, which is a process of filtering through large stores of textual data, such as categorizing Web pages into a pre-defined classes, or selecting a set of specific news or e-mail from a stream of constantly arriving data (e.g. net-news or mail magazine) [2,9–11].

Following these researches, we use boolean and proximity relation to improve the performance of standard relevance feedback. However, we cannot adopt their inductive learning algorithm directly because they assume a large amount of training data sets, which cannot be obtained in relevance feedback environment. Thus, we developed a learning algorithm, which is suitable to exploit word relations in the process of relevance feedback. We describe it in Section 4. Before describing our learning algorithm, we explain how to apply the knowledge learned by our algorithm.

2.2. Ranking refinement with rule-based filtering

Our approach for utilizing word relations is to incorporate additional procedures to exploit and to apply word relations into standard relevance feedback. This extension does not lose any advantage of standard method. Fig. 1 shows the outline of our relevance feedback model. Following the explanation of this figure, we introduce our approach.

1. *Initial search*. At the beginning of a retrieval a user inputs a query (usually a few words) to an IR system. Setting an appropriate initial query is an important factor of relevance feedback because an initial query is a basis of following relevance feedback processes, and it affects the total effectiveness of the process.
2. *Evaluation and feedback*. If a user cannot find enough relevant documents from the result of initial search, he/she starts a relevance feedback process by marking relevancy of each document he/she has already checked. A system stores those documents as *training documents* for learning of rulesets.
3. *Analysis of training documents*. In this stage, a system analyzes training documents to make statistical analysis for query modification and to extract effective word

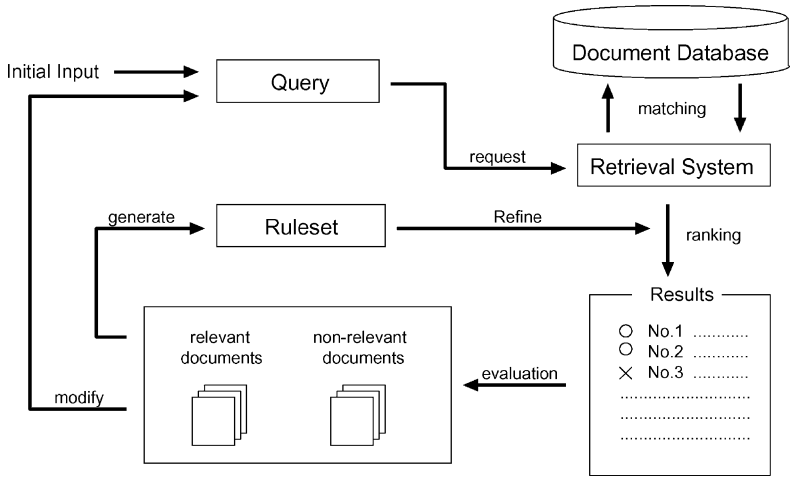

Fig. 1. Relevance feedback with ruleset filtering.

relations for a ruleset. We explain more concrete procedures in the following.

- *Expansion of keywords*. Keywords inputted by a user at initial search are too few to modify a query and to make a ruleset. Thus, a system selects additional keywords from training documents. There are several methods for the selection [12,13]. We do not care about which method to use here.
- *Extraction of word relations*. Using expanded keywords, a system prepares a set of word relations appearing in training documents. These relations are used to make rulesets.

4. *Learning rulesets*. After analyzing training documents, a system constructs a ruleset by our learning algorithm. We explain the learning algorithm in detail in Section 3.
5. *Query modification and next search*. A system modifies a query using expanded keywords, and then makes a new document ranking using the modified query.
6. *Ranking refinement*. Before showing a new document ranking, a system refines the ranking using a ruleset. Refinement procedure is simple. As shown in Fig. 2, a system checks a document from top of the ranking. Only if a document is identified as non-relevant by a ruleset, the document is moved to the lowest rank of the ranking. After all, this procedure classifies documents with the preservation of the order of each document ranking. This refined ranking is shown to a user. If he/she does not satisfy the ranking, he/she repeats from the evaluation procedure of this process.

This process can be used not only to overcome the drawbacks of traditional relevance feedback, but to improve the performance of many kinds of IR systems.

