In Proceedings ASAI 2000, Argentine Symposium on Artificial Intelligence (29 JAIIO) - Tandil, Argentina - pp. 29 - 41 - September 2000

The QueryGuesser agent

Silvia N. Schiaffino[1] and Analía Amandi

ISISTAN Research Institute, Facultad de Ciencias Exactas
Universidad Nacional del Centro de la Pcia. de Buenos Aires
Campus Universitario - Paraje Arroyo Seco - (7000) - Tandil, Bs. As., Argentina
[1] also CONICET
{sschia, amandi}@exa.unicen.edu.ar

Abstract. Querying a database via different kinds of Internet and Intranet access is generally a repetitive and time-consuming task. QueryGuesser is an intelligent agent with the capability of generating personalized queries to a database system, according to a particular user's interests. This agent observes a user behavior while he is querying a database and builds a model of his preferences with the information obtained in this process. QueryGuesser uses a technique that integrates Case-Based Reasoning and Bayesian Networks to gradually build a user profile. This profile is used to generate the most relevant queries in advance and offer him the information he needs in a timely manner.

Keywords: intelligent agents, Bayesian networks, case-based reasoning

1 Introduction

The Web is allowing broader access to vast amounts of information. The volume and variety of this information makes finding items of interest progressively more difficult. Filtering and managing this information is becoming an enormous challenge, even with the most advanced search tools [11].

An equally set of challenges, but one that has not yet become as widely recognized, exists in the area of structured and semistructured information sources. By structured information sources we mean sources that can be queried by using well-defined, general query languages, such as relational databases, object-oriented databases and knowledge bases. By semistructured information sources we include a variety of sources that contain sufficient structure to be treated as databases, even though they do not provide the full generality and power of a query language. In the context of WWW, this structure is provided in one or both ways: by an informal form-based query interface or by the presence of HTML markups and other textual markers used according to site specific conventions [21].

There are some reasons why structured and semistructured information sources will be rapidly increasing in importance in the Internet and Intranet worlds. One of them is that enterprises have large investments in databases, and naturally want to leverage these in the context of the WWW. One straightforward way to leverage existing databases is simply to make them accessible to employees via web interfaces.

In this context, a user who queries an online database regularly, is usually looking for information related to his interests. If a user's information needs belong to a specific set of subjects, queries made by a user in different opportunities will be similar. The task of querying a database becomes a boring and repetitive activity for such a user.

Besides, a user of such systems is always interested in getting the information he needs as soon as possible. Finding information on the Web via different kinds of Internet and Intranet access is generally a time-consuming task because of the current speed of network connections [13].

An alternative to assist a user when he queries a database using Internet or Intranet services could be giving him the information he needs before he asks the system for it. Then, the user will have relevant information available at the right time without having to look for it.

The idea of personal assistants supporting humans in their work has emerged in recent years. The primary thrust of this trend is that computer programs take over boring, repetitive and time-consuming tasks in order to increase human productivity and creativity [15]. Interface agents are computer programs that employ Artificial Intelligence techniques to provide active assistance to a user with computer-based tasks [20].

In this paper, we describe an intelligent agent that assists a user who often queries a database looking for the information he needs. QueryGuesser is an intelligent agent with the capability of generating personalized queries to a database system, according to a particular user's interests. This agent observes a user behavior while he is submitting queries to the database and records the features or attributes involved in these queries. Then he builds a model of the user preferences with the information obtained in this process. QueryGuesser uses a technique that integrates Case-Based Reasoning (CBR) and Bayesian Networks (BN) to gradually build a user profile. This profile is used to generate the most relevant queries in advance and offer him the information he needs in a timely manner. Because the agent performs the task autonomously, it saves the user's time.

CBR is a problem-solving paradigm that is able to utilize the specific knowledge of previously experienced, concrete problem situations: cases. Basically it solves a new problem by remembering a previous similar situation and by reusing information and knowledge of that situation [2]. Each query submitted by a user represents an experience that gives the agent information about a user's information needs. Queries made in previous situations offer some indications about what kind of queries would be relevant for the user in a similar new situation.

