ROSA – Multi-agent System for Web Services Personalization

Przemysław Kazienko[1] and Maciej Kiewra[2]

[1] Wrocław University of Technology, Department of Information Systems, Wybrzeże S. Wyspiańskiego 27, 50-370 Wrocław, Poland
kazienko@pwr.wroc.pl
[2] Fujitsu España General Elio, 2 – entlo. dcha. – 46010 Valencia, Spain
mkiewra@mail.fujitsu.es, matixy@wp.pl

Abstract. Automatic and non-invasive web personalization seems to be a challenge for nowadays web sites. Many web mining techniques are used to achieve this goal. Since current web sites evolve constantly, web mining operations should be periodically repeated. A multi-agent architecture facilitates integration of different mining methods and permits the discovered knowledge to be verified and updated automatically. We propose ROSA (Remote Open Site Agents) — a system that may be easily incorporated into an existing web site. It consists of multiple heterogeneous agents such as: User Session Monitor, Crawler, Content Miner, Usage Miner, Hyperlink Recommender, Banner Recommender, etc., that are responsible for specific web mining and personalization tasks. They integrate various mining techniques using common representation of documents in the vector space model in order to recommend hyperlinks and banners. Verification process is represented by a task graph. ROSA agents not only detect when their information should be verified, but they are also able to coordinate knowledge update operations (using method presented in this paper). The practical part describes the usage of FIPA- RDF0 and ACL languages.

1 Introduction

Personalization is one of the most important techniques used for increasing the number of clients. Developers can create web sites that use efficient and automatic personalization techniques that do not require any user's intervention. Those systems acquire information about clients' behaviour in order to provide content adapted to individual necessities [16]. Hyperlinks to probably interesting documents [14,18] or to commercial offers (e.g. banners [1,11]) are the most typical examples.

1.1 Personalization Problems

As the information about individual interests and preferences is hidden among a huge volume of data, web mining techniques are very useful in personalization

E. Menasalvas et al. (Eds.): AWIC 2003, LNAI 2663, pp. 297–306, 2003.

issue. These techniques are grouped in accordance with the type of analysed data into [13]: *web usage mining* (analysis of navigation patterns and other data related to users' activity) [12,14,18] and *content mining* (where documents' content is mined) [2]. Both web mining branches can complement each other in order to provide more effective personalization [15]. Our system ROSA (Remote Open Site Agents) also integrates web usage and content mining (by means of vector structure) in one coherent system that can be incorporated into every web site [9,10].

Since the web mining process consists of many time-consuming steps, it was originally divided into two parts. The former consisted of operations that could be performed off-line (data cleaning and selection, clustering, etc.). The latter was composed of tasks that must take place on-line (current user session classification, ranking list creation, etc.). Since the current sites evolve constantly (the site content and users' activities are changing), the off-line operations should be periodically repeated to update the mined information. Typically off-line operations are up to the site administrator who had to decide when and how often the whole process should be performed again.

Obviously, the update will be quite trivial, if the web site is small, but it may be a real headache in case of professional huge portals. Manual monitoring of changes can be very tedious administrative duty. The more data a web site generates, the longer the update process is and the greater probability of system inconsistencies is. Therefore, a strict data coordination method is required. Additionally, the off-line operations should be carried out when the users' activities are minimal.

1.2 Multi-agent System

For all these reasons, ROSA has been evolved towards multi-agent system, whose expert-agents cooperate with one another and may be distributed among many hosts [5]. Every agent is responsible for a single web mining task, so it encapsulates specific functions that would be available for the rest of the system. Agents not only interchange information, but they also possess their own knowledge.

ROSA can be included in every web site. From the end-users' point of view, ROSA is an assistant that facilitates the site navigation. The current version is able to recommend hyperlinks and banners using information about site content, previously visited pages and typical web usage patterns. Moreover, it provides a search engine and some administration tools (e.g. statistics and association rules discovering).