3. Learning rulesets

In Section 2, we described an overview to learn and apply a ruleset in the process of relevance feedback. Here, we explain description of a ruleset and a learning algorithm in detail.

3.1. Rule representation

A ruleset is a set of rules, each of which is represented in the form of Horn clause. A rule consists of two parts—a *head* part and a *body* part. For all the rules, a head part is represented in a unique form, 'rel(A):-…' (…is a body part). It means a document A is relevant if it satisfies a condition described in a body part. A body part is a conjunction of the following two types of literals.

ap(A, w_i) This literal is true *iff* a word w_i appears in a document A.

near(A, w_i, w_j) This literal is true *iff* both words w_i and w_j appear within a sequence of n words somewhere in a document A. We do not care about the appearance order of those two words.

Each literal describes a feature, which is useful to characterize a document. Combining several features, a rule provides requirements for identifying a document as relevant. In general, as the number of documents to be covered increases, we need more rules if we want to specify

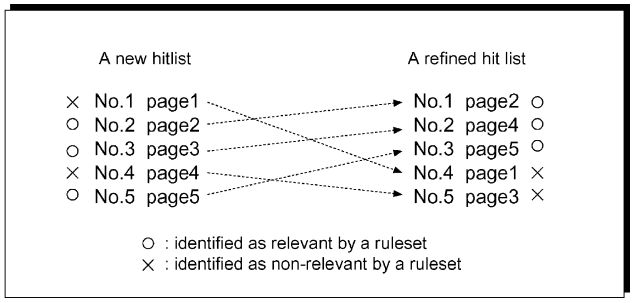

Fig. 2. Ranking refinement.

each document precisely. Thus, a ruleset usually consists of several rules like below]

$$\begin{cases} \text{rel}(A) : \text{-ap}(A, \text{mammal}), \text{near}(A, \text{species}, \text{protect}). \\ \text{rel}(A) : \text{-ap}(A, \text{species}), \text{near}(A, \text{mammal}, \text{protect}). \end{cases}$$

This ruleset means that a document A is relevant if a word 'mammal' appears in A, and 'species' and 'protect' closely appear in A, or 'species' appears in A, and 'mammal' and 'protect' closely appear in A.

3.2. An extension for semi-structured documents

Information resources are not only plain texts, but semi-structured ones like Web pages, which are decorated with a set of tags (e.g. a tag set of HTML). Since some of them work well as features to discriminate documents [14,15], we extend above literals by adding an argument of *region*, which specifies a position of a literal in a document with several kinds of tags. The meaning of each tag is slightly changed as follows.

ap(A, region, w_i) This literal is true *iff* a word w_i appears within a region of *region* in a document A.

near(A, region, w_i, w_j) This literal is true *iff* both of words w_i and w_j appear within a sequence of n words somewhere in a region of *region* of a document A. We do not care about the appearance order of those two words.

For example, we can choose the following tags which can be instances of *region* for Web pages.

- *title*. This specifies an area surrounded by <TITLE> and </TITLE> tags.
- *anchor*. This specifies an area surrounded by <A HREF=…> and </A>.
- *head*. This specifies an area surrounded by <Hn> and </Hn> tags (n=1–4).
- *para*. This specifies an area surrounded by <P> and </P> tags.

We can obtain the following ruleset with extended literals

$$\begin{cases} \text{rel}(A) : \text{-ap}(A, \text{title}, \text{mobile}), \text{ap}(A, \text{anchor}, \text{PDA}). \\ \text{rel}(A) : \text{-near}(A, \text{para}, \text{palm}, \text{os}). \end{cases}$$

This ruleset means that a Web page is relevant if a word 'mobile' appears in the title and 'PDA' appears in an anchor text, or 'palm' and 'OS' closely appear in the same paragraph.