A BN is a graphical model for probabilistic relationships among a set of variables [14]. In this work, BN are used to model the qualitative and quantitative relationships among the different attributes used to perform the queries, as well as the relationships between different queries. This Bayesian model is built with information recorded in the form of cases.

A BN represents beliefs and knowledge about a particular class of situations. Given a BN for a class of situations and evidence about a particular situation in that class, conclusions can be drawn about that situation. QueryGuesser agent uses Bayesian inference mechanisms to derive the attributes a user would use to query the database.

Integrating both techniques the agent builds a user profile which consists of a ranked set of relevant inferred queries for a user. The user feedback is used to modify the user profile, in order to improve subsequent suggestions.

This paper is organized as follows. Section 2 describes the QueryGuesser agent focusing attention on its capabilities. Section 3 shows an example that explains QueryGuesser functionality and the use of each one of the techniques. Section 4 describes some related work. Finally, section 5 presents the conclusions of the paper and some future work.

2 QueryGuesser overview

The idea of personal assistants supporting people do their work has emerged in recent years. Interface agents are computer programs that employ Artificial Intelligence techniques to provide active assistance to a user with computer-based tasks [20].
In order to help users, agents need some knowledge about the task they have to perform and they have to be aware of the interests, habits and preferences of each user. Agents then use this knowledge to assist users in a range of different ways. The main application areas are in helping humans to cope with information overload and to help users in performing repetitive tasks.

The idea behind interface agents is that such agents can acquire the knowledge they need to assist users by themselves. Initially, agents are given a minimum of background knowledge and they learn appropriate behavior from users. In order to apply this learning approach, two conditions must be fulfilled. First, the use of the application has to involve a substantial amount of repetitive behavior. Second, this repetitive behavior is potentially different for different users. This approach is inspired by the metaphor of a personal assistant [20]. These agents become gradually more effective at achieving their goals, as they learn a user's interests, habits and preferences and adapt as they change over time.

QueryGuesser is an interface agent that assists a user who queries a database to get the information he needs. QueryGuesser automates the execution of queries to a database via Internet or Intranet access. The goal of the agent is detecting relevant queries for a user and suggesting him their execution at an appropriate moment. Besides, by performing these queries in advance the agent saves the user's time. This task matches the requirements of the learning approach. A user who queries an online database regularly, is usually looking for information related to his interests. Assuming that user actions are consistent and that a user's information needs belong to a restricted set of subjects, queries made by a user in different opportunities will be similar. As a consequence, the task of querying a database becomes a repetitive activity for such a user. On the other hand, different users will usually have different interests, so queries made by different users will be also different.

Our agent learns a user's information needs by observing the user and by acting according to his feedback. The agent observes a user behavior while he is querying a database. He records data obtained from this observation in order to learn the user's interests. These data consist of the attributes used as filters to perform a query, data about the moment in which the user made a query – date, hour – and information about the sequences of actions – previous queries – performed by a user.

The agent uses information obtained from observation to build the user's interest profile. A user profile is a representation of a user's information needs through which the agent should act upon some goals based on that profile. A user profile contains information about the types of queries generally made by a user. It contains information obtained by observing a user behavior, such as data involved in the query, date and time, query results, sequences of queries; previously suggested queries and user feedback related to those queries. A user profile consists of a ranked set of relevant inferred queries for a user. These suggested queries are represented as a set of features or attributes most frequently used by the user. The ranking is made considering the probability of each combination of attributes being used to query the database. A user profile also contains statistic information about most frequent attributes and most frequent values of attributes.

QueryGuesser then suggests the user the execution of certain relevant inferred queries according to his profile. Besides, the agent performs these relevant queries in advance. In this way, a user does not have to make them explicitly and does not have to wait for the answers.