2 ROSA Agents in Personalization Process

Every ROSA expert-agent possesses its own characteristics related to personalisation process (Fig. 1).

Crawler retrieves the site content using HTTP, extracts terms from documents and counts the document-term frequency (*itf*). This capability is also

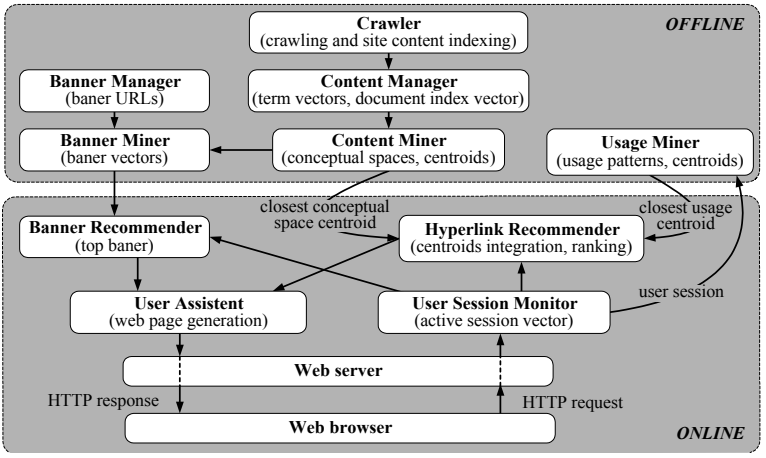

Fig. 1. Personalization process

supplied as a service (used for example by Banner Miner). Crawler is also responsible for a periodical web site monitoring and deciding whether changes are serious enough to start the update process. Crawler takes the first step in the personalization process.

Content Manager creates term vectors using *itf* indices generated by Crawler. For every term t_i a N-dimensional vector $c_i^t = \langle w_{i1}^c, w_{i2}^c, \ldots, w_{iN}^c \rangle$ is created. w_{ij}^c denotes the weight of the term t_i in the document d_j. N is the number of documents. As vector representation is common for the whole system, Content Manager stores URL index that assigns vector coordinate to the specific URL address (*document index vector*).

Content Miner clusters term vectors. Each cluster corresponds to a thematic group denoted *conceptual space* [9], which is represented by the term vector denoted *centroid*. Not all terms are used in the clustering process, only the best content descriptors are chosen by means of the information of their occurrences in documents and queries sent to the Search Engine [9].

User Session Monitor captures users' HTTP requests and groups them into sessions using JSP servlet session mechanism [8]. For every session s_i a N-dimensional vector $c_i^t = \langle w_{i1}^s, w_{i2}^s, \ldots, w_{iN}^s \rangle$ is created. The coordinate w_{ij}^s has non-zero value, if the document d_j was visited in the session s_i [9]. User Session Monitor possesses historical sessions and active user sessions (sessions of the users who are currently on-line). Since it is able to estimate the number of requests per time, it informs the Host Manager about WWW server overload.

Usage Miner clusters historical sessions into typical usage patterns. Each pattern is represented by the mean cluster vector — *centroid*.

Hyperlink Recommender is responsible for creating hyperlink ranking lists. It receives the current user's session vector from User Session Monitor and sends it to Usage Miner and Content Miner. The former finds the closest usage pattern centroid and the latter — the closest conceptual space centroid, using cosine

similarity measure [17]. Both centroids and the current user session vector are joined together by Hyperlink Recommender. It forms a ranking list according to the algorithm presented in [9]. n top documents from the ranking list are presented to the user (by means of User Assistant).

Banner Manager is responsible for removing, adding and modifying banner related data.

Banner Miner creates a banners' vector $b_i = \langle w^b_{i1}, w^b_{i2}, \ldots, w^b_{iM} \rangle$ for every conceptual space, where w^b_{ij} corresponds to the similarity between i^{th} conceptual space andj^{th} banner; M is the number of banners. The value of w^b_{ij} is calculated using terms common for the conceptual space and the banner target page.

