The Bayesian Optimal Algorithm for Query Refinement in Information Retrieval

Yasunari Maeda, Fumitaro Goto, Hiroshi Masui, Fumito Masui and Masakiyo Suzuki

Kitami Institute of Technology, 165 Koen-cho, Kitami-shi, Hokkaido 090-8507 Japan

Summary
To realize more efficient information retrieval it is critical to improve the user's original query, because novice users can not be expected to formulate precise and effective queries. Queries can often be improved by adding extra terms that appear in relevant documents but which were not included in the original query. This is called query expansion. Query refinement, a variant of query expansion, interactively recommends new terms related to the original query. Because previous research did not offer any criterion to guarantee optimality, this paper proposes an optimal algorithm for query refinement with reference to the Bayes criterion.

Key words:
Information retrieval, query refinement, Markov decision processes, Bayes criterion

1. Introduction

These days many people use IR(Information Retrieval) systems. It is very important to improve the user's original query, because it is difficult to formulate precise and effective queries. Queries can often be improved by adding extra terms that appear in relevant documents but which were not included in the original query. This is called query expansion. There are two forms of query expansion: relevance feedback and query refinement.

In relevance feedback[6] the system interactively recommends documents related to the original query, and the user chooses the documents preferred and performs IR using the new keys. Obviously there is a trade-off between IR result accuracy and the burden placed on the user. This trade-off can be resolved using statistical decision theory[1].

In query refinement[7] the system interactively recommends new terms related to the original query, and the user chooses the terms preferred and performs IR using the new keys. The same trade-off exists between IR result accuracy and the user's burden, but no previous paper has resolved this problem theoretically. This paper applies statistical decision theory to provide the first solution to this trade-off.

Query refinement can be considered as the task of recommending related terms and search documents wanted by the user under the condition that the user's reaction in selecting related terms is unknown. More generally, it can be considered as the task of control given some unknown information. This kind of task is also being studied in the field of RL(Reinforcement Learning). Most models in the RL field employ MDP(Markov Decision Processes)[5] with some unknown information. This research also adopts MDP modeling with some unknown information for query refinement. Two RL previous algorithms that adopt MDP are described below.

QL(Q-Learning)[8] is the most famous RL algorithm and guarantees convergence to the optimal solution when the number of samples for learning is infinite. Obviously this assumption is not practical in the IR field because in query refinement the number of samples is the number of times that the system recommends related terms.

Martin considered this task based upon statistical decision theory[4]. His algorithm guarantees optimality with reference to the Bayes criterion when the number of samples is finite. However, it fails to represent the trade-off between IR result accuracy and the user's burden. Accordingly, we propose a new algorithm that can represent this trade-off and guarantee optimality with reference to the Bayes criterion when the number of samples is finite. The proposed algorithm is regarded as a generalization of Martin's algorithm. Hereafter, optimality means optimality with reference to the Bayes criterion.

We describe MDP in section 2, and Martin's algorithm in section 3. The model of our research is described in section 4. The proposed algorithm is introduced in section 5. Section 6 concludes the paper.

2. Markov Decision Processes

Query refinement can be regarded as a RL problem, and most RL models are based on MDP. An MDP combines finite states, finite actions, a transition probability matrix and a reward function. When an action is chosen, a state transition occurs depending on the transition probability matrix, and when a state transition occurs a reward occurs. The purpose of MDP is to maximize the total reward received by repeatedly choosing actions.

Generally, there are two kinds of tasks in MDP, the discounted problem and the undiscounted problem. The

Manuscript received October 5, 2011
Manuscript revised October 20, 2011

goal of the discounted problem is to maximize the expected total discounted reward which is described as follows:

$$v = E\left(\sum_{t=0}^{\infty} \beta^t z_t \right), \tag{1}$$

where β , $0 \le \beta \le 1$ is a discount factor and z_t is the reward received at time t . In the discounted problem, rewards are discounted. The goal of the undiscounted problem is to maximize the average reward which is described as follows:

$$h = E\left(\lim_{T \to \infty} \left(\sum_{t=0}^{T} z_t / T + 1 \right) \right). \tag{2}$$

3. Previous Research

In this section we explain Martin's algorithm which is closely related to our research. In Martin's algorithm the total discounted reward is maximized with reference to the Bayes criterion when the number of samples is finite. At first we define some terms.