The agent becomes more competent as his knowledge about a user's interests is incremented and refined over time. The user's feedback is used to refine this knowledge. A user can provide feedback regarding recommended queries. User feedback causes modifications in a user profile, by adjusting the relevance associated to suggested queries.

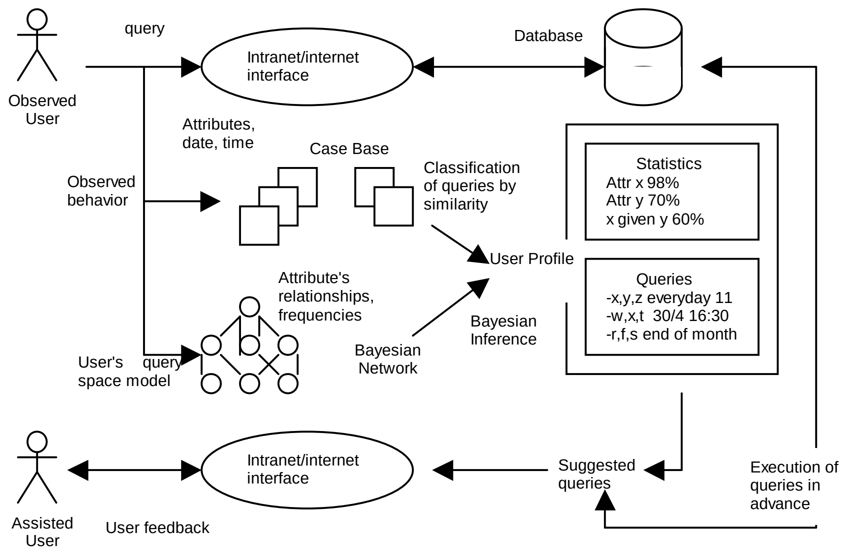

Fig. 1. General schema of QueryGuesser

QueryGuesser uses a technique that integrates CBR and BN to determine the types of relevant queries for a particular user and build his profile with this information. Figure 1 is a schema of QueryGuesser that shows the process that takes place since the user submits a query until he provides feedback to suggestions made by the agent.

2.1 QueryGuesser capabilities

In order to fulfill his task, QueryGuesser agent must have the capabilities of observing a user behavior, classifying queries, building a user profile, suggesting relevant queries, performing these queries in advance and processing user feedback.

- **Observation**: A learning agent acquires its competence by continuously "looking over the shoulder" of a user while he is performing actions [20]. An interface agent can monitor the activities of a user, keep track of all his actions over long periods of time, find regularities and recurrent patterns and offer to automate them. In order to suggest relevant queries to a user it is necessary to get information about his information needs. The agent observes a user behavior while he is submitting queries and records the attributes or keywords used by the user to perform each query. He records both the type of attribute (a general category to which it belongs) and its value. Assuming that a user is interested in certain topics, he will regularly use the same attributes or keywords to make his queries, or at least related ones. The amount of times an attribute or an attribute value is involved in queries executed by a user, gives the agent an idea of its relevance.
  The agent also records data related to the moment in which a query was executed, such as day number, day of week, month, season, hour, shift in case of an employee or worker, and so on. These data are important to detect repetitive patterns in user behavior and to determine which types of queries are executed in different situations.
- **Classification of queries**: In order to determine subjects of interest, the agent clusters similar queries by classifying them according to their characteristics. The agent makes two different classifications: one regarding attribute similarity and another one considering temporal similarity. Classification regarding attributes and attribute values is made by determining the similarity among different queries and grouping those which are related. Similarity metrics compare the correspondence between the types of attributes involved in the queries and the similitude of their values. Similar queries are assigned a label that identifies them as a topic of interest for a given user. Temporal categories concern the time of the year, the day of the week, if it is the beginning or the end of the month, and so on. Classification considering temporal aspects of queries is done in order to determine the user routine, i.e. the moments in which he queries the database. Using this information, the agent can suggest the right queries at the right time.
- **Building a user profile**: A user profile is a representation of the preferences of any individual user. A profile contains information about the types of queries frequently made a given user and the situations in which these queries are performed. It consists of both statistic and inferred information. The statistic part of the profile contains, for example, the occurrence frequency of each attribute in queries. Inferred information contains items that tell us, for example, the probability that a certain value of an attribute appears in a query given that a value of a related attribute appears too. The user's routine belongs to the inferred profile and comprises a series of situations in which a user makes queries. Each situation, in turn, has a set of queries associated to it. However, there can exist some queries that are not made in a particular situation, but at any moment.