Banner Recommender chooses a banner that should be presented to the user on the current page. Banner Recommender relays to Content Miner the current user session vector obtained from User Session Monitor. Content Miner returns the closest conceptual space centroid. Next, Banner Recommender asks Banner Miner for the b_i vector corresponding to the conceptual space. Banner Recommender creates a banner ranking list using the b^{th}_i vector coordinates, each banner life-time and the number of times the banner must be exposed (according to the advertise contract conditions). Additionally, the information about m-last exposed banners is stored for each on-line user in order to prevent the same banner to be showed too many times.

Search Engine finds and ranks the documents that fulfil query's criteria. It extracts terms from the query and asks Content Manager for a list of corresponding term vectors. The vectors are joined together in a result vector in accordance with the operators placed between terms in the query. The result list is created from the result vector: documents are ordered by corresponding result vector's coordinates. Additionally, the frequencies of queried terms are stored in order to permit Content Miner to select appropriate conceptual space descriptors.

User Assistant sends requests to agents and generates ROSA page area.

Every host on which ROSA agents reside has an auxiliary *Host Manager* agent. It is responsible for mediation between agents that try to use system resources from the same host simultaneously.

3 Knowledge Verification Method

The knowledge of an agent depends on other ROSA agents' knowledge. Therefore, the update process can be represented as a directed acyclic *task graph* where knowledge verification tasks performed by single agents are modelled by nodes and dependency between nodes are represented by directed arcs (Fig. 2). Task graphs are widely used especially in parallel and distributed computing [3,4]. Changes in the knowledge of an agent implicate a necessity of knowledge update in its all explicit and implicit dependent nodes. Agents that start and manage the update process are called *Update Managers*. There are four Update Managers, thus the whole task graph can be divided into a four overlapped task graphs (Fig. 2). By chance, only trees are presented on the Fig. 2, but it is not a system restriction. It is possible to add an agent whose knowledge would depend

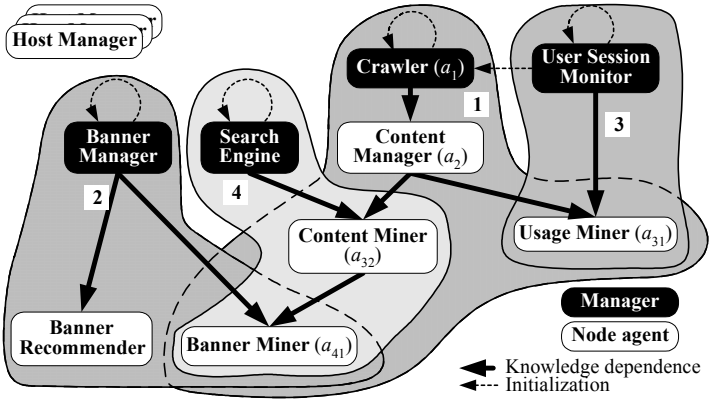

Fig. 2. Task graphs for knowledge verification

on Content Miner's and Usage Miner's knowledge. For commodity, the "tree nomenclature" (parent, child, leaf) is applied to this paper.

The update is the four-stage process that consists of:

1. Initialisation — detection of needs for knowledge updating and approval
2. Time scheduling and global time-out estimation
3. Acquisition and propagation of knowledge changes
4. Synchronization — the acceptance of changes carried out by agents.

3.1 Initialisation

Some agents may suggest to Update Manager that external knowledge has changed significantly. Such initialising agent may but does not have to belong to the task graph. Verification suggestions were marked with dotted arrows (Fig. 2):

- User Session Monitor detects non-existing pages and suggests to Crawler reindexing the site content. When User Session Monitor finds many new sessions in its database, it initiates the recalculation of usage clusters by itself.
- Crawler discovers changes in the site content (new pages, updates and deletion) on the grounds of periodically made sampling.
- Banner Manager is responsible for banner insertion and erasing, thus it suggests updating the knowledge related to banner recommendation process.
- Search Engine possesses statistics of user queries and may initialise an update of content mining data with the new most frequently asked terms.

