Interactive Document Retrieval with Relational Learning

Masayuki Okabe and Seiji Yamada
CISS, IGSSE, Tokyo Institute of Technology
4259 Nagatuta Midori-ku
Yokohama, JAPAN
{okabe, yamada}@ymd.dis.titech.ac.jp

Keywords
information retrieval, relevance feedback, relational learning

ABSTRACT
This paper describes an approach to enhance the effectiveness of human help in interactive document retrieval, where the system supports the user to find documents effectively through relevance feedback. At present vector space model is a typical representation method to realize relevance feedback. However it can neither express relationship such as proximity nor keep several features separately. We supplement these defects with a set of rules, which are constructed by relational learning and used to identify relevant documents. The learning algorithm consists of separate-and-conquer strategy and top-down heuristic search with limited backtracking. Background relations are made only from keywords, thus constructed rules represent useful keyword combinations to search relevant documents. We evaluate the effectiveness of our approach on document retrieval experiments using a test bed database. The results show our method enhances both effectiveness and efficiency compared to a normal method with only query vector.

1. INTRODUCTION
Document retrieval is an essential part of an IR (Information Retrieval) system which has recently become an indispensable tool to utilize a large number of electronic documents. One of the important requirements for the system is effective search through human computer interaction because of the limitation of fully automatic IR. Such approach has been studied under relevance feedback[9], which is a commonly accepted process of improving retrieval effectiveness with human help. The system with relevance feedback usually repeats retrievals until the user satisfies his information need. After each retrieval, he is asked to judge a small part of the results and indicate relevant documents from them. These documents will be used to extract valid information so as to return newly better results. Since the effectiveness of relevance feedback significantly depends on the extraction method, several ones have been proposed[5, 10]. However they are based on statistical ways, which hardly extract structural patterns such as the positioning of the words in documents. Recently machine learning techniques has been tested in IR and revealed to be hopeful for more effective IR.[1, 2, 3]

In this paper, we propose an effective extraction method using relational learning. Relational learning is a machine learning technique developed mainly in ILP(Inductive Logic Programming)[7], which is superior to learn relational patterns such as protein secondary structure. We applied this technique to the extraction of characteristic document structures. They are represented in the form of logical rules which consist of propositional and proximity relations[6] among words in documents. We use these rules in order to decide whether a document is relevant to our need or not, thus we call them *decision rule*. Our learning algorithm constructs as many rules as possible until they cover all the relevant documents. This strategy is very effective since every relevant document doesn't have the same feature. To verify the structural relation is promising information for precise IR, we fully implemented our IR system which integrated relevance feedback and relational learning. Then we make experiments in which we applicate our approach to a traditional statistical IR system.

The reminder of the paper is organized as follows. Section 2 describes the interactive process and the way how to apply relational learning. Section 3 describes the representation and the learning algorithm for decision rule. Section 4 mentions experiments and the results using a test bed database of newspaper.

2. INTERACTIVE RETRIEVAL PROCESS
Recent high capacity hardwares allow us to manage larger document databases than before. Information retrieval from such databases becomes difficult when we want to obtain as many of relevant documents as possible. In such a situation, we usually have to iterate search by changing query in order to minimize the reviewing of documents. Interactive process is useful for such a task and relevance feedback is the most successful method until now. It can shield the user from the details of the query formulation and break down the search operation into a sequence of small search steps by utilizing human feedbacks[11].