We say θ is a continuous parameter that dominates the transition probability matrix, Θ is a set of parameters and θ^* , $\theta^* \in \Theta$ is the true parameter. S , $s_i \in S$ is a finite set of states. A , $a_k \in A$ is a finite set of actions. $r(s_i, a_k, s_j)$ is a reward function representing the reward for transition from state s_i to state s_j when action a_k is chosen. $p(s_j | s_i, a_k, \theta)$ is an element of the transition probability matrix dominated by θ , which means the probability of transition from state s_i to state s_j when action a_k is chosen. T is the number of samples, and is regarded as the length of the task. x_t , $x_t \in S$ is a state at time t . x_0 is the initial state. y_t is the action chosen at time t . $x_0 y_0 x_1 y_1 \cdots x_{t-1} y_{t-1} x_t$ is the transition history from time 0 to time t . And $x_0 y_0 x_1 y_1 \cdots x_{t-1} y_{t-1} x_t$ is also described as $x(yx)^t$. $p(\theta)$ is the prior probability density function. $p\left(\theta \middle| x(yx)^t\right)$ is the posterior probability density function when the transition history is $x(yx)^t$. $\pi\left(x(yx)^t\right)$ is the policy determining the action at state x_t when transition history is $x(yx)^t$. Π , $\pi(\,\cdot\,) \in \Pi$ is a finite set of policies. In this task, the true parameter θ^* is unknown; all other things are known.

In Martin's algorithm the optimal action at state x_t , when the transition history is $x(yx)^t$, is chosen as follows:

$$\begin{aligned} &\pi_{Martin}\left(x(yx)^t\right) \\ &= \arg\max_{a} \sum_{s_i} \bar{p}\left(s_i \middle| x_t, a, x(yx)^t\right)\left(r(x_t, a, s_i) + \beta \bar{r}\left(x(yx)^t a s_i\right)\right), \end{aligned} \tag{3}$$

where

$$\begin{aligned} \bar{r}\left(x(yx)^t a s_i\right) = &\sum_{s_j} \bar{p}\left(s_j \middle| s_i, \pi_{Martin}\left(x(yx)^t a s_i\right), x(yx)^t a s_i\right) \\ &\left(r\left(s_i, \pi_{Martin}\left(x(yx)^t a s_i\right), s_j\right) + \beta \bar{r}\left(x(yx)^t a s_i \pi_{Martin}\left(x(yx)^t a s_i\right) s_j\right)\right), \end{aligned} \tag{4}$$

$$\bar{p}\left(s_i \middle| x_t, a, x(yx)^t\right) = \int_{\Theta} p\left(\theta \middle| x(yx)^t\right) p\left(s_i \middle| x_t, a, \theta\right) d\theta. \tag{5}$$

Formula (3) is calculated by applying DP(Dynamic Programming).

Martin's algorithm, however, can not represent the trade-off between IR result accuracy and the user's burden. Accordingly, we expand Martin's model by applying query refinement in the next section.

4. Model of Query Refinement

4.1 Outline of Query Refinement

We outline query refinement by dividing it into several steps. (M_0 M_1 , M_2 and M_3 are constants.)

Step1: The user determines M_1 keywords.

Step2: The system returns M_2 related terms or M_3 documents.

Step3: If the user wants to continue, he answers acceptance or rejection on each related term or document returned by the system in Step2. If the user wants to finish, he answers that he is satisfied or not satisfied and finishes.

Step4: Until the user finishes, Step2 and Step3 are repeated M_0 times at most. After repeating Step2 and Step3 M_0 times, the user has to answer that he is satisfied or not satisfied, and this process is finished.

4.2 Definitions

We give some definitions which are different from Martin's definitions or new here. Other definitions are the same as in Martin's paper.

$WORD$, $w_i \in WORD$ is a finite set of words. M_1 keywords are chosen from $WORD$. DB , $DB \subseteq WORD$ is a finite set of words which exist in documents in a document database. DOC is a finite set of documents in the document database. S , $s_i \in S$ is a finite set of states. s_i is represented by a vector as follows except for s_1 and $s_{|S|}$:

$$\begin{aligned} s_i = &\left(s(i,1), s(i,2), \cdots, s(i,|DB|), \cdots, s(i,|WORD|),\right. \\ &\left. \cdots, s(i,|WORD| + |DOC|)\right), \end{aligned} \tag{6}$$

where

$$s(i,j)=\begin{cases}0, & \textit{default value};\\ 1, & \textit{keyword};\\ 2, & \textit{accepted by the user};\\ 3, & \textit{rejected by the user},\end{cases} \tag{7}$$