The detected need is sent to the Update Manager with specific parameters, e.g. list of unavailable pages. The manager verifies the necessity of update (accepts or rejects it) according to the premises obtained from initialization agent, the build-in decision rules, its previous experiences and its own knowledge.

3.2 Time Scheduling and Global Time-Out Estimation

Since the whole verification process consists of various long-lasting and dependent steps, it has to be coordinated. The start time, and duration of each step and the global time-out should be estimated. The below description of time scheduling process (Fig. 3) is based on the first task graph (Fig. 2):

1. After the manager a_1 has decided to start the update process, it sends the predicted duration d_1 of its own process (determined on the basis of its experience) to its Host Manager and it negotiates the predicted best start time t_0 and finish time t_1.
2. Manager a_1 sends the predicted finish time t_1 to all its children with the questions concerning the suggested finish time for all their descendants. The children (a_1 has only one child a_2) negotiate with their Host Managers a start time (closest after t_1) and calculate their finish times. They relay the question to their children with the negotiated time (a_2 sends t_2). If an agent had two or more parents it would ask the Host Manager for the time period after $\max(t_{p1}, t_{p2}, \ldots)$ where t_{pn} is the estimated finish time of the n^{th} parent.
3. Leaves a_{31} and a_{41} return responses to their parents. Answers include only their predicted finish time (t_{31} and t_{41} respectively). Parents relay their children replies upwards in the graph. If an agent has more than one child, it will return the greatest value, i.e. a_2 will return t_{41}.
4. The global time-out is equal to the latest time returned to the manager (t_{41}).

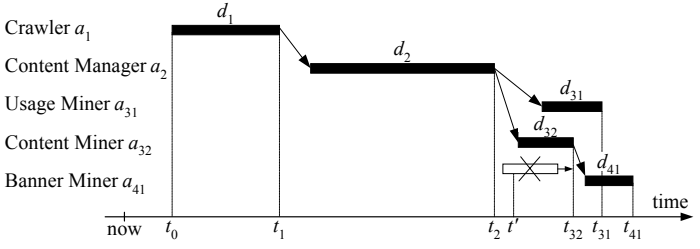

Fig. 3. Knowledge verification suggested timeline for the graph No. 1

Please note that, every agent may have already been engaged in an update process P' of another task graph, e.g. Content Miner a_{32} might be executing the process from the fourth graph. In this case, the agent asked P' Update Manager (Search Engine), through their ascendants, for the whole time-out t' of the process P'. The agent a_{32} will seek for its starting time after t' (Fig. 3). Generally, an agent must not start second process, until the first one finishes. It prevents processes from overlapping each other.

3.3 Acquisition and Propagation of Knowledge Changes

The manager and all other agents wait for the answers (concerning global time-out) only for a short time. If not all answers come to the root, the whole process is cancelled. Otherwise the manager sets up the global process time-out. At the time t_0 the manager (Crawler) begins indexing web pages. None of the involved agents overwrite its *current data*. New information are stored in the *new data*. After the manager has finished, it sends an appropriate message to its children. If any node detects that no data was changed, its children obtain a *nothing-changed message*. If a child receives a *nothing-changed message* from all its parents, it sends the same message downwards in the graph. Otherwise it starts its own verification. The process repeats recursively. As soon as a leaf agent finishes, it reports this to its parents, which once having answers from all its children relay them upwards.

If another update request (from another task graph) comes while processing, an agent waits for the first process to be finished and remembers the processes' order.

3.4 Canceling

If the whole process does not end until the global time-out (t_{41}), the manager keeps waiting. However if another initialisation suggestion comes, it cancels the unfinished process after taking decision about the new process. The cancel message is sent downwards. An agent which obtains such a message deletes all unnecessary data.