SAC 2001, Las Vegas, NV

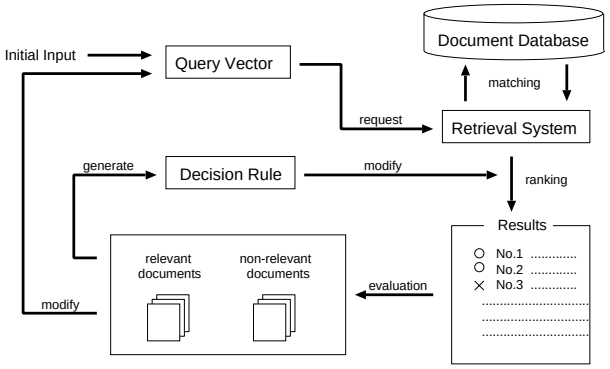

Figure 1: Interactive retrieval process

Relevant feedback was developed with vector type query(we call this *query vector*) and has been one of the best match representations. It is a popular retrieval model nowadays, thus we use it as a basic interactive retrieval model and integrate our *decision rule* into this model. Figure 1 shows the model. Here we describe the outline of the process. For precise procedures are explained in Section 4. At first the user inputs initial query(generally a few words) into the system. Then the system makes a vector whose elements are the weights of input words. With that vector, the system ranks all the documents in the database and returns the results to the user. After the user evaluates a certain group of documents, if he is satisfied with the results, this retrieval process finishes. If not, he can ask the system to return another results by indicating each relevancy of the judged documents. According to this information, the system modifies the vector and ranks all the documents again. In addition to this process, in our model a set of decision rules is generated. As shown in Figure 2, we use them to change the document ranks attached by the vector matching. The documents which satisfy the condition of the rules have higher rank with this method.

The use of decision rule is the unique point of this research, which has not been tested before. The rules are constructed by the algorithm based on relational learning. They are composed of several rules and each rule mainly represents the characteristic positioning of keywords. We describe them in the next section.

3. DECISION RULE

This section explains decision rule in detail. We deal with the construction of the rules as learning problem, in which relevant and non-relevant documents are training examples.

3.1 Rule representation

We use horn clause to represent rules. The body of a rule consists of the following relations.

- $ap(A, w_i)$: 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 5 words somewhere in a document A. We don't consider the order of the words in *near* literal.

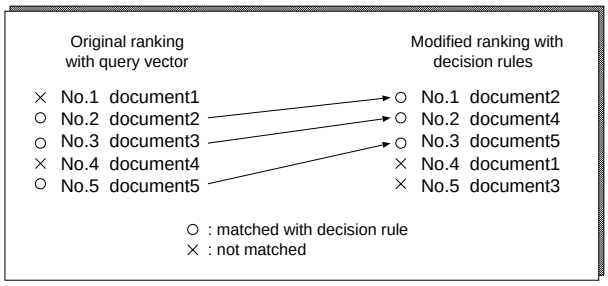

Figure 2: Ranking modification

We can represent various features of documents by combining these relations. Here is an example set of rules.

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

They mean that a document A is relevant if "*mammal*" appear in A and "*species*", "*protect*" closely appears in A, or "*species*" appears in A and "*mammal*", "*protect*" closely appears in A.

3.2 Learning algorithm

Figure 3 shows the learning algorithm for decision rule. This algorithm generates a set of rules under the separate-and-conquer strategy[4]. 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 E^+ becomes empty.

A rule starts from a empty clause 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 is prepared by inserting keywords into the two types of relations. The procedure to prepare is as follows.

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

For a brief example, if there are the following keywords,

$$\{mammal,\ species,\ protect\}$$

a set of literals like below is prepared.

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

The most gainful literal has the highest *information gain* calculated by the following formula[8].

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

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

Input
E^+ : relevant documents
E^- : non relevant documents
C : literal candidates
Output
R : a set of decision rules
Variable
rule : a rule
S : exception literals
Initialize
$R \leftarrow empty$
$S \leftarrow empty$
$rule \leftarrow rel(A)$:-
Repeat
if *rule* exclude all the documents of E^- **then**
· add *rule* to R and rsemove documents covered by *rule* from E^+
if E^+ becomes *empty* **then** *exit*
else initialize *rule* and S
else
for each literal in C except ones in S, calculate *information gain* G
if no literal such as $G > 0$ **then** *exit*
if *rule* is *empty clause* **then** *exit*
else
· initialize S
· append the first literal in the body of *rule* to S
· initialize *rule*
else
· add the most gainful litaral to *rule* and S

Figure 3: Learning Algorithm

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

Rule construction using information gain is efficient because it is greedy. However it sometimes selects bad literal and stops before completion. Thus we incorporated a backtrack mechanism into this algorithm. Backtrack occurs when there is no literal to select despite *rule* still covers non-relevant documents. For such an occasion, this algorithm tries again from the first literal selection. This method usually avoids redundant search.