- **Generating suggestions**: the user profile is used to generate suggestions to the user. Suggested queries are obtained combining attributes and attribute values inferred to be relevant for a user while building the user profile. Possible relevant queries that were generated via this mechanism are filtered using previously recorded queries made by the user.
- **Submitting suggested queries**: the agent gives the suggested queries to the component of the system in charge of executing queries, who translates them to a specific query language and executes them, showing the results afterwards.
- **Processing user feedback**: The user feedback is used to refine the agent knowledge. A user can provide feedback for the queries recommended by QueryGuesser. User feedback causes modifications in the user profile, by adjusting the relevance associated to suggested attributes and values.

3 Example

In order to visualize QueryGuesser functionality, this section describes an example in which a user, John Smith, frequently queries a database looking for information. John Smith is a professor at a University. This university has got a database that stores information about departments, careers, professors, students, courses, attendance to courses, students' marks, rooms, and so on. John Smith is a professor at the Computer Science Department and teaches Logic Programming to Computer Science Engineering students. After every class, he always sends some material to read to the ones who attended his class. He queries the database to get the students' information and then he sends them some homework via email. This is an example of a repetitive task in which a user performs repeatedly the same or similar query to get the information he needs. QueryGuesser can detect this behavior and automate the task for the user. In this way, when John Smith enters the system after a Logic Programming class, he will have a list of all the students who were present in his last class ready for him to send them an email.

3.1 Case-Based Reasoning

The QueryGuesser agent uses a technique that integrates CBR and BN to determine the types of relevant queries for a particular user and builds his profile with this information.

In CBR, a reasoner remembers previous situations similar to the current one and uses them to help solve the new problem. CBR considers reasoning as a process of remembering one or a small set of concrete instances or cases and basing decisions on comparisons between the new situation and old ones [17].

Since this kind of reasoning is based on previous experiences, queries made by a given user in past situations give us some information about the queries that the user would possibly submit to the database in a future time.

In the example we are developing, information about previous queries made by John Smith after a Logic Programming course gives the agent some clues about queries he would make in similar situations. As he always queries the database similarly, the agent can infer the user information needs. Considering queries made in past situations, the agent can suggest him the execution of particular queries at the right time.

Each query made by a user is represented in the form of a case. A case is an in-context piece of knowledge representing an experience, and any case worth recording in a case library teaches a lesson fundamental to achieve the goals of the reasoner who will use it [17]. Each query made by the user becomes a case that will help the reasoner to acquire information about the user's information needs. The attributes or features used to perform a query are very helpful for determining the topics the user is interested in.

A case has three main parts: the description of the situation or problem, the solution, and the outcome or results of applying the solution to the problem. In the chosen application domain, the description of the situation includes the attributes used to make a query (commonly named filters), the user goals, information about the user and data related to temporal aspects. This latter item is useful to determine the user's routine and includes information such as date and time when a query was performed. The solution is, in this example, a code that identifies a certain topic of interest. This code is determined by comparing the new query with previous recorded ones and is used to group together similar queries. The outcome describes in some way the query results. Figure 2 shows a case representing a query made by John Smith.

In order to detect different preferences of a user, cases are classified according to the similarity of the queries they represent. Similar queries are assigned a code that identifies them as part of the same topic of interest. Similarity metrics compare the type of attributes involved in the queries and their values, to verify their correspondence.