Any agent may break its process down for any reason before finishing, informing the manager about it. In such case the manager cancels the whole process similarly to the previous case. The manager informs the system administrator (e.g. by sending e-mail) every time the process is cancelled.

3.5 Knowledge Synchronization

If the manager has obtained positive responses form all its children, it begins synchronization process, sending *lock request* to all dependant nodes with short t_{lr} time-out. The agents wait for *lock report* from their children. All agents from the graph after *lock request* works in default mode (not based on *current data*) during the personalization process. If the manager does not obtain the *lock report* from all its children until t_{lr} time-out, it will cancel the process (roll it back) and it will try again after a long t_r time-out. Otherwise, the manager sends *change data request*. All agents backup *current data* to *old data* and rewrite *new data* into *current data*. Next, they report it to their parents. The manager sends *finish request* to all agents after receiving reports from its children. Agents from the task graph delete *old data* and unlock *new data*. Initialisation agents reset their parameters used for initialisation.

If two verification processes are performed in the same time and there is at least one agent that is involved in both of them, the second process must not be carried out in the conflictive node until the first is totally finished.

4 ACL and FIPA-RDF0 Usage

Each multi-agent system needs a communication language that provides a framework for every communicative act defining some standard types of messages: (inform, request, accept, etc.) and a content language that permits agents to express their world (objects, propositions, and functions). Agent Communication Language [6] can be used as the former and FIPA-RDF0 [7] as the latter. Since ROSA architecture is thought to be easily extended, task graphs are created dynamically by means of ACL messages. For example, if an agent a_1 wants to be an element of knowledge dependency graph it must request it (sending the following message):

```
(request
 :sender agent_a1
 :receiver agent_a2
 :content (<?xml version="1.0"?>
   <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-
                       rdf-syntax-ns#"
            xmlns:fipa="http://www.fipa.org/schemas#">
     <fipa:Action rdf:ID="unique-identifier">
       <fipa:actor>agent_a2</rdf:actor>
       <fipa:act>join-to-graph</rdf:act>
       <fipa:argument>
         <rdf:bag> <rdf:li>agent_a1</rdf:li> </rdf:bag>
       </fipa:argument>
     </fipa:Action>
   </rdf:RDF>)
  :language fipa-rdf0)
```

Every agent that provides any information should store a list of its knowledge dependant nodes (children and parents). Initialisation and the global time-out estimation is presented on Fig. 4, as an example of a communication language usage. Message content (*KnowledgeVerification*, *ProcStarted*, *FinTime*, etc.) is

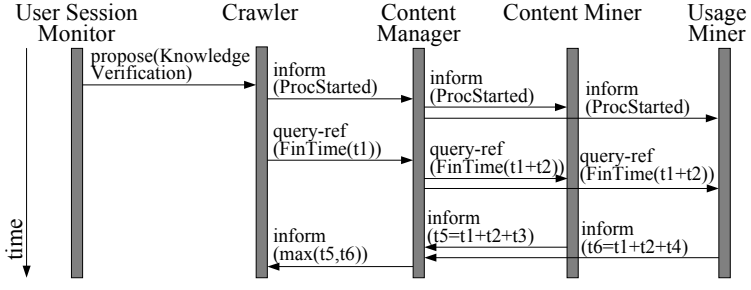

Fig. 4. Sequence diagram with ACL commands used for time scheduling

expressed in FIPA-RDF0 language. For clearness, the communication between agents and corresponding host administrators was omitted.

5 System Implementation

Agents and their communication were developed in Java. Although the server part of ROSA was implemented using JSP/Servlet technologies ROSA can be incorporated into every web site (Fig. 5). It results from that User Assistant adds only a simple HTML frame into the current page returned by the web server. The small java script code is attached to every site page in order to capture user activities by User Session Monitor.

