



ELSEVIER

Information Processing and Management xxx (2004) xxx–xxx

www.elsevier.com/locate/infoproman

**INFORMATION
PROCESSING
&
MANAGEMENT**

A Bayesian network approach to searching Web databases through keyword-based queries

Pável Calado ^{a,*}, Altigran S. da Silva ^b, Alberto H.F. Laender ^a,
Berthier A. Ribeiro-Neto ^a, Rodrigo C. Vieira ^a

^a Department of Computer Science, Federal University of Minas Gerais, Belo Horizonte, MG, Brazil

^b Department of Computer Science, Federal University of Amazonas, Manaus, AM, Brazil

Received 3 March 2004; accepted 3 March 2004

Abstract

On-line information services have become widespread in the Web nowadays. However, Web users are non-specialized and have a great variety of interests. Interfaces for Web databases must, therefore, be both simple and uniform. In this paper, we present a solution for querying Web databases using keywords only. A Bayesian network model is used to generate a set of one or more plausible structured queries derived from the initial user input. These queries can then be submitted to Web databases and the retrieved results presented as a set of ranked answers. To structure the user queries, full access to the database is not required. Instead, only a small portion of its content, extracted through a public Web interface, is used by the network model. This approach not only reduces the complexity of existing on-line interfaces, but also offers a solution to the problem of querying several distinct Web databases with a single interface.
© 2004 Published by Elsevier Ltd.

Keywords: Web databases; Bayesian networks; Query structuring

1. Introduction

On-line information services, such as on-line stores and digital libraries, have become widespread in the Web, nowadays. Such services allow a great number of users to remotely access data stored in a local database, also called a *Web Database*. Web users, however, are often non-specialized and their interests vary greatly. Thus, two important problems are posed to designers of interfaces for Web databases. First, interfaces are expected to be simple, since they are intended for layman users. Second, if an on-line service is to provide access to different types of information, its interface should be as uniform as possible, otherwise, users will be required to learn how to use a different interface for each distinct database.

* Corresponding author.

E-mail addresses: pavel@dcc.ufmg.br (P. Calado), alti@dcc.fua.br (A.S. da Silva), laender@dcc.ufmg.br (A.H.F. Laender), berthier@dcc.ufmg.br (B.A. Ribeiro-Neto), rcvieira@dcc.ufmg.br (R.C. Vieira).

28 The most common solution for implementing on-line services that access Web databases is the use of
29 customized forms, navigation menus, and similar browsing mechanisms. Although useful in some cases,
30 this approach has important shortcomings. Web sites that provide access to multiple databases, like
31 *Amazon* (<http://www.amazon.com>), *MySimon* (<http://www.mysimon.com>), or *Travelocity*
32 (<http://www.travelocity.com>), include dozens of different forms, one for each type of product,
33 each being composed of a large number of fields. From the point of view of a Web user, this type of
34 interface might seem rather complex. From the point of view of a Web developer, it increases the devel-
35 opment time and maintenance costs.

36 Another common inconvenient of traditional Web database interfaces is the fact that the answer set to a
37 query is frequently very large. In a traditional database system, appropriate tools and query languages are
38 available to restrict the search results. Such methods, however, are not usually available in Web database
39 interfaces. In Web search engines, document ranking (Baeza-Yates & Ribeiro-Neto, 1999) is the most used
40 method to deal with this problem. We argue that query interfaces for Web databases should also provide
41 similar resources to help the user handle large answer sets.

42 In this work, we propose the use of keyword-based querying, as in Web search engines, as an alternative
43 to the use of interfaces based on multiple forms. Additionally, we propose to rank the possibly large
44 number of answers returned according to a relevance criterion. A Bayesian network model is used to derive
45 structured queries from a set of keywords specified by the user. This network uses information from the
46 Web database to determine the probability of a structured query fitting the user's needs. Since the database
47 may be accessible only through a Web interface, an algorithm is proposed that can collect information, via
48 Web, and store it locally, to be used by the network model as a sample of the database content. After
49 structuring, queries are submitted and a second Bayesian network model is used to rank the objects re-
50 turned according to the probability of being relevant to the user.

51 Our approach implies that there is no need for the user to know the exact schema of the underlying
52 database being queried. Thus, an interesting feature is that it allows a single search-engine-like text box to
53 be used for querying several distinct databases. The user needs only to fill the search box to obtain the
54 results. In sum, our approach is able to provide on-line services with (1) a search engine-like interface that is
55 simple and intuitive to Web users and (2) the possibility of querying several heterogeneous databases using
56 a single interface. Further, by ranking the structured queries based on the user's input, the system is also
57 able to estimate which of the databases should be queried.