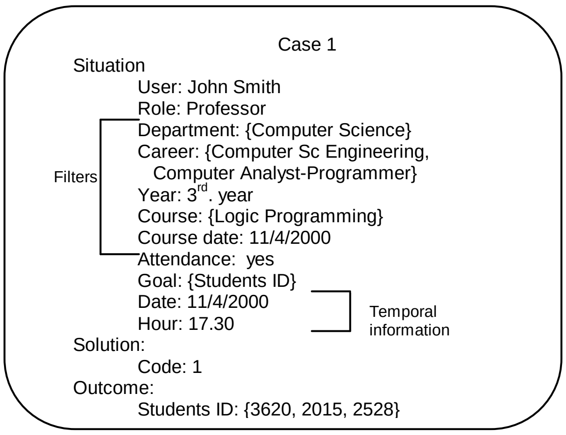

Fig. 2. A case representing a query

For example, if John Smith also taught Logic at the department of Mathematics, and he used the same methodology to send material to his students, he would make similar queries with different attribute values. Similarity metrics would associate a code to Logic Programming queries and a different one to Logic ones, because of the difference in the values of attributes such as department, career, course, course date.

Each metric computes a similarity function that gives as result a score representing how similar the queries are. If this score is higher than a threshold value specified by the metric, then the queries are similar and they are assigned the same code. If not, a new code is assigned to the new query.

There are also other metrics that compare temporal aspects of queries to determine a user's routine. For example, queries can be classified according to the similarity between the day and hour of execution (e.g. Tuesdays, 6:30), or according to the time of the month (e.g. beginning or end of the month).

3.2 Bayesian Networks

A BN is a compact, expressive representation of uncertain relationships among parameters in a domain [8]. In particular, a BN can model the relationships between the attributes involved in user queries and also between subsequent queries.

A BN is a directed acyclic graph that represents a probability distribution. Nodes represent random variables and arcs represent probabilistic correlation between the variables. Conditional probability tables specify quantitative probability information. For each node, a table specifies the probability of each possible state of the node given each possible combination of states of its parents. Tables for root nodes just contain unconditional probabilities [12].

In our technique, BN are used to model the information needs of a given user. Each node represents an attribute or feature used by the user to query a database. Arcs correspond to existing relationships between attributes used as filters. These relationships are domain dependent. Probability values are obtained by determining how many times a certain attribute is involved in the user queries. The frequency in which each attribute or feature appears in the queries submitted by a particular user, represents the importance of that feature for the user. This frequency is obtained analyzing the cases stored in the case base.

In the example application domain, we can point out some relationships between features that reflect dependencies between them. For example, for each department the user can ask about careers belonging to it. Each career has certain courses, students and professors. These restrictions can be modeled using BN by establishing relationships between the correspondent nodes. These relationships model the probability that a child node (e.g. a career) is used as filter given that a parent node is used as a filter too (e.g. a department).

Generally, a domain expert determines relationships between attributes, but they can also be learned as a user makes queries applying learning algorithms [14]. Relationships between attributes and the associated conditional probabilities can give the agent information about, for example, the amount of times an attribute corresponding to a course is involved in a query when an attribute representing a particular career is also involved.

A BN is built gradually as the user queries the database. When a user submits a query, the query is stored as a case and a node is added to the network for each attribute involved in the query. Arcs are drawn between the correspondent nodes, considering the relationships established for the particular domain. Probability values are updated as attributes frequencies in queries are modified with each new query.

Figure 3 shows an example of a BN that models John Smith's query space in the example application domain. Each variable can have only two values: true representing that the attribute is present in the query and false, indicating that the attribute is absent in the query.