$|\cdot|$ is the cardinality of a set. s_1 is an absorbing state when the user wants to finish and is satisfied. $s_{|S|}$ is an absorbing state when the user wants to finish and is not satisfied. When s_1 or $s_{|S|}$ is reached, the query refinement process is finished. A, $a_k \in A$ is a finite set of actions. a_k indicates which M_2 related terms or M_3 documents are returned, and is represented by a vector as follows:

$$a_i=\left(a(i,1),a(i,2),\cdots,a(i,|WORD|+|DOC|)\right), \tag{8}$$

where

$$a(i,j)=\begin{cases}0, & \textit{this word or document is not returned};\\ 1, & \textit{this word or document is returned}.\end{cases} \tag{9}$$

$S(s_i,a_k)$, $S(s_i,a_k)\subseteq S$ is a finite set of states that could be reached from state s_i when action a_k is chosen. $A(s_i)$, $A(s_i)\subseteq A$ is a finite set of actions that could be chosen at state s_i. It becomes possible to represent the trade-off between IR result accuracy and the user's burden by defining a reward function, discount factor and increase factor as follows. $r(s_i,a_k,s_j)$ is the reward function representing the reward for transition from state s_i to state s_j when action a_k is chosen. It is described as follows:

$$r(s_i,a_k,s_j)=\begin{cases}R_1, & s_i\neq s_1, s_i\neq s_{|S|}, s_j=s_1;\\ R_2, & s_i\neq s_1, s_i\neq s_{|S|}, s_j=s_{|S|};\\ 0, & \textit{else},\end{cases} \tag{10}$$

where R_1, $0\le R_1<\infty$ is the positive reward gained when the user is satisfied, R_2, $-\infty<R_2\le 0$ is the negative reward gained when the user is not satisfied. α_1, $0\le\alpha_1\le 1$ is a discount factor. When R_1 is gained at time t, R_1 is valued at ${\alpha_1}^t R_1$. α_2, $1\le\alpha_2\le\infty$ is an increase factor. When R_2 is gained at time t, R_2 is valued at ${\alpha_2}^t R_2$. Initial state x_0 is determined by M_1 keywords. $XY(x_0)$, is a finite set of transition histories all of which are from x_0 to s_1 or $s_{|S|}$. $XY(x_0,\pi)$, $XY(x_0,\pi)\subseteq XY(x_0)$ is a finite set of transition histories all of which are from x_0 to s_1 or $s_{|S|}$ when policy π is used. In this task, the true parameter θ^* is unknown; all other things are known.

4.3 Utility Function

$u\left(x(yx)^n,\pi,\theta\right)$ is the utility function that indicates the value of the reward given when the true parameter is θ, policy π is used, and the transition history is $x(yx)^n$. $u\left(x(yx)^n,\pi,\theta\right)$ is described as follows:

$$u\left(x(yx)^n,\pi,\theta\right)=\begin{cases}\alpha_1^{n-1}R_1, & x(yx)^n\in XY(x_0,\pi), x_n=s_1;\\ \alpha_2^{n-1}R_2, & x(yx)^n\in XY(x_0,\pi), x_n=s_{|S|};\\ 0, & x(yx)^n\notin XY(x_0,\pi).\end{cases} \tag{11}$$

In fact, the value of the utility function doesn't depend on π, θ but on $x(yx)^n$. Accordingly, it can also be described as follows:

$$\begin{aligned}u\left(x(yx)^n,\pi,\theta\right)&=u'\left(x(yx)^n\right)\\ &=\begin{cases}\alpha_1^{n-1}R_1, & x(yx)^n\in XY(x_0), x_n=s_1;\\ \alpha_2^{n-1}R_2, & x(yx)^n\in XY(x_0), x_n=s_{|S|};\\ 0, & x(yx)^n\notin XY(x_0).\end{cases}\end{aligned} \tag{12}$$

4.4 Bayes Expected Utility Function

The Bayes expected utility function yields the expected value of the utility function when the prior probability density function is $p(\theta)$, the policy π is used, and the initial state is x_0. The Bayes expected utility function is described as follows:

$$\begin{aligned}&B\overline{u}(x_0,\pi,p(\theta))\\ &=\int_{\theta\in\Theta}p(\theta)\sum_{n=1}^{M_0}\sum_{x(yx)^n\in XY(x_0,\pi)}p\left(x(yx)^n|x_0,\pi,\theta\right)u\left(x(yx)^n,\pi,\theta\right)d\theta.\end{aligned} \tag{13}$$

