Annals of University of Craiova, Math. Comp. Sci. Ser.
Volume 36(2), 2009, Pages 25–34
ISSN: 1223-6934

Adaptive Web Recommendation Systems

Mircea Preda, Ana-Maria Mirea, Constantin Teodorescu-Mihai, and Doina Lavinia Preda

Abstract. The online recommendations are used by a large number of Web sites to increase the revenues or to reduce the operational costs. This paper presents a recommender system based on reinforcement learning. The system uses an ontological structure to represent relations between the various information elements from site and to facilitate the generalization. The performances of the proposed method are compared with other two well known recommending techniques.

Key words and phrases. Intelligent tutoring systems, learning control systems, unsupervised learning, web recommendations.

1. Introduction

The Web pages with dynamical content, which can vary according to the user profile, are important components of the modern Web sites. The Web recommendations [1], [6] are a popular form of dynamical content used by all types of Web sites to increase the visitors satisfaction and the overall efficiency. The good quality recommendations directly influence the performances by increasing the sales level for the commercial online stores, by reducing the costs involved by the online support sites of the large manufactures or simply by reducing the Web site loading due to the fact that the visitors are better guided to their points of interest. It is worth pointing that, in the recent years, the business researchers revealed the need to develop customized products to meet the needs of the different clients. The recommender systems can be at least a partial solution to achieve this goal.

There were developed many techniques to generate Web recommendations, techniques that involve methods from statistics, machine learning, data mining and other computer related fields and are applied on various attributes of the available information like user profile, buying history, product class, etc.

The paper presents a new method to generate Web recommendations that applies a reinforcement learning algorithm [8], [5] to optimize the recommendations according to the users' feedback. The relations between the different information elements from Web site are represented by an ontological structure involving a partial order relation. This allows to measure the differences between two information elements and facilitates the generalization between the similar ones. The approach has the following desirable characteristics:

(1) the method is permanently adapting to the changes in the visitors' behavior;
(2) it is nonintrusive for the visitors because it does not require any special feedback from them;

All authors contributed equally to the paper.

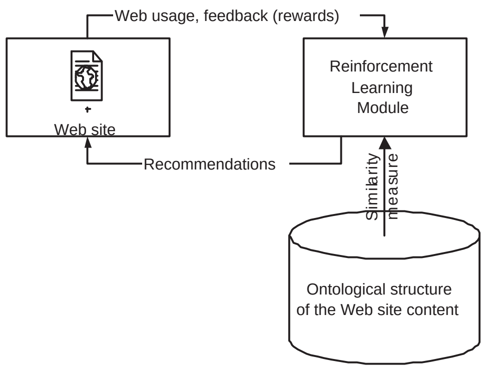
Web usage, feedback (rewards)
Web site
Reinforcement Learning Module
Recommendations
Similarity measure
Ontological structure of the Web site content

FIGURE 1. The generic architecture of a recommendation system that uses a reinforcement learning + ontological structure based recommender method

(3) it is largely applicable, practically it can be used in any interactive system that presents information to the human users like e-learning applications, intelligent user interfaces, etc.

The possibility to employ reinforcement learning methods for Web recommendations was discussed for several times. In a recent paper [2], the reinforcement learning was used to refine the recommendation database, which contains recommendations provided by many different algorithms, based on the feedback obtained from the Web site. Another related work is [3], which defines a tour guide software agent for assisting users browsing Web. The reinforcement learning is one of the employed learning methods for this agent with the states represented by Web pages and the actions by hiperlinks.

The next section contains an overview of the recommender system architecture. After that, we detail the ontological structure used to represent the relations between the information elements presented on the Web site (Section 3) and the reinforcement learning method used to obtain an optimal behavior (Section 4). Finally, we describe the test environment used to assess our method and provide brief conclusions.

2. Architecture

The Fig. 1 presents the main components of a generic recommendation system that employs the new introduced reinforcement learning based method and the relations between them.

(1) The Web site interacts with the visitors, presents recommendations, records the Web usage and generates feedback for the reinforcement learning algorithm.
(2) The reinforcement learning method computes the recommendations according to the current state of the interaction between visitor and Web site. It is permanently adapting based on the received feedback.
(3) The ontological structure of the content provides a metric used by the reinforcement learning module to generalize between Web site interactions and between different recommendations.