3.3. Learning algorithm

Fig. 3 shows our learning algorithm to make a ruleset. This algorithm generates a set of rules under the separate-and-conquer strategy [16]. According to this strategy, one rule is made at a time. When a rule is generated, it is added to a rule set R, and relevant documents covered with this rule are removed from E^+. This procedure continues until

Input
  C : background literals, T : truth value table,
  E^+ : relevant documents, E^- : non relevant documents
Output
  R : a set of query rule
Variable
  rule : a query rule, S : exception literals
Initialize
  $R \leftarrow empty$, $S \leftarrow empty$, $rule \leftarrow rel(A)$:-
Repeat
  if *rule* exclude all the documents in E^- **then**
    add *rule* to R and remove documents covered by *rule* from E^+
    if E^+ is *null* **then** *exit* **else** initialize *rule* and S
  else
    for each literal in C except literals in S, calculate *information gain* G
    if no literal such as $G > 0$ **then** *exit*
      if no literal appended to *rule* **then** *exit*
      else
        initialize S, append the first literal in the body of *rule* to S, and initialize *rule*
    else
      add the most gainful litaral to *rule* and S

Fig. 3. Learning algorithm.

E^+ becomes empty. A rule starts from an empty conjunction of conditions, and develops by adding a gainful literal one after another until it excludes all the elements of E^-. Gainful literals are selected from a set C of candidate literals, which are prepared by inserting keywords into the two types of features. The procedure to prepare literals is as follows.

1. For each of keywords, make an ap literal.
2. For each combination of keywords, make a near literal.
3. Examine these literals are true or not in each training document and record those information.

For a brief example, if there are the following keywords

{mammal, species, protect},

this algorithm first prepares a set of literals like below

$$\left\{ \begin{array}{l} \mathrm{ap}(A,\ \mathrm{mammal})\ \mathrm{near}(A,\ \mathrm{mammal},\ \mathrm{species}) \\ \mathrm{ap}(A,\ \mathrm{species})\ \mathrm{near}(A,\ \mathrm{species},\ \mathrm{protect}) \\ \mathrm{ap}(A,\ \mathrm{protect})\ \mathrm{near}(A,\ \mathrm{protect},\ \mathrm{mammal}) \end{array} \right\},$$

The most gainful literal has the highest value of information gain calculated by the following formula [17]

$$G = e^{\oplus}_{\mathrm{after}}\{I(e^{\oplus}_{\mathrm{before}}, e^{\ominus}_{\mathrm{before}}) - I(e^{\oplus}_{\mathrm{after}}, e^{\ominus}_{\mathrm{after}})\},$$

$$I(e^{\oplus}, e^{\ominus}) = -\log_2 \frac{e^{\oplus}}{e^{\oplus} + e^{\ominus}},$$

The $e^{\oplus}_{\mathrm{before}}, e^{\ominus}_{\mathrm{before}}, e^{\oplus}_{\mathrm{after}}, e^{\ominus}_{\mathrm{after}}$ are, respectively, the numbers of relevant and non-relevant documents covered with the before and after appending a literal.

Rule construction using information gain is efficient because it is basically greedy search. However, in this type of search, it sometimes selects a bad literal and stops before completion. Thus, we incorporated backtracking mechanism into this algorithm. Backtracking is a mechanism to avoid deadlock of search. In our algorithm deadlock occurs when there is no literal to be selected on the way of rule construction. In such a case, this algorithm stops the present search and restart from the first literal selection. The point here is that our algorithm restarts from the first literal selection. Normal backtraking restarts from one previous step selection. However, this method often makes a futile search because we empirically observed the importance of the first literal selection is much higher than the second, third or the other ones. Thus, our backtracking method can avoid much redundant search compared to the normal backtracking.

4. Experiments

We evaluated our method in two types of IR systems. One is a traditional IR system, which uses the vector space model as a retrieval model and a newspaper collection as database. The other is a Web search engine. We use this as a representative of systems, which retrieve semi-structured documents. In both experiments, relevance feedback was conducted following the procedures described in Section 2.

4.1. Experiment 1: IR with a traditional system

4.1.1. Settings

In this experiment, a query is a vector of word weights. Query words are initially set to five words, and 15 words are automatically added after first feedback. After every feedback, those 15 words are replaced by a set of new 15 words selected from training documents with a simple query expansion method [18]. Word weights are calculated by a method which is proven to be quite effective in the framework of the vector space model [6], and are modified by the formula shown in Section 2 after every feedback.

As a document database, we used a set of newspaper articles (The Los Angels Times, about 130,000 articles, ave. 526 words/article), which is a part of TREC-7 collection [19]. This collection provides topics (subjects of retrieval, or test questions) and relevance judgments (lists of relevant documents for each subject). We selected 20 topics in which each has more than 40 relevant documents because relevance feedback is usually used to collect not a few relevant documents.