4. EXPERIMENTS AND RESULTS

We made retrieval experiments for testing our algorithm according to the process described in section 2.

4.1 Test collection

We used a test collection provided by the TREC[13] as a document database. It is a popular collection used in various IR researches. From this collection, we selected a set of newspaper articles(The Los Angels Times, about 130,000 articles, ave. 526 words/article). We also used a set of topics and their judgments attached to this database. Topics are the statements described about requirements for relevant documents. We selected 20 topics from them which have more than 40 relevant documents. Judgments are the record of relevant articles for each topic.

$$L = \frac{1 + log(tf)}{1 + log(\text{average } tf \text{ in text})}$$

$$t = log(\frac{N+1}{df})$$

$$u = \frac{1}{0.8 + 0.2\frac{n_w}{\text{average } n_w \text{ per document}}}$$

n_w : number of unique words in text
tf : term's frequency in text
N : total number of relevant documents
df : number of documents that contain the term

Lnu weighting $= L * u$
Ltu weighting $= L * t * u$

Table 1: Term weights

4.2 Retrieval steps

Here are the steps involved in the use of *decision rule*.

1. *Initial search*: Create *query vector* whose elements are the weight of 5 keywords selected from the topic statement, and set the same weight to each keyword. Using this vector, rank all the documents in a database which are represented as *Lnu* weighted vectors.
2. Judging 20 documents from the top of the results, prepare a set of training documents(E^+ and E^- in section 3) for feedback.
3. Pick up 3 words from E^+ which have the top 3 highest value v calculated by the following formula.

   $$v = (\text{ average } tf \text{ in } E^+) \times (df \text{ in } E^+)$$

   tf and df are the same in Table 1. This procedure is called *query expansion* which is necessary for effective IR because the user usually doesn't know the useful words beforehand.
4. Modifies the query vector $\vec{q}$ according to the following formula[11].

   $$\vec{q}_{new} = \vec{q}_{old} + \sum_{d \in E^+} \vec{d} - \vec{d}_{not}$$

   $\vec{d}$ is a *Ltu* weighted vector made from each document in E^+. $\vec{d}_{not}$ is a *Ltu* weighted vector made from the most high ranked non-relevant document in E^-. Using modified query vector, rank all the documents.
5. According to the learning algorithm described in section 3, make a set R of *decision rules*. Using R, change the ranking of all the documents.
6. Goto step2.

Step 1 is a manual procedure and the other was automatically done by the system. *Lnu* and *Ltu* weighting methods