The ontological knowledge is fundamental in modeling of the Web surfing behavior as was already signaled in other papers. The following section describes the concepts and the relationships that exist for our reinforcement learning agent. For the rest of the paper, the words *concept* and *information element* will be used interchangeably.

3. Ontological structure of the content

Let C be a finite nonempty set of information elements that will be displayed on the Web site. The cardinal of C is $|C| = p > 0$. The relations between the components of the set C are specified by a binary relation $<\subseteq C \times C$. A relation $c_1 < c_2$, $c_1, c_2 \in C$ signifies that the information element c_1 influences the meaning of the information element c_2.

The relation $<$ is supposed to have the following properties:

(1) $<$ is reflexive: $c < c, \forall c \in C$;
(2) Let $\Gamma = (C, <)$ be the directed graph defined by $<$. Γ does not contain non trivial cycles $c_1 < c_2 < ... < c_n < c_1$ with $n > 1$ and $c_i \neq c_j, \forall i \neq j$.

The binary relation $<$ can be extended to a partial order relation $<^* \subseteq C \times C$. For every two elements $c_1, c_2 \in C$, we have $c_1 <^* c_2$ if and only if there is a possibly empty sequence $c_{i_1}, c_{i_2}, ..., c_{i_n}$ such that $c_1 < c_{i_1} < c_{i_2} < ... < c_{i_n} < c_2$. Obviously, $<^*$ is reflexive, antisymmetrical and transitive.

Let $Inf(C)$ be the set $Inf(C) = \{c^* \in C | \nexists c' \in C, c' \neq c^* \text{ such that } c' < c^*\}$ and $|Inf(C)| = N > 0$ the cardinal of this set. The elements of $Inf(C)$ represent elementary concepts and they will be denoted by $c_1^*, ..., c_N^*$.

In order to quantize the influence of an information element c_1 on another information element c_2, influence signaled by the statement $c_1 < c_2$, we suppose known a function $\beta : C \times C \to [0, 1]$, function that has the following properties:

$$\beta(c_1, c_2) = \begin{cases} 1 & \text{if } c_1 = c_2 \\ \neq 0 & \text{if } c_1 < c_2 \text{ and } c_1 \neq c_2 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Based on β, we define a function $\beta^* : C \times C \to [0, 1]$ by:

$$\beta^*(c_1, c_2) = \begin{cases} 1 \text{ if } c_1 = c_2 \\ M[\theta^n \beta(c_1, c_{i_1})\beta(c_{i_1}, c_{i_2}) \cdot ... \cdot \beta(c_{i_n}, c_2)] \\ \quad \text{if } c_1 <^* c_2 \text{ and } c_1 \neq c_2 \\ 0 \text{ otherwise} \end{cases} \tag{2}$$

where $M[.]$ denotes the arithmetic mean computed for all sequences of distinct elements $c_{i_1}, c_{i_2}, ..., c_{i_n} \in C \setminus \{c_1, c_2\}$ such that $c_1 < c_{i_1} < c_{i_2} < ... < c_{i_n} < c_2$ and $\theta \in (0, 1]$ is a parameter that controls how the influence of c_1 on c_2 decreases according to the distance between c_1 and c_2 in the graph Γ. The arithmetic mean is one of the many different operators that can be used to combine the values provided by the various paths between two concepts in the graph Γ. For example, another possible choice for $M[.]$ is maximum.

We suppose that each information element has a textual description on the Web site. Consequently, the values $\beta(c_1, c_2), c_1 < c_2, c_1 \neq c_2$ can be computed by statistical natural language understanding techniques that support similarity comparisons between texts. Examples of such techniques are Weighted Inverse Document Frequency, Latent Semantic Analysis (LSA) [4], and, its extension, Probabilistic Latent Semantic Analysis. All these methods assume that the simplified *bag-of-words* or vector-space representation of documents will in many cases preserve most of the relevant information and perform operations over this space to measure the similarities between texts.