In order to evaluate the effectiveness of our approach, we tested two types of relevance feedback. One is a feedback based only on query reformulation (we call this VSM). The other is a feedback based not only on query reformulation, but on filtering with a ruleset (we call this RULE). In each cycle, we carried out four iterations of feedback. At each feedback, top 10 documents were used as training documents (except for the documents which had already been seen). Training documents were accumulated for the next feedback to be used to create a new query and a new ruleset.

4.1.2. Results

Fig. 4 shows the average performance of 20 topics. We conducted four feedbacks about one topic. The horizontal line means the number of feedbacks and the vertical line means the average of the three-point average precision for 20 topics. Three-point average precision is the mean of precisions at recall 0.25, 0.5, 0.75, where *recall* is defined

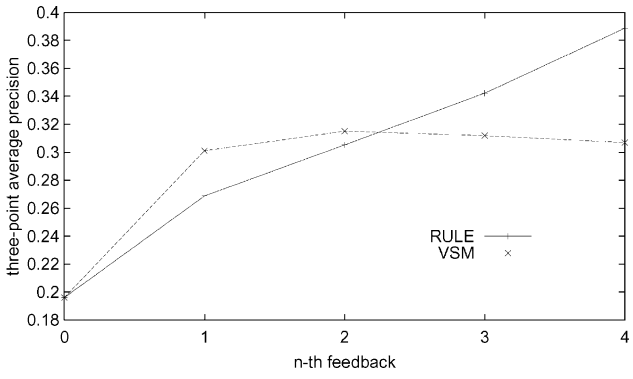

Fig. 4. Average performance of 20 topics.

as the proportion of relevant documents that are retrieved from the collection, and *precision* is the proportion of retrieved documents that are relevant. This value is often used to evaluate the total performance of retrieval methods.

The effects of RULE are rising after the third feedback because we cannot usually get enough relevant documents which are necessary to create a good ruleset at an early stage of feedback. Comparing the third results with the fourth ones, RULE keeps improving the performance though VSM has no improvements as seen in the graph. Actually, it is testified in another experiments that RULE can keep improvements much longer than VSM. In addition, RULE can improve the performance of a certain topics which VSM cannot improve.

Table 1 compares three-point average precisions of the rankings that each of two systems VSM and RULE produced after nth feedback ($n=1$–4). In the table, TOPIC corresponds to the topic number provided by the TREC. INI is the results of the initial retrieval. The effectiveness of RULE for a topic considerably differs from each other. For more than half of topics, RULE improves retrieval performance compared to VSM. Though some topics have no improvements or even worse results, those effects are much less than the improvements of other topics.

Table 2 shows another values which are the number of relevant documents included in training documents, If a user conducts the procedures described in Section 2, these number correspond to the number of relevant documents which he/she actually found during a retrieval process. Different from the results of threes-point average precision, RULE can get more relevant documents than VSM in almost all the situation.

Table 1
Comparison of VSM and RULE (three-point average precision)

Topic	INI	VSM				RULE			
		1	2	3	4	1	2	3	4
301	0.36	0.56	0.57	0.52	0.53	0.53	0.55	0.68	0.67
304	0.11	0.08	0.11	0.12	0.11	0.10	0.24	0.32	0.35
306	0.09	0.01	0.02	0.03	0.02	0.01	0.11	0.03	0.06
315	0.04	0.03	0.05	0.04	0.05	0.04	0.05	0.04	0.05
332	0.17	0.39	0.43	0.45	0.45	0.38	0.39	0.71	0.72
335	0.73	0.78	0.78	0.79	0.78	0.65	0.58	0.73	0.84
343	0.10	0.04	0.07	0.07	0.07	0.04	0.08	0.12	0.11
349	0.10	0.54	0.49	0.52	0.48	0.64	0.42	0.56	0.70
354	0.14	0.06	0.05	0.04	0.04	0.11	0.06	0.07	0.09
358	0.38	0.42	0.36	0.35	0.36	0.41	0.37	0.37	0.55
360	0.20	0.29	0.37	0.38	0.32	0.18	0.55	0.47	0.36
366	0.01	0.30	0.38	0.31	0.38	0.08	0.13	0.10	0.33
367	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
368	0.53	0.74	0.63	0.51	0.45	0.80	0.83	0.50	0.85
370	0.04	0.04	0.01	0.06	0.06	0.04	0.01	0.11	0.05
374	0.08	0.25	0.39	0.40	0.41	0.07	0.43	0.47	0.39
383	0.04	0.02	0.06	0.12	0.11	0.02	0.06	0.14	0.05
390	0.26	0.69	0.66	0.61	0.60	0.72	0.68	0.67	0.65
393	0.03	0.05	0.26	0.22	0.27	0.01	0.06	0.08	0.24
398	0.51	0.70	0.60	0.69	0.65	0.55	0.52	0.67	0.72
Ave.	0.20	0.30	0.32	0.31	0.31	0.27	0.31	0.34	0.39