Fig. 5. ROSA HTML frame incorporated into a web page

6 Conclusions and Future Work

Usage and integration of information coming from different sources (web usage mining, site content mining, search-engine statistics), especially when the site evolved rapidly, has many inconveniences. First of all, the process of verification of discovered knowledge is very tedious. Moreover, manual update can lead to periodical system inconsistency. Multi-agent architecture, apart from encapsulation of typical web mining tasks in heterogeneous entities, solves those problems. ROSA agents not only detect when their information should be verified, but they are also able to coordinate knowledge update tasks. Additionally, they treat local system resources like limited goods that should be shared in an effective way.

Since presented system is quite easy to expand, the future works will concentrate on implementation of new agents, extending personalization functionality of ROSA. It is considered to include Shop Miner – an agent that would recommend the products from an internet shop and give some advice for the hesitating

customers. Another challenge is to develop an agent that would be responsible for prompting sponsored hyperlinks (during the search result presentation).

References

1. Aggarwal C.C., Yu P.S.: An Automated System for Web Portal Personalization. 28^{th} VLDB Conference, Morgan Kaufmann (2002).
2. Boley D., et al.: Document Categorization and Query Generation on the World Wide Web Using WebACE. Artificial Intelligence Review 13 (5-6) (1999) 365–391.
3. Casavant T.L., Kuhl J.G.: A Taxonomy of Scheduling in General-Purpose Distributed Computing Systems. IEEE Transactions on Software Engineering, 14 (2) (1988) 141–154.
4. Coffman E.G., Graham R.L.: Optimal Scheduling for Two-Processor Systems. Acta Informatica, 1 (1972) 200–213.
5. Ferber J.: Multi-Agent Systems. Addison Wesley Longman (1999).
6. FIPA Agent Communication Language Specification. Foundation for Intelligent Physical Agents (2000).
7. FIPA RDF Content Language Specification. Foundation For Intelligent Physical Agents (2001).
8. Java Servlet Specification Version 2.3, `http://java.sun.com`.
9. Kazienko P., Kiewra M.: Link Recommendation Method Based on Web Content and Usage Mining. New Trends in Intelligent Information Processing and Web Mining Conference Proceedings, Advances in Soft Computing, Springer, to appear.
10. Kiewra M.: Web Management Using Users' Data and Activities. Wrocław University of Technology, M.Sc. Thesis (2002).
11. Langheinrich M., Nakamura A., Abe N., Kamba T., Koseki Y.: Unintrusive Customization Techniques for Web Advertising. Computer Networks 31 (11-16) (1999) 1259–1272.
12. Lin W., Alvarez S.A., Ruiz C.: Efficient Adaptive-Support Association Rule Mining for Recommender Systems. Data Mining and Knowledge Discovery 6 (1) (2002) 83–105.
13. Madria S.K., Bhowmick S.S., Ng W.-K., Lim E.P.: Research Issues in Web Data Mining. Lecture Notes in Computer Science 1676 Springer (1999) 303–312.
14. Mobasher B., Cooley R., Srivastava J.: Automatic personalization based on Web usage mining. CACM 43 (8) (2000) 142–151.
15. Mobasher B., Dai H., Luo T., Sun Y., Zhu J.: Integrating Web Usage and Content Mining for More Effective Personalization. EC-Web 2000, Lecture Notes in Computer Science 1875 Springer (2000) 156–176.
16. Perkowitz M., Etzioni O.: Adaptive Web sites. CACM 43 (8) (2000) 152–158.
17. Salton G., McGill M.J.: Intruduction to Modern Information Retrieval. McGraw-Hill Book Co. (1983).
18. Yao Y.Y., Hamilton H.J., Wang X.: PagePrompter: An Intelligent Agent for Web Navigation Created Using Data Mining Techniques. RSCTC 2002, Lecture Notes in Computer Science 2475 Springer (2002) 506–513.