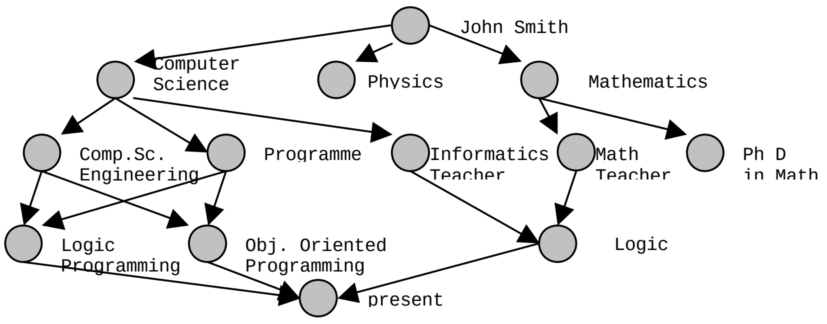

Fig. 3. BN representing the query space model

The model is completed by establishing the probability values associated to each node of the graph. Simple probabilities are those associated to variables that do not depend on other variables (except on the user); departments in our example. Table 1 shows an example of simple probabilities associated to departments.

Table 1. Simple probability values

	Computer Science	Mathematics	Physics
John Smith	0.6	0.2	0.2

The previous values indicate that 60% of queries made by John Smith are related to Computer Science Department, 20% to Mathematics Department and 20% to Physics Department. Table 2 shows an example of conditional probability values.

Table 2. Conditional probability values

	Logic Prog OO Prog	¬Logic Prog OO Prog	Logic Prog ¬OO Prog	¬Logic Prog ¬OO Prog
Comp Sc Eng.	0.375	0.5	0.125	0
¬Comp Sc Eng	0	0	0	0

The cell whose value is 0.375 means that in queries made by John Smith where the Computer Science department appears, the Logic Programming and Object Oriented Programming courses will also appear with a probability of 37,5%.

3.3 Integration of both techniques

Integration of BN and CBR can be done with either of the methods as the master and the other as the slave, depending on which method uses information provided by the other [9]. The technique described in this paper uses CBR as the slave. Cases recorded in the case base are used to calculate the probability values associated with each node of the BN.

Efficient inference algorithms exist for deriving answers to queries given a probability model expressed as a BN [8]. Considering the relationships between features modeled by the network, these inference mechanisms are used to derive which the most probable queries are. Cases are used to filter queries suggested using Bayesian inference mechanisms.

A user profile is built with the information stored in the Case Base and information stored in the BN. A user profile contains statistic data about queries made by the user and inferred information about his topics of interest. Statistic information includes items such as: most frequent queried attributes, for each attribute it contains the most frequent queried values, most frequent goals, most frequent requested query codes, most frequent values of an attribute given values of another attributes. Figure 4 shows an example of John Smith statistic profile.

```
                      Statistic Profile
- Most Freq. Attr.
          Department            100%
          Career                80%
          Course                80%
- Most Freq. Values
          Department: Computer Sc        70%
          Career: Comp. Sc. Eng.         100%
                    Programmer           100%
          Course: Logic Prog    50%
                    OO Prog              87%
```

Fig. 4. Statistic profile

Inferred information contains items such as relevant inferred values, relevant inferred values given a value of another attribute, combinations of relevant inferred attributes. Figure 5 shows an example of John Smith inferred profile.

```
                              Inferred Profile
- Most relevant indep. attr.               - Suggested Queries
  Department: Comp Sc. 70%                 {Comp Sc, Comp Sc Eng,Logic Prog,present}
- Most relevant dep. attr.                 {Comp Sc, Comp Sc Eng, OO Prog, present}
  Career/department
         Comp. Sc. Eng/Comp Sc  100%
         Programmer/Comp Sc     100%
  Course/career
         Logic Prog/Comp Sc Eng 50%
         OO Prog/ Comp Sc Eng   87%
  Attendance/Course
         Present/Logic Prog    100%
```

Fig. 5. Inferred profile

This example does not consider the user's routine. In order to take into account the user's routine the agent has to discover first the most frequent query situations and then for each situation analyze attributes and attribute relationships.