58 Experiments performed on seven different databases indicate that the proposed models are able to
59 accurately structure user queries and return correct answers with a minimum intervention from the user.
60 For a set of 266 unstructured queries, all correct answers are retrieved with a tolerable interference of
61 spurious results. Up to 95% of the time, one of the top three resulting structured queries is the proper one.
62 When the user interacts with the system, selecting the query to be processed among the top three ranked
63 alternatives, most correct answers are retrieved with minimal interference, presenting average precision
64 figures close to 90%. Also, by using the network model to structure the same query for the seven different
65 databases, the system is able to correctly determine the database to be queried 88% of the time. All this
66 results are achieved by using only a small sample of the objects in the database.

67 This paper expands the work presented in Calado, da Silva, Vieira, Laender, and Ribeiro-Neto (2002) by
68 using the query structuring process with different databases simultaneously and expanding the experiments
69 with a grater set of queries and databases. Further, we propose the use of only a part of the database as a
70 source of information for the query structuring process and an algorithm is presented to select the infor-
71 mation to be used.

72 The remainder of this paper is organized as follows. Section 2 discusses prior related work. Section 3
73 presents an overview of our approach. Section 4 gives a brief introduction to Bayesian networks and de-
74 scribes the models used to structure queries and rank the returned objects. Section 5 presents the results of
75 the experiments performed. Finally, Section 6 presents our conclusions and suggests some future work.

76 **2. Related work**

77 The use of keywords for query formulation is a rather common resource in information retrieval systems
78 (Baeza-Yates & Ribeiro-Neto, 1999). In the context of structured databases, however, only recently pro-
79 posals for the use of keyword-based queries have appeared (Agrawal, Chaudhuri, & Das, 2002; Dar, Entin,
80 Geva, & Palmon, 1998; Florescu, Kossmann, & Manolescu, 2000). The DataSpot system, described in Dar
81 et al. (1998), proposes the use of “plain language” queries and navigations to explore a *hyperbase* for
82 publishing content of a database on the Web. Agrawal et al. (2002) introduce a system that allows querying
83 databases through keywords. Their work, however, focuses on relational databases and does not provide
84 any ranking for the retrieved answers. Differently, the work described in Florescu et al. (2000) proposes the
85 extension of XML-QL, a well known query language for XML, with keyword based searching capabilities.
86 Our approach is distinct since it adopts a much simpler query language. Although this can make our
87 solution less expressive, it also makes it more accessible to regular Web users. Most important, we propose
88 a Bayesian network model to rank the answers, thus avoiding the effects of spurious data, which will
89 necessarily be present.

90 The vector space model has been the most widely used solution to the problem of ranking query results
91 in text databases (Salton & McGill, 1983). This model is also employed in the work described in Cohen
92 (1999) to determine the similarity between objects, in a database composed of relations, whose attribute
93 values are free text. For ranking the objects returned as answers to queries, the work described in Chau-
94 dhuri and Gravano (1999) combines application-oriented scoring functions. An extension of this idea is
95 proposed in Bruno, Gravano, and Marian (2002) for the case of Web accessible databases. In this case the
96 scoring functions are based on evidences coming from distinct Web databases. In Goldman, Shivakumar,
97 Venkatasubramanian, and Garcia-Molina (1998), object ranking is based on a measure of the proximity
98 between objects, that are seen as nodes, in the database, that is seen as a graph. This approach leads to a
99 search system similar to ours in functionality, since it also allows keyword-based queries, but based on a
100 rather distinct approach.

101 In this work, experiments are performed using a sample of the content of database to be queried, instead
102 of the full database. This sample can be built by collecting information from different Web sources, as in
103 Doorenbos, Etzioni, and Weld (1997), using general data extraction tools (Laender, Ribeiro-Neto, & da
104 Silva, 2002; Liu, Pu, & Han, 2000). Once the data is collected, it needs not to be stored in a fully structured
105 manner, but can be a simple list of attribute-value pairs, to be used as a knowledge base for the query
106 structuring system. The sample data used in this work can, thus, be seen as a simplified form of an ontology
107 (Swartout & Tate, 1999).

108 In our work, query structuring and the ranking of query results is performed using vectorial techniques,
109 within a Bayesian network framework. Bayesian network models were first used in IR by Turtle and Croft
110 (1990). Their model takes the viewpoint that the observation of a document induces belief on its set of index
111 terms, and that the specification of such terms induces belief in a user query, or information need. Later, a
112 different model was proposed by Ribeiro-Neto and Muntz (1996), in which the elements of an IR system are
113 formally defined as concepts in a sample space. Their work also demonstrates that the combination of
114 evidence from past queries with the vector space ranking yields better results than the use of a vector space
115 ranking alone. More recently, Acid, de Campos, Fernández-Luna, and Huete (2003) presented a model
116 whose network topology is defined in such way that an exact propagation algorithm can be used to effi-
117 ciently compute the relevance probabilities of the documents. When compared to Turtle and Croft’s work
118 for the task of document ranking, this model shows better performance in four out of five reference col-
119 lections.

120 Bayesian networks have also been applied to other IR problems besides ranking as, for example, rele-
121 vance feedback (Haines & Croft, 1993), automatic construction of hypertext (Shin, Nam, & Kim, 1997),

122 query expansion (de Campos, Fernández-Luna, & Huete, 1998), information filtering (Callan, 1996), and
123 document classification (Dumais, Platt, Heckerman, & Sahami, 1998).

124 In this paper, we adopt the Bayesian framework proposed by Ribeiro-Neto and Muntz (1996) for
125 modeling structured queries and ranking the results of querying Web databases.

126 3. The query structuring framework

127 This section presents an overview of our query structuring approach. To simplify the discussion, we start
128 with some basic definitions. Following, we proceed to describe the process of generating structured queries
129 and obtaining the results from a Web database.

130 3.1. Basic concepts

131 Consider a database D accessible through a Web query interface (for instance, an HTML form). We
132 define this database as a collection of *objects*, $D = \{o_1, o_2, \dots, o_n\}$, $n \geq 1$. Each object o_i is a set of *attribute-*
133 *value* pairs $o_i = \{\langle A_1, v_{1i} \rangle, \dots, \langle A_{k_i}, v_{k_i i} \rangle\}$, $k_i \geq 1$, where each A_j is an attribute and each v_{ji} is a value
134 belonging to the domain of A_j . We note that the attributes do not need to be the same for all objects.

135 For some attributes, instead of a single value, we may have a list of values. For instance, in an object
136 representing a movie, the attribute *actor* might be a list of names. To represent this using our notation, we
137 allow the same attribute to appear several times. Thus, if attribute A_j , in object o_i , has n different values, we
138 can represent object o_i as $o_i = \{\dots, \langle A_j, v_1 \rangle, \langle A_j, v_2 \rangle, \dots, \langle A_j, v_n \rangle, \dots\}$. Attributes containing several values
139 are called, *value list* attributes.

140 We define a *database schema* as the set of all attributes that compose any of the stored objects. Thus, the
141 schema of a database D is defined as $S_D = \{A | A \text{ is an attribute of some object } o \in D\}$.

142 We define an *unstructured query* U as a set of keywords (or *terms*) $U = \{t_1, t_2, \dots, t_k\}$. As for an object, a
143 *structured query* Q is defined as a set of ordered pairs: $Q = \{\langle A_1, v_{1q} \rangle, \dots, \langle A_m, v_{mq} \rangle\}$, $m \geq 1$, where each A_j is
144 an attribute and each v_{jq} a value belonging to the domain of A_j .

145 This simplified definition of a database allows us to ignore the details of how its structure is internally
146 represented. We can regard the database simply as data repository, available through some high level user
147 interface, which can be based on a relational table, a set of relational tables, an XML repository, or any
148 other data model.

149 Throughout the text, we informally use the term *application domain* of a database to refer to the main
150 topic associated with the objects in the database. For instance, a database with the application
151 domain Book stores objects with information on books. The database to be queried by the user is called
152 the *target database*. As an example, consider a database D with the application domain Book. An object
153 o in D could be $o = \{\langle \text{Title, "I, Robot"} \rangle, \langle \text{Author, "Isaac Asimov"} \rangle\}$. An example of an unstructured query
154 is $U = \{\text{"Asimov"}, \text{"Robot"}\}$. An example of a structured query is $Q = \{\langle \text{Author, "Asimov"} \rangle,$
155 $\langle \text{Title, "Robot"} \rangle\}$.

156 3.2. The querying process

157 The querying approach proposed in this work has four basic steps: (1) inputting the unstructured user
158 query, (2) building a set of candidate structured queries, derived from the unstructured query, (3) selecting
159 one, or possibly more, of the candidate structured queries as being the “best” ones, and (4) processing the
160 results of these selected queries. We now explain each of these steps in detail.

161 Step 1 consists of receiving the unstructured query from the user, as a set of terms. This can be easily
162 accomplished using a simple search box interface, in which queries are specified by entering the appropriate



Fig. 1. Interface for the query structuring system.

163 keywords, as shown in Fig. 1. This process is very familiar to Web users and independent of the database to
164 be queried.

165 In Step 2, for a given unstructured query $U = \{t_1, t_2, \dots, t_n\}$, we need to determine a set P_U of possible
166 structured queries, called *candidate queries*. To this effect, let D be a database, with a schema S_D . A simple
167 way to determine P_U is to build all possible combinations of each query term $t_i \in U$ and each attribute
168 $A_j \in S_D$. Clearly, in the worst case, this operation would be of exponential complexity. In practice, however,
169 it is possible to discard many term-attribute pairs in advance. For instance, if there are little or no
170 occurrences of the word “Hamlet” in the field Author, we can discard all possible queries that make such an
171 assignment.

172 Once we have determined a set $P_U = \{Q_1, \dots, Q_n\}$, we proceed to Step 3, which consists of selecting the
173 candidate queries that are most likely to match the user needs. More precisely, for every query in P_U , we
174 determine the probability of its attributes corresponding to objects in D . The best candidate structured
175 query is the one that maximizes the likelihood that its component term-attribute pairs correspond to the
176 database content. For instance, the unstructured query $U = \{\text{“Asimov”}, \text{“robot”}\}$ might have as the most
177 probable structured query $Q_1 = \{\langle \text{Author}, \text{“Asimov”} \rangle, \langle \text{Title}, \text{“robot”} \rangle\}$.

178 For the same user query, Step 2 can also be applied to several databases with different application
179 domains. In this case, P_U will contain all candidate queries, each associated with the corresponding data-
180 base. We can, again, apply Step 3 to P_U and sort all the queries according to their computed probabilities.
181 Thus, together with the rank of the candidate queries, we now also have a rank for the databases, according
182 to the likelihood of each one being the database intended by the user. This is an extension of the work
183 proposed in Calado et al. (2002) that allows the system to automatically determine where to submit the
184 user’s query.

185 Finally, in Step 4, we take the ranking of structured queries in P_U and submit one of the top ranked ones
186 to the database to be queried. We can, for instance, pick the highest ranked query and submit it, without
187 user interference, or present the top ranked queries to the users and let them choose one for processing.
188 Alternatively, we can submit several queries and merge the incoming results. This alternative, however, was
189 not explored in this work, being left for future experiments. In all cases, a structured query Q is submitted to
190 the target database D and a set of returned objects RD_Q is obtained.

191 The objects in RD_Q are also ranked according to a probability that they satisfy the user’s needs, as we
192 discuss in Section 4.3. Ranking the objects in RD_Q is especially important when the query Q can be sent to
193 more than one target database and the results merged together. For instance, in an application domain like
194 Computer Science Bibliographic References, the same unstructured query can be used to search the ACM
195 and IEEE digital libraries. Notice that this is accomplished without the need to define a unifying database
196 schema.

197 To rank the candidate queries from set P_U in Step 3, and the returned objects from set RD_Q in Step 4, we
198 adopt the framework of Bayesian belief networks (Pearl, 1988). Belief networks provide a flexible and
199 formally sound framework, which can be conveniently adapted to combine distinct sources of evidence. In
200 Section 4.2, we discuss the Bayesian model proposed for ranking candidate queries. In Section 4.3, we
201 present the Bayesian model for ranking the retrieved objects.

202 3.3. Building a sample database

203 In the work presented in Calado et al. (2002), to determine the probabilities required to rank the
204 candidate queries in P_U , we need access to the objects in the target database D . However, for most on-
205 line services, the target database might be available only through its Web interface. Thus, it may be
206 necessary to build a local database, containing objects representative of the application domain in
207 question. This local database, called the *sample database*, can be, for instance, a subset of the target
208 database and does not need to be rigidly structured, but can be simply a list of possible values for each
209 attribute. In this work, we assume that only a sample database, instead of the whole target database, is
210 available to the query structuring mechanism. We now describe how a sample database can be gen-
211 erated.

212 In the context of this work, there are only two requirements for a sample database: (1) it must contain a
213 representative part of the target database and (2) it should be as small as possible. Without the first
214 requirement our framework would evidently be unable to produce useful results. Requirement two is
215 especially important for multiple application domain environments, in which the size of the local database
216 should be minimal for efficiency reasons. A way to achieve both requirements is to extract from the target
217 database a subset of objects that can accurately answer the most frequent queries submitted by users. This
218 information can, usually, be found in server logs and is used as the basis for the simple algorithm here
219 presented to generate a sample database.

220 The sample database generation algorithm proposed is described in Fig. 2. It assumes that a log
221 indicating all queries submitted during a given period is available and that we have access to the target
222 database only through its Web interface. Parameter L represents the set of queries present in the
223 available query log (including repetitions). Parameters, p and v are used to control the size of the sample
224 database generated. The value of p is the percentage of queries in the log that will