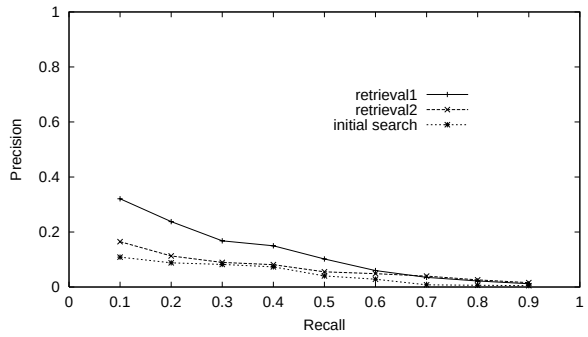

Figure 4: recall−precision curve after 1 feedback

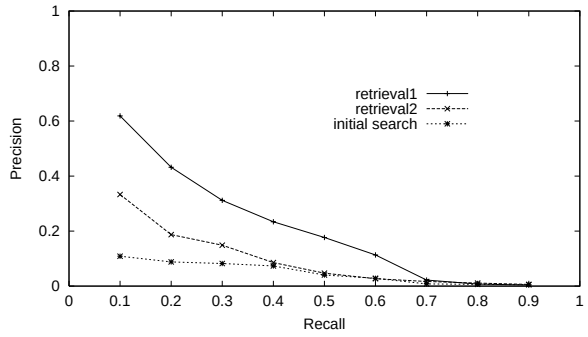

Figure 5: recall−precision curve after 4 feedbacks

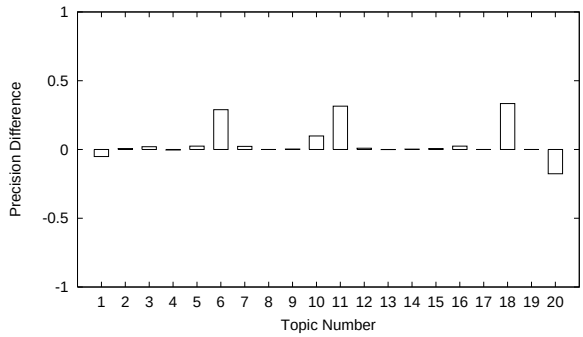

Figure 6: precision difference after 1 feedback

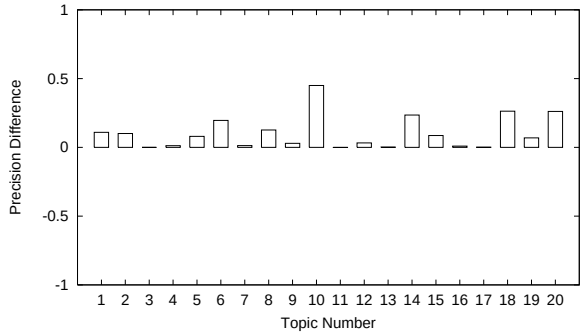

Figure 7: precision difference after 4 feedbacks

are shown in Table 1. The combination of vector and this term weighting is one of the best statistical way in modern IR[12].

In order to evaluate the effect of decision rule, we carried out two series of the above steps for each topic. The one is a retrieval **with** decision rule(act no.5 step), the other is a retrieval **without** decision rule(don't act no.5 step). In each retrieval, we repeated the above steps until 4 feedbacks(4 loops) finished.

4.3 Results

We used *recall* and *precision* measures to evaluate retrieval effectiveness, where recall is defined as the proportion of relevant documents that are retrieved from the collection, and precision is the proportion of retrieved documents that are relevant. Suppose that we examined x documents and found y relevant documents from them. If there are z relevant documents in the whole collection, recall and precision are calculated as follows.

$$Precision = \frac{y}{x}, \qquad Recall = \frac{y}{z}$$

Figure 4 and Figure 5 shows the recall−precision curves made of the results after 1 and 4 feedbacks. This curve is commonly used to evaluate the total performance of a method, in which the precision value is plotted at each recall value ranging from 0.1 to 1.0. In our experiment, the precision is the average value of 20 topics. The curve indicated by **retrieval1** shows the performance of the retrieval **with** decision rule, and **retrieval2** shows the one **without** decision rule. The criterion to evaluate the performance of the curve is its location in a graph. The curve of **retrieval1** always lie over **retrieval2** in both graphs. This means that decision rule enhances the effectiveness of relevance feedback.

Figure 6 and Figure 7 shows the performance for each topic. Let $p1$ and $p2$ be the average precision at 3 recall point(0.25, 0.5, 0.75) of retrieval1 and retrieval2 respectively, precision difference is calculated as follows.

$$Precision\ Difference\ =\ p1\ -\ p2$$

After 1 feedback, decision rule works well for only a few topics. After 4 feedbacks, however, half of topics have positive difference and no topic has negative difference.

4.4 Analysis

According to the results, it reveals that the effect of decision rule differs from every topic. In this section, we present a good example set of decision rules and a bad one which are constructed during retrievals. The following is a sample topic in which decision rule worked very well. Here is the statement of the topic.