If the function β is available, then β^* can be computed by the following method. Let B be the matrix: $B \in [0,1]^{p\times p}, B = (b_{ij})_{1\le i,j\le p}$ where

$$b_{ij} = \begin{cases} \beta(c_i, c_j) & \text{if } i \neq j \\ 0 & \text{otherwise} \end{cases}, 1 \le i, j \le p. \tag{3}$$

L is a matrix with same size as B obtained by applying the signum function to B. Each element from L is 1 if the corresponding element from B is greater than zero and 0 otherwise. We compute the powers $B^2 = B \cdot B, ..., B^{p-1} = B^{p-2} \cdot B$ and $L^2 = L \cdot L, ..., L^{p-1} = L^{p-2} \cdot L$ of the matrices B and L and denote by b_{ij}^k and l_{ij}^k, $1 \le i, j \le p$ the elements of the matrices B^k and L^k, $1 \le k \le p-1$. Here B^1 and L^1 stands for the matrices B and L. In the above mentioned conditions, we can prove that $\forall i, j$, $1 \le i, j \le p$, $i \neq j$:

$$\beta^*(c_i, c_j) = \frac{1}{\max(1, \sum\limits_{k=1}^{p-1} l_{ij}^k)} \sum_{k=1}^{p-1} \theta^{k-1} \cdot b_{ij}^k. \tag{4}$$

We consider that it is possible to learn patterns from the users' interactions with the Web site by employing reinforcement learning methods. In order to achieve that, the state of an interaction between user and Web site must be represented as an array of real numbers. Formally, a state s is a sequence of visited information elements $c_1, c_2, ..., c_{n_s}$ where c_1 is the most recently visited element and c_{n_s} the first visited one. Same element can appear for many times in the state s. S is the set of the all states s that can be constructed based on C. S is an infinite discrete set.

The application $x : S \to \mathrm{I\!R}^N$ defined by

$$\begin{aligned} &x(s) = (x_1(s), ..., x_N(s)), \\ &x_i(s) = \tfrac{1}{n_s} \sum_{j=1}^{n_s} \theta_x^{j-1} \beta^*(c_i^*, c_j),\ i = 1...N \end{aligned} \tag{5}$$

transforms the state s in an array $x(s)$ of real numbers. $\theta_x \in (0, 1]$ is a parameter used to give a higher importance to the latest visited elements.

A recommendation for a visitor in a state s is a set of information elements $a \subseteq C$ selected to be the most suitable for the current state s of the interaction. The recommendation a must to satisfy the following restrictions:

(1) $|a| \le M$, the number of components from a recommendation cannot exceed a specified upper bound M;
(2) $\forall c \in a$, $c \neq c_1$ and $c <^* c_1$ or $c_1 <^* c$ where c_1 is the last visited concept.

Let us denote by $A(s)$ the set of all possible recommendations for a state s. The recommendations are represented by arrays of real numbers following a procedure similar with that employed for states: $u(s) : A(s) \to \mathrm{I\!R}^N$

$$\begin{aligned} &u(s)(a) = (u_1(s)(a), ..., u_N(s)(a)), \\ &u_i(s)(a) = \tfrac{1}{|a|} \sum_{c\in a} \beta^*(c_i^*, c),\ i = 1...N. \end{aligned} \tag{6}$$

The above definitions can be slightly changed by defining an order over the components of a recommendation. This extension is natural, for example, the first component from a recommendation seen by a visitor is maybe the most important for his decision.

The suitability of the recommendations is measured based on the positive feedback provided by users. Examples of such feedback are: the user follows a recommended link, downloads a document, prints a page, gives a high ratting to a page, etc.

The next section presents the theoretical framework used to convert the users' feedback into optimal recommendations. We have chosen the reinforcement learning framework due to the following attractive features: the learning is unsupervised and it can proceed online, a model of the user is not required in order to learn, the methods can learn from delayed feedback that is common in Web browsing and there is a large amount of work on the theoretical properties of these methods.