Table 2
Comparison of VSM and RULE (the number of relevant documents in training sets)

Topic	INI	VSM				RULE			
		1	2	3	4	1	2	3	4
301	4	9	13	15	17	8	16	22	28
304	3	7	8	8	9	6	11	12	13
306	1	2	2	2	2	2	3	3	5
315	2	2	3	3	3	2	3	3	4
332	6	16	25	34	43	16	25	35	45
335	9	19	26	32	34	19	27	32	35
343	2	3	4	4	4	5	7	8	11
349	4	13	16	17	18	13	17	19	21
354	2	7	10	11	11	11	13	16	21
358	8	13	14	14	14	13	16	20	23
360	6	11	14	14	14	12	17	19	19
366	3	9	11	12	14	5	8	9	14
367	0	0	0	1	1	0	0	1	1
368	8	17	21	22	22	18	26	30	35
370	0	0	4	5	6	0	4	6	6
374	5	10	12	15	19	11	16	22	24
383	2	2	7	9	11	3	5	8	12
390	6	16	23	27	29	16	23	28	32
393	1	8	13	15	16	9	11	12	16
398	6	16	22	25	26	15	17	23	27
Ave.	3.9	9.0	12.4	14.2	15.7	9.2	13.2	16.4	19.6

4.2. Experiment 2: IR with a web search engine

4.2.1. Settings

In this experiment, we use a Web search engine as IR system in Fig. 1. Since we have no standard IR model in this case, we compare the quality of original rankings (hit lists) with refined ones, which is filtered by our rulesets. As in the previous experiment, we used 20 topics which are parts of test questions in the TREC-8 Small Web Track [20], and manually judged 50 Web pages from top-ranked ones. To make a refined ranking, we remake a ruleset which is used to filter the rest of documents in the original ranking after finishing judgment of 10 Web pages. Thus we make a new ruleset four times in each cycle. A retrieval cycle with no filtering, namely, a cycle of checking an original hit list is called WSE. The other cycle, which iteratively makes and uses a ruleset, we call this cycle RULE. We used the Google search engine[1] as a test Web search engine, which is recognized as one of the most powerful search engines. Fig. 5 shows an interface which mediates for a search to interact with a Web search engine. This interface provides several functions like check buttons for marking the relevancy of each item in a hit list. It also makes a ruleset based on the above marking, and filter a hit list returned by a Web search engine.

4.2.2. Results

Table 3 compares the number of relevant documents which are found after nth feedback in each cycle of WSE

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

Fig. 5. Interface for relevance feedback on the Web.

and RULE. TOPIC and INI indicates the same meaning as in the previous experiment.

The difference in the number of relevant pages constantly increases after the first feedback for every topic as seen in Fig. 6. Moreover RULE is superior to WSE in all the situations though the performance differs for each topic. As an average result, RULE got about five relevant Web pages more than WSE after the fourth feedback (after finishing the marking of 50 Web pages).