4.5 Bayes Decision

Bayes Decision is the optimal decision with reference to the Bayes criterion. In this task the Bayes decision is the policy that maximizes the Bayes expected utility function when the prior probability density function is $p(\theta)$, and is described as follows:

$$BD\left(p(\theta)\right)=\arg\max_{\pi} B\overline{u}\left(x_0,\pi,p(\theta)\right). \tag{14}$$

5. Proposed Algorithm

The algorithm proposed herein calculates the Bayes decision described by formula (14). It represents the trade-off between IR result accuracy and the user's burden by adopting the reward function of formula (10), together with a discount factor and an increase factor.

The Bayes decision of formula (14) can also be described as follows:

$$
\begin{aligned}
&BD(p(\theta)) = \\
&\arg\max_{y_0 \in A(x_0)} \sum_{x_1 \in S(x_0, y_0)} \bar{p}(x_1 | x_0, y_0, x_0)\Big(u'(x_0 y_0 x_1) \\
&+ \max_{y_1 \in A(x_1)} \sum_{x_2 \in S(x_1, y_1)} \bar{p}(x_2 | x_1, y_1, x_0 y_0 x_1)\Big(u'(x_0 y_0 x_1 y_1 x_2) \\
&+ \max_{y_2 \in A(x_2)} \sum_{x_3 \in S(x_2, y_2)} \bar{p}\left(x_3 \middle| x_2, y_2, x(yx)^2\right)\Big(u'\left(x(yx)^3\right) \\
&\cdots \\
&+ \max_{y_{M_0-1} \in A(x_{M_0-1})} \sum_{x_{M_0} \in S(x_{M_0-1}, y_{M_0-1})} \\
&\bar{p}\left(x_{M_0} \middle| x_{M_0-1}, y_{M_0-1}, x(yx)^{M_0-1}\right) u'\left(x(yx)^{M_0}\right)\Big)\cdots\Big).
\end{aligned}
\tag{15}
$$

The Bayes decision of formula (15) can be calculated by applying DP from time $M_0 - 1$ to time 0. When $t = M_0 - 1$ the action is decided as follows:

$$
\begin{aligned}
&\pi_{pro}\left(x(yx)^t\right) \\
&= \arg\max_{a \in A(x_t)} \left(\alpha_1^t R_1 \bar{p}\left(s_1 \middle| x_t, a, x(yx)^t\right) + \alpha_2^t R_2 \bar{p}\left(s_{|S|} \middle| x_t, a, x(yx)^t\right)\right).
\end{aligned}
\tag{16}
$$

When $t = M_0 - 2$ the action is decided as follows:

$$
\begin{aligned}
&\pi_{pro}\left(x(yx)^t\right) = \arg\max_{a \in A(x_t)} \Big(\alpha_1^t R_1 \bar{p}\left(s_1 \middle| x_t, a, x(yx)^t\right) \\
&+ \sum_{\substack{s_j \in S(x_t, a) \\ s_j \neq s_1, s_{|S|}}} \bar{p}\left(s_j \middle| x_t, a, x(yx)^t\right) \bar{r}\left(x(yx)^t a s_j\right) + \alpha_2^t R_2 \bar{p}\left(s_{|S|} \middle| x_t, a, x(yx)^t\right)\Big),
\end{aligned}
\tag{17}
$$

where

$$
\begin{aligned}
\bar{r}\left(x(yx)^t a s_j\right) =& \alpha_1^t R_1 \bar{p}\left(s_1 \middle| s_j, \pi_{pro}\left(x(yx)^t a s_j\right), x(yx)^t a s_j\right) \\
&+ \alpha_2^t R_2 \bar{p}\left(s_{|S|} \middle| s_j, \pi_{pro}\left(x(yx)^t a s_j\right), x(yx)^t a s_j\right).
\end{aligned}
\tag{18}
$$