4. Reinforcement learning

Reinforcement learning is the name of a set of methods and algorithms for control systems that automatically improve their behaviors by trying to maximize the rewards received from environment. A typical reinforcement learning problem is described by a quadruple $\{X, U, T, R\}$ where X is a finite set of states, U is a finite set of actions, $T : X \times U \times X \to [0, 1]$ is a transition function, $T(x, u, x')$ represents the probability to observe the state x' if the action u is performed in the state x, and R is a reward function, $R : X \times U \to \mathbb{R}$. We considered the temporal difference (TD(λ)) reinforcement learning algorithms that are combinations of concepts belonging to Monte Carlo and dynamical programming methods. They permanently adjust the behavior of the system by updating a value function $Q : X \times U \to \mathbb{R}$, where $Q(x, u)$ approximates the expected long term return that can be obtained by executing a specified action u in a given state x and after that following the current policy. If we consider a sequence of n rewards, $r_0, r_1, ..., r_{n-1}$, the expected return will be $\sum_{i=0}^{n-1} \gamma^i r_i$, where $0 < \gamma \leq 1$ is a discount factor that shows how important are the rewards received later during the interaction.

In many practical applications, the states and actions involve continuous values or are very large discrete sets. This is also our case, our state space contains an infinite number of real vectors and for each state there is a large discrete set of possible recommendations represented as arrays of real values. Consequently, the value function Q cannot be represented in memory as a table and its learning requires some techniques of function approximation. Fortunately, some recent papers discuss the convergence of the TD algorithms like SARSA (State Action Reward State Action) if linear function approximations are used. We used a class of linear function approximations named Cerebellar Model Articulation Controllers or CMACs that compute approximations $\overline{Q}$ of the function Q following the pattern: a state - action input pair (x, u) activates a specific set of memory locations and the arithmetic sum of the contents of these locations represents the estimated value of the function Q (a complete description of the estimation procedure can be found in [5]).

The approximation $\overline{Q}$ can be used to identify the best action that can be performed for a certain state of the interaction. The action is selected by computing the maximum $u^* = \arg\max_{u \in U} \overline{Q}(x, u)$. Different algorithms can be employed to obtain the optimal action; most of them are dependent on the method used to approximate the function Q [5]. A reasonable choice is an algorithm based on simulated annealing because this method behaves well in on large spaces and is widely applicable.

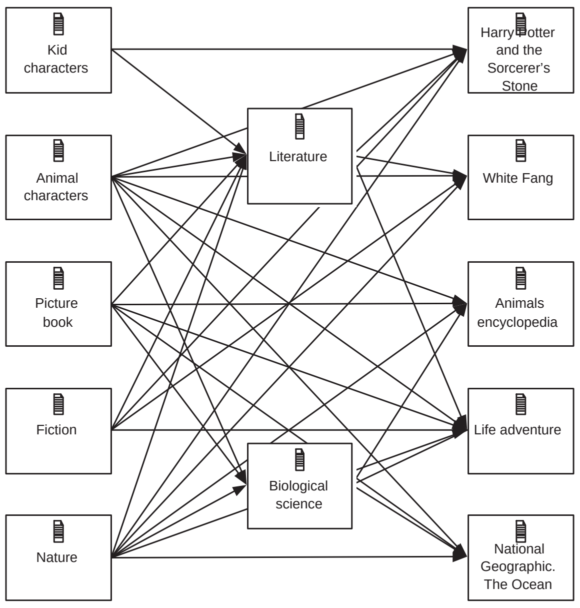

FIGURE 2. A part from the graph Γ defined by the binary relation $<$

TABLE 1. A part from the matrix B used in tests

	Literature	Biological science	Harry Potter and the Sorcerer's Stone	White Fang
Kid characters	0.5	0	0.9	0
Animal characters	0.8	0.2	0.2	0.9
Picture book	0.3	0.9	0	0
Fiction	0.8	0	0.9	0.9
Nature	0.1	0.9	0.1	0.3
Literature	0	0	0.9	0.9
Biological science	0	0	0	0

5. Implementation and experimental results

A prototype of this method was installed and tested on the digital library intranet Web site of a local secondary school. The library has 796 items and it is used by $\approx$ 500 scholars and teachers. The partial order relation $<$ was manually specified by a human editor according to the Web site structure using 849 information elements. The matrix B was defined using the weighted inverse document frequency and the cosine similarity measure. The obtained values were reviewed and adjusted by a human editor. The Fig. 2 and Table 1 present parts of the graph Γ and matrix B. In order to simplify their interpretation, the β values from B were truncated to one decimal place.