The most important item of the inferred profile is Suggested Queries. Suggested queries are formulated combining attribute values that were inferred to be relevant for the user. An independent attribute (one whose node does not have parents, but the user name) is relevant if its simple probability value is higher than a specified threshold. To determine the importance of a dependent value, the agent first sets as evidence relevant values of its parents. Then, via inference Bayesian mechanisms it determines which of the values of the child attribute have higher probability values. Combining the obtained values for each attribute the agent build possible queries. Possible queries are filtered using cases in order to suggest only queries that make sense for the user.

4 Related work

Related work considering agents, includes a variety of software agents developed to assist users. The set of tasks or applications an agent can assist the user with is virtually unlimited: automating desktop tasks [15], mail management, meeting scheduling, selection of books, movies, music, and other forms of entertainment [20], electronic commerce [10], and specially information filtering and information retrieval [6, 16, 19, 22]. A system called FallQ [18] supports users in a telecommunications company. QueryGuesser agent helps users in an unexploded domain, structured information sources.

Related work considering the technique used by QueryGuesser involves other researches. CBR and BN can be integrated in different ways and its integration can be applied in a variety of domains. There are some integrated systems, which include:

- Microsoft Research has developed two prototypes for a fault diagnosis task called Aladdin. These systems use three-layer BN to store information about failures in Microsoft Word and Microsoft Windows NT. They link causes of cases (i.e. issues) with observable symptoms. The different levels describe, respectively: one or more causes, i.e. factors that contribute to the error; the fact that occurs if these conditions are met; and one or more observable symptoms caused by this fact. A repair plan is stored with each cause. An expert builds the BN and it is updated every time it is used [5].
- Croft and Turtle [7] address document retrieval, where they evaluate the precision and recall of retrieved documents given queries. Each document is considered as a case and queries take the form of word stems. They use a BN to link documents with term nodes and queries. The probabilities of a term node given a case reflects that word stem's frequency in the case, while a query is simply connected to its word stems.
- In [9] consists of a thesis which investigates the integration of algorithms for Data Mining (DM) and CBR. As an example, it is described an integrated CBR-DM system. They implemented a prototype that uses a BN to compute similarity metrics during retrieval of cases in a CBR system. The algorithm is called CBRDM.
- In [3] it is proposed an approach to knowledge intensive CBR. Explanations about why two cases are similar are generated from a domain model consisting of a combination of a semantic network and a BN. The domain model is used to support the processes of retrieval and reuse of past cases.

- In [1] it is described an integration of BN and CBR in an architecture called INBANCA. Both techniques work cooperatively on multiagent planning tasks in a soccer environment. BN are used to characterize action selection and CBR is used to determine how to implement actions.

Other integrated systems have also been developed. Some of them are described in the surveys that can be found in [1, 3, 9].

5 Conclusions and future work

In this paper, we describe an intelligent agent with the capability of generating personalized queries according to a user's information needs. The agent builds an interest profile for each user and makes relevant queries for them in advance. In this way, users have the right information available at the right time.

The agent has been implemented in Java. JavaLog [4], a Prolog engine implemented in Java, has been used to manage knowledge represented in the form of cases. The current version of the agent is being used and tested by different users. The chosen application domain to test QueryGuesser is the part of a LIMS (Laboratory Information Management System) in charge of sample tracking.

Contributions of this paper include the application of a technique that integrates CBR and BN to manage information stored in database on behalf of a user.

Future work includes detecting the behavioral pattern of a user when he queries the database. By recording information about the date and time of each query, the agent could schedule the relevant queries and offer a user the information he needs at an appropriate moment.

So far, relationships between parameters in the model are pre-established by a domain expert. Algorithms for learning in BN [14] could be added in order to detect new relationships among attributes.