When $0 \le t \le M_0 - 3$ the action is decided as follows:

$$
\begin{aligned}
&\pi_{pro}\left(x(yx)^t\right) = \arg\max_{a \in A(x_t)} \Big(\alpha_1^t R_1 \bar{p}\left(s_1 \middle| x_t, a, x(yx)^t\right) \\
&+ \sum_{\substack{s_j \in S(x_t, a) \\ s_j \neq s_1, s_{|S|}}} \bar{p}\left(s_j \middle| x_t, a, x(yx)^t\right) \bar{r}\left(x(yx)^t a s_j\right) + \alpha_2^t R_2 \bar{p}\left(s_{|S|} \middle| x_t, a, x(yx)^t\right)\Big),
\end{aligned}
\tag{19}
$$

where

$$
\begin{aligned}
\bar{r}\left(x(yx)^t a s_j\right) =& \alpha_1^t R_1 \bar{p}\left(s_1 \middle| s_j, \pi_{pro}\left(x(yx)^t a s_j\right), x(yx)^t a s_j\right) \\
&+ \sum_{\substack{s_k \in S\left(s_j, \pi\left(x(yx)^t a s_j\right)\right) \\ s_k \neq s_1, s_{|S|}}} \bar{p}\left(s_k \middle| s_j, \pi_{pro}\left(x(yx)^t a s_j\right), x(yx)^t a s_j\right) \\
&\qquad \bar{r}\left(x(yx)^t a s_j \pi_{pro}\left(x(yx)^t a s_j\right) s_k\right) \\
&+ \alpha_2^t R_2 \bar{p}\left(s_{|S|} \middle| s_j, \pi_{pro}\left(x(yx)^t a s_j\right), x(yx)^t a s_j\right).
\end{aligned}
\tag{20}
$$

Thus we can maximize the expected reward, which represents the trade-off between IR result accuracy and the user's burden, with reference to the Bayes criterion.

6. Conclusion

This paper has described how to optimize query refinement. Previous research[7] on query refinement did not theoretically consider the trade-off between IR result accuracy and the user's burden. We proposed a new algorithm that maximizes the expected reward, which represents the trade-off between IR result accuracy and the user's burden, with reference to the Bayes criterion; we apply MDP with an unknown parameter. The proposed algorithm is also regarded as a generalization of Martin's algorithm[4] which solves the discounted problem of MDP with reference to the Bayes criterion.

This paper models the task of query refinement as the MDP problem of length M_o, and the proposed algorithm is the optimal algorithm for solving that MDP problem. Roughly speaking , the algorithm in the previous research[7] on query refinement can be regarded as an approximate algorithm that solves the MDP problem whose length is 1, note that the MDP problem that should be solved has length of M_o. Simulations showed that with the discounted problem, the reward increases with task length[3]. It can be easily imagined that the task of query refinement shows the same property.

The order of the computational complexity of the proposed algorithm is exponential with respect to M_0. The order of Martin's algorithm is also exponential. An improved algorithm has been proposed for discounted problem[2]. The order of the improved algorithm is polynomial under the condition that the Bayes decision is calculated. We expect that our algorithm can be improved in the same way. Its computational complexity is still very big, even though it has polynomial order. A future task is to further reduce its computational complexity.

In the proposed algorithm, the transition probability matrix dominated by an unknown parameter is learned while performing query refinement. The learned transition probability matrix can be seen as a kind of thesauri or ontology for query refinement or IR.

References

[1] Kaji, M., Ukita, Y., Matsushima, T., "A Note on Information Retrieval by Relevance Feedback"(in Japanese), The 21st Symposium on Information Theory and Its Applications, 1998.

[2] Maeda, Y., Ukita, Y., Matsushima, T., Hirasawa, S., "The Optimal Algorithms for the Reinforcement Learning Problem Separated into a Learning Period and a Control Period"(in Japanese), Trans.IPS.Japan, Vol.39, No.4, pp.1116-1126, 1998.

[3] Maeda, Y., Ukita, Y., Matsushima, T., Hirasawa, S., "Proposition of Learning Algorithms on Markov Decision Processes with Unknown Parameter"(in Japanese), The 19th Symposium on Information Theory and Its Applications, pp.597-600, 1996.

[4] Martin, J.J., "Bayesian Decision Problems and Markov Chains", John Wiley & Sons, 1967.

[5] Martin,L.P., "Markov Decision Processes", John Wiley & Sons, 1994.

[6] Salton, G., Buckley, C., “Improving Retrieval Performance Relevance Feedback”, Journal of the American Society for Information Society, Vol.41, No.4, pp.288-297, 1990.
[7] Velez, B., Weiss, R., Sheldon, M.A., Gifford, D.K., “Fast and Efficient Query Refinement”, SIGIR97, pp.6-15, 1997.
[8] Watkins, C.J.C.H., Dayan, P., “Q-Learning”, Machine Learning, Vol.8, pp.279-292, 1992.