β^* was computed using $\theta = 0.9$. It is worth mentioning the non monotonic character of the obtained ontological structure. By non monotonic behavior, we understand that a more general feature of a concept is overridden by a more specifical one. For example, let us consider the *Life adventure* information element presented in Fig. 2 and in Table 1. Its meaning is influenced by the membership to the category *Biological science* and from that we can conclude that there is no relationship between it and *Fiction* concept. But the graph Γ and the function β show us that

Fiction directly influences *Life adventure* information element. There is also an indirect influence with *Literature* as intermediary. Using data from Table 1 we have $\beta^*(Fiction, Life\ adventure) = 0.5 \cdot (0.1 + 0.9 \cdot 0.8 \cdot 0.1) = 0.086$. Small values of the parameter θ make the specifical features more and more important compared with the inherited ones. The non monotonic behavior is common in concept graphs.

The maximal number of presented recommendations was $M = 3$ and the parameter θ_x was chosen to be 0.9.

The algorithms TD(λ) are influenced by a large number of parameters and their values can be crucial for the success or failure of the control process. The identification of an optimal set of parameters is basically a trial and error process. The parameter λ is used to share the current reward with the past state-action pairs that contributed to it. The conducted experiments showed that in our settings, for the parameter λ, the small values are preferable. In tests we used the value $\lambda = 0.5$.

Regarding the configuration of the CMAC approximation (tile coding), [7] provides a detailed discussion about its performances and the optimal parameters' settings. A larger value for the number of tilings (the number of activated memory locations) involves better generalization capacities and improved performances in the early stages of the training. Later, the approximation quality is affected and the best solution is to use an adaptive algorithm. The largest value used in our tests was 8.

The reward function R and the initial values of the weights attached to the features of the CMAC approximator have a great influence on the convergence speed to an optimal solution. Some choices for these parameters can favor a deep exploration of the state-action space. Other variants can make an early preference to already explored regions.

The parameter γ is used to find equilibrium between the exploitation and exploration necessities of the control system. Our experiments show that a small value for γ, which favors an early exploitation of the already achieved knowledge, is preferable for our task.

The ϵ-greedy policy chooses the action $arg\ \max_u Q(x, u)$ with the probability $1-\epsilon$ and, otherwise, it selects a random action according with a uniform distribution. It assures that there is no state-action pair that is ignored and it is another way to balance the exploration and exploitation. Suitable values for ϵ are between 0.1 and 0.01.

In the above configuration we are placed under the conditions that guarantee us the SARSA algorithm will converge with probability 1 to a fixed bounded region.

The performances of our method were compared in these settings with other two well known recommender techniques: mining association rules to develop *top-N* recommender systems and recommender systems based on collaborative filtering. For *top-N* systems, we considered the *top-N from category* recommendation method where, for each item category, the N most frequently selected items from category are recommended. The collaborative filtering methods are represented by the *item-to-item collaborative filtering* recommendation method. According to this method, items, which often appear together in a user's session, are recommended to each other. Similarly to our reinforcement learning recommendation method, the recommendations provided by *top-N from category* and *item-to-item collaborative filtering* apply to all users of the system.

The experiments were conducted using a training set of 6210 traces starting from the library home page and an evaluation set consisting from 1537 traces of interactions. The results in table 2 provide a comparison of the recommendation methods. The

TABLE 2. The usefulness of the recommendation methods measured on an evaluation set of 1537 traces

Recommendation method	Usefulness
top-N from category	41.3%
item-to-item collaborative filtering	44.5%
reinforcement learning	44.8%

TABLE 3. Impact of changes in visitors' behavior on the usefulness of the recommendation methods

Recommendation method	Usefulness
top-N from category	30.4%
item-to-item collaborative filtering	31.4%
reinforcement learning	39.7%

TABLE 4. Impact of adding new items to library on the usefulness of the recommendation methods

Recommendation method	Usefulness
top-N from category	39.1%
item-to-item collaborative filtering	41.2%
reinforcement learning	43.5%

usefulness measures the percentage of interactions for which the user followed one of the presented recommendations. The *item-to-item collaborative filtering* and the reinforcement learning recommendation method have roughly same performances and are significatively better than *top-N from category* on our task.