5. Discussion

As seen in the results of the previous two experiments, the retrieval with our system enhanced the effectiveness for most topics. We show two types of examples, a good one

Table 3
Comparison of WSE and RULE

Topic	INI	WSE				RULE			
		1	2	3	4	1	2	3	4
401	2	6	11	13	13	6	10	15	22
402	6	8	8	8	9	8	9	14	19
403	3	10	17	23	31	11	18	26	35
404	3	5	7	10	15	7	10	11	15
405	2	4	6	6	6	3	6	7	10
406	7	14	20	28	32	14	21	29	35
407	2	3	7	13	19	9	13	20	27
408	6	11	14	19	24	12	19	25	28
409	7	10	11	14	15	10	13	15	22
410	4	7	10	14	20	9	13	15	21
411	1	8	15	15	19	7	10	14	14
412	3	8	9	11	14	9	16	22	27
413	6	9	12	16	18	9	12	16	19
414	4	8	9	10	11	6	12	17	21
415	8	13	15	23	27	11	17	22	28
416	7	12	15	23	27	14	18	24	29
417	5	9	10	12	15	10	15	17	20
418	4	9	11	14	15	9	14	23	28
419	5	10	13	14	16	10	13	17	18
420	5	12	17	23	29	12	19	27	35
Ave.	4.5	8.8	11.9	15.5	18.8	9.3	13.9	18.8	23.7

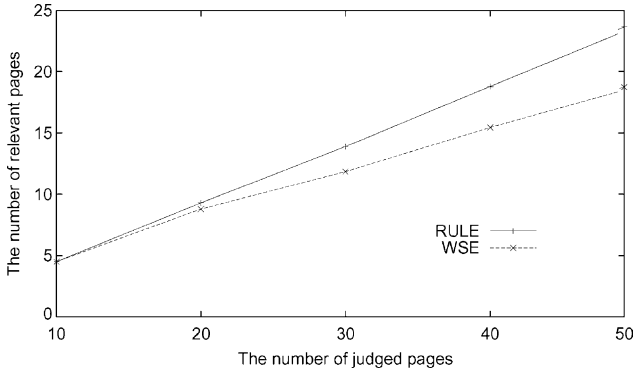

Fig. 6. The number of relevant pages.

that our system effectively worked, and a bad one that our system did not work well.

> "Income Tax Evasion
> This query is looking for investigations that have targeted evaders of US income tax…"

This is a precise description of no. 332 topic used in the first experiment. Relevant documents for this topic include words like 'income', 'tax', 'evasion' 'fraud', 'court', 'trial', 'illegal' and so on. Those words, however, appear frequently in so many and various types of documents. In VSM, many documents about 'fraud crime' and 'illegal crime' which are not related to this topic are retrieved. In contrast, RULE carefully selected documents using a ruleset shown in Table 4. This ruleset emphasizes proximity relations among words, which actually filter out much non-relevant documents.

In the second experiment, topic no. 12 shows very good results. The description of this topic is as follows.

> "Airport Security
> What security measures are in effect or are proposed to go into effect in airports?…"

The objective of topic 412 is 'to identify a specific airport and describe the security measures already in effect or proposed for use at that airport'. WSE returns many non-relevant Web pages which introduce 'the security, which travelers must prepare,'. In contrast, RULE carefully selected documents using a ruleset shown in Table 5. This ruleset emphasizes single words by tags, and utilizes proximity relations to specify the meaning of words. For example, 'faa' and 'system' in the fourth rule are very abstract words in general. Considering together, however, 'faa' indicates the federal aviation administration in

Table 4
A ruleset for topic no. 332

```
rel(A):- near(A, charge, hunter), ap(A, count)
rel(A):- near(A, income, evasion)
rel(A):- ap(A, evasion), near(A, tax, hunter)
rel(A):- near(A, evasion, convict), ap(A, illegal)
rel(A):- near(A, convict, charge)
```

Table 5
A ruleset for topic no. 412

rel(*A*):- ap(*A*, anchor, screening)
rel(*A*):- near(*A*, para, security, system), ap(*A*, title, airport)
rel(*A*):- near(*A*, para, security, airports), near(*A*, para, security, access)
rel(*A*):- near(*A*, para, security, airports), near(*A*, para, faa, system)

Table 6
A ruleset for topic no. 411

rel(*A*):- ap(*A*, anchor, shipwreck)
rel(*A*):- ap(*A*, anchor, shipwreck), ap(*A*, anchor, salvaging)

the United State, and 'system' indicates a system about aviation, especially security system in airports.

As these examples show, rulesets work very well if the meanings of keywords can be straiten by pairing two words.

In spite of the usefulness of rulesets, we must care about some noisy rules, which may sometimes be built. A ruleset shown in Table 6 is an example ruleset that did not work well. This ruleset is built for topic no. 411 which has the following description.