References

1. Aha, D., Chang L. W. : Cooperative Bayesian And Case-Based Reasoning for Solving Multiagent Planning Tasks - Technical Report - Navy Center for Applied Research in Artificial Intelligence (NCARAI), Washington DC, USA - Number AIC-96-005 - (1996)
2. Aamodt, A., Plaza E.: Case-Based Reasoning: Foundational Issues, Methodological Variations and System Approaches. AI Com - Artificial Intelligence Communications, v. 7, n.1., (1994) 39-59
3. Aamodt, A., Langseth, H.: Integrating Bayesian networks into knowledge-intensive CBR. In American Association for Artificial Intelligence, Case-based reasoning integrations; Papers from the AAAI workshop. Technical Report WS-98-15. AAAI Press - (1998) 1-6
4. Amandi, A., Zunino, A., Iturregui R.: Multi-paradigm languages supporting Multi-Agent Development - MultiAgent System Engineering - 9th European Workshop on Modeling Autonomous Agents in a Multiagent World - MAAMAW 99 - Lecture Notes in Artificial Intelligence, LNAI 1647 - Springer Verlag - (1999) 128 - 139
5. Breese, J., Heckerman D.: Decision-Theoretic Case-Based Reasoning - In Proceedings of Fifth International Workshop on Artificial Intelligence and Statistics - (1995) 56 - 63

6. Chen, L., Sycara K.: WebMate: A Personal Agent for Browsing and Searching - (1997)
7. Croft W.B., Turtle H.R.: Retrieval Strategies for hypertext - Information Processing and Management, (1993) 29, 313-324
8. D'Ambrosio, B: Inference in Bayesian Networks - AI Magazine, SUMMER - (1999) 21 - 35
9. Dingsoyr, T.: Integration of Data Mining and Case-Based Reasoning - Master Thesis - Department of Computer and Information Science - Norwegian University of Science and Technology (1998)
10. Etzioni, O.: Moving Up the Information Food Chain. Deploying Softbots on the World Wide Web - AI Magazine - Volume 18, Nº 2 - Intelligent Systems on the Internet - (1997) 11-18
11. Gordon, E.: Verity Agent Technology: Automatic Filtering, Matching and Dissemination of Information – In Proceedings of the Practical Application of Intelligent Agents and Multiagent Technology 1997 - PAAM 97 - London, UK (1997) 287-301
12. Haddawy, P: An overview of Some Recent Developments in Bayesian Problem-Solving Techniques - Introduction to this Special Issue - AI Magazine, SUMMER - (1999) 11-19
13. Han, Y., Sterling, L.: Agents for Citation Finding on the World Wide Web - In Proceedings of the Practical Application of Intelligent Agents and Multi-Agent Technology 1997 – PAAM 97 - London, UK (1997) 303 - 318
14. Heckerman, D.: A tutorial on Learning with Bayesian Networks - Technical Report MSR-TR-9506 - Advanced Technology Division, Microsoft Research - (1995)
15. Hoyle, M., Lueg, C.: Open Sesame!: A Look at Personal Assistants – In Proceedings of the Practical Application of Intelligent Agents and Multi-Agent Technology 1997 – PAAM 97 - London, UK (1997) 51 - 60
16. Joachims, T., Freitag D., Mitchell T.: WebWatcher: A tour Guide for the World Wide Web - In 15th International Joint Conference on Artificial Intelligence, IJCAI - (1997).
17. Kolodner, J.: Case-Based Reasoning - Morgan Kaufmann Publishers (1993)
18. Kunze, M., Hübner, A.: CBR on Semi-structured Documents: The Experience Book and the FallQ Project - In Proceedings 6th German Workshop on CBR - (1998).
19. Lieberman, H.: Autonomous Interface Agent - In Proceedings of the ACM Conference on Computers and Human Interface - (1997).
20. Maes , P.: Agents that Reduce Work and Information Overload - Communications of the ACM (1994) -
21. Martin, D., Oohama, H., Moran, D., Cheyer, A.: Information Brokering in an Agent Architecture - In Proceedings of the Practical Application of Intelligent Agents and Multiagent Technology 1997 - PAAM 97 - (1997) 467 - 486
22. Pazzani, M., Billsus, D.: Learning and Revising User Profiles: The Identification of Interesting Web Sites - In Machine Learning 27, 313-331 - June 1997