To evaluate the ability of the recommendation methods to deal with changes in the users' behavior, we used an evaluation set consisting from 409 interactions where the users were only teachers. The results in table 3 show that the reinforcement learning recommendations are significatively better due to the fact that they are permanently adapting to the users' behavior.

Finally, the recommender systems were tested for the new items problem. 15 new items were inserted in the digital library and the ontological structure was extended accordingly. The table 4 shows, on an evaluation set of 1512 interactions, that the decreasing in performance is smaller in the case of the reinforcement learning recommender system. This result is justified by the generalization capacity of the employed reinforcement learning method.

The presented performance results are dependent on problem and on the employed settings but anyway they show that the proposed reinforcement learning recommender can be a competitor for the two well known recommendation methods.

6. Conclusion

The paper presents a novel method to generate recommendations using a reinforcement learning algorithm and an ontological structure of the concepts. This structure is used to measure the similarities between concepts and to generalize between similar states of the interaction visitor - Web site. The characteristics of the method like its adapting capacity, its applicability, its non intrusive behavior and the performances obtained during during evaluation tests are arguments to propose the method as a real alternative to implement a recommender system.

The development of the method in the near future will be focused on three directions:

(1) We plan to extensively test the performances of the method on various high complexity tasks and to compare them with the results of other well known recommenders;
(2) A topic of particular interest is how to measure the similarities between concepts when these are represented by logic programs. Also, the developments in the semantic networks domain should be useful to determine the similarities.
(3) The Semantic Web framework allows inclusion in Web pages of information in formats that can be easily interpreted by software. This should facilitate the implementation of a recommendation system that employs ontological structures to measure the resemblances between different situations in order to generalize.

References

[1] R. Burke, Hybrid recommender systems: Survey and experiments, User Modeling and User-Adapted Interaction, 12(4), 2002, 331-370.
[2] N. Golovin and E. Rahm, Reinforcement learning architecture for Web recommendations, Proc. of the International Conference on Information Technology: Coding and Computing, 2004, 398-404.
[3] T. Joachims and D. Freitag and T. Mitchell, WebWatcher: A tour guide for the World Wide Web, Proc. of IJCAI97, 1997
[4] T. K. Landauer and S. T. Dumais, A Solution to Plato's Problem: The Latent Semantic Analysis theory of acquisition, induction and representation of knowledge, Psychological Review, 104, 1997, 211-240
[5] J.C. Santamaria and R.S. Sutton and A. Ram, Experiments with reinforcement learning in problems with continuous state and action spaces, Adaptive Behavior, 6(2), 1998, 163-218
[6] B. Sarwar and G. Karypis and J. Konstan and J. Riedl, Analysis of recommendation algorithms for e-commerce, Proc. of ACM E-Commerce, 2000, 158-167
[7] A. A. Sherstov and P. Stone, On Continuous-Action Q-Learning via Tile Coding Function Approximation", Under Review., June, 2004
[8] R. S. Sutton and A. G. Barto, Reinforcement learning: An introduction, MIT Press, Cambridge, MA, 1998

(Mircea Preda) University of Craiova, Department of Computer Science, Al. I. Cuza 13, 200585 Craiova, Romania
E-mail address: `mpreda@acm.org`

(Ana-Maria Mirea) University of Craiova, Department of Computer Science, Al. I. Cuza 13, 200585 Craiova, Romania
E-mail address: `ammirea@acm.org`

(Constantin Teodorescu-Mihai) University of Craiova, Department of Computer Science, Al. I. Cuza 13, 200585 Craiova, Romania
E-mail address: `constantintm@ymail.com`

(Doina Lavinia Preda) University of Craiova, Department of Computer Science, Al. I. Cuza 13, 200585 Craiova, Romania
E-mail address: `dpreda@acm.org`