> "salvaging, shipwreck, treasure
> Find information on shipwreck salvaging: the recovery or attempted recovery of treasure from sunken ships…"

Relevant pages for this topic include various types of pages such as links, bulletin board, news and individual home pages. This ruleset is too general to select pages appropriately because it uses only two keywords which are generally insufficient to restrict relevant pages.

6. Conclusion

We proposed an approach for refining a document ranking by learning filtering rulesets through relevance feedback. Relevance feedback can be regarded as a process of producing a refined document ranking. Our approach directly refines a document ranking by removing non-relevant documents based on a ruleset. Experiments show that our learning algorithm can extract useful word relations, which are effective to identify relevant documents accurately. Analyzing rulesets learned through relevance feedback, we confirmed how each rule restricts relevant documents. We also showed that the possibility to apply our approach into relevance feedback with Web search engines. To utilize tags more effectively is future work.

References

[1] G. Salton, C. Buckley, Improving retrieval performance by relevance feedback, Journal of the American Society for Information Science 41 (1990) 288–297.

[2] W.W. Cohen, Text categorization and relational learning, in: Proceedings of the Twelveth International Conference on Machine Learning, Morgan Kaufmann, Los Altos, CA, 1995, pp. 124–132.

[3] G. Salton, E.A. Fox, E. Voorhees, Advanced feedback methods in information retrieval, Journal of the American Society for Information Science 36 (3) (1985) 200–210.

[4] D. Haines, W.B. Croft, Relevance feedback and inference networks, in: Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 1993, pp. 2–11.

[5] J.J. Rocchio, Relevance feedback in information retrieval, in: G. Salton (Ed.),, The SMART Retrieval System: Experiments in Automatic Document Processing, Prentice-Hall, Englewood Cliffs, NJ, 1971, pp. 313–323.

[6] C. Buckley, The importance of proper weighting methods, in: Proceedings of ARPA Human Language Technology Workshop'93, Morgan Kaufmann, Los Altos, CA, 1994, pp. 349–352.

[7] M. Mitra, A. Singhal, C. Buckley, Learning routing queries in a query zone, in: Proceedings of the Twentieth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 1997, pp. 25–33.

[8] C. Buckley, G. Salton, Optimization of relevance feedback weights, in: Proceedings of the Eighteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 1995, pp. 351–357.

[9] J. Callan, Document filtering with inference networks, in: Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 1996, pp. 262–269.

[10] T. Joachims, Text categorization with support vector machines: learning with many relevant features, in: Proceedings of the Tenth European Conference on Machine Learning, vol. 1398 of LNAI, Springer, Berlin, April 21–23 1998.

[11] D.D. Lewis, An evaluation of phrasal and clustered representations on a text categorization task, in: Proceedings of the Fifteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 1992, pp. 37–50.

[12] J. Xu, W.B. Croft, Query expansion using local and global document analysis, in: Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 1996, pp. 4–11.

[13] A. Singhal, M. Mitra, C. Buckley, Improving automatic query expansion, in: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Query and Profile Modification, 1998, pp. 206–214.

[14] W.W. Cohen, W. Fan, Learning page-independent heuristics for extracting data from Web pages, Computer Networks (Amsterdam, Netherlands: 1999) 31 (11–16) (1999) 1641–1652.

[15] M. Okabe, S. Yamada, Interactive web page filtering with relational learning, in: Proceedings of the First Asia-Pacific Conference on Web Intelligence, 2001, pp. 443–447.

[16] J. Fürnkranz, Separate-and-conquer rule learning, Artificial Intelligence Review 13 (1) (1999) 3–54.

[17] J.R. Quinlan, R.M. Cameron-Jones, Induction of logic programs: FOIL and related systems, New Generation Computing, Special issue on Inductive Logic Programming 13 (3/4) (1995) 287–312.

[18] M. Okabe, S. Yamada, Interactive document retrieval with relational learning, in: Proceedings of Sixteenth ACM Symposium on Applied Computing, 2001, pp. 27–31.

[19] E. Voorhees, D. Harman, Overview of the Seventh Text REtrieval Conference, 1998.

[20] E. Voorhees, D. Harman, Overview of the eighth Text REtrieval Conference, 1999.