– *"Income Tax Evasion*
This query is looking for investigations that have targeted evaders of U.S. income tax ..."

Relevant documents for this topic relatively often use the words of 'income', 'tax', 'evasion', 'fraud', 'court', 'trial', 'illegal'. However there are so many other documents using these words. Thus query vector retrieved much documents concerned with "fraud crime", "illegal crime", "supreme

```
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 2: An example set of effective decision rule

```
rel(A) :- near(A,blood,alcohol),ap(A,caus).
rel(A) :- near(A,blood,alcohol),near(A,car,kill).
rel(A) :- near(A,accident,legal).
rel(A) :- near(A,three,kill).
rel(A) :- near(A,automobil,kill).
```

Table 3: An example set of ineffective decision rule

court", most of which have no relation to this topic. In contrast, decision rule listed in Table 2 distinguished documents precisely because it makes various features matching closely to relevant documents.

The next example is a representative of bad decision rule. The statement for this topic is as follows.

– *"blood-alcohol fatalities*
What role does blood-alcohol level play in automobile accident fatalities? ..."

The rule set showed in Table 3 seems to be very effective for this topic. However the fourth rule spoils the other rules because the combination of 'three' and 'kill' is no relation to this topic. This is mainly caused by adding 'three' in automatic query expansion step. Avoiding such fail is future work.

5. CONCLUSION

We applied relational learning to document retrieval which utilizes interaction between human and computer through relevance feedback. We proposed the use of decision rule made with relational learning technique, and presented their representation and learning algorithm. We finally evaluated their effectiveness compared to a statistical method on retrieval experiments.

Our learning method constructs effective rules combined with good keywords and good relations which general users scarcely create. Our method also can be useful for searching web pages in the WWW because Web pages are structured text and seem to have useful relations. Thus we are developing an IR system using relational rules in the WWW.

REFERENCES

[1] Califf, E.M.: Relational Learning of Pattern-Match Rules for Information Extraction, *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pp.328-334 (1999)

[2] Chen, H.: Machine Learning for Information Retrieval: Neural Networks, Symbolic Learning, and Genetic Algorithms, *Journal of the American Society for Information Science*, Vol.46, No.3, pp.194-216 (1995)

[3] Cohen, W.W.: Text categorization and relational learning, *Proceedings of the Twelfth International Conference on Machine Learning*, pp.124-132 (1995)

[4] Furnkranz, J.: Separate-and-Conquer Rule Learning, *Artificial Intelligence Review*, Vol.13, No.1 (1999)

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

[6] Keen, E.M.: Some aspects of proximity searching in text retrieval system, *Journal of Information Science*, Vol.18, No.2, pp.89-98 (1992)

[7] Muggleton, S.: Inductive logic programming, *New Generation Computing*, Vol.8, No.4, pp.295-318 (1991)

[8] 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)

[9] Rocchio, J.J.: Relevance feedback in information retrieval, In *The SMART Retrieval System-Experiments in Automatic Document Processing*, Prentice Hall, Inc., pp.313-323 (1971)

[10] Salton, G., Fox, E.A., and Voorhees, E.: Advanced feedback methods in information retrieval, *Journal of the American Society for Information Science*, Vol.36, No.3, pp.200-210 (1985)

[11] 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)

[12] Singhal, A., et al.: Pivoted document length normalization, In *Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp.21-29 (1996)

[13] Voohees, E.M., Harman, D.: Overview of the Seventh Text REtrival Conference(TREC-7), In *Proceedings of the Seventh Text REtrieval Conference*, NIST Special Publication (1999)

BIOGRAPHICAL NOTES

Masayuki Okabe is a Ph.D. student in the Department of Computational Intelligence and Systems Science at Tokyo Institute of Technology. His research interests include machine learning, intelligent information retrieval.

Seiji Yamada is an Associate Professor in the Department of Computational Intelligence and Systems Science at Tokyo Institute of Technology. His research interests include artificial intelligence, planning, machine learning for a robotics, intelligent information retrieval in the WWW, human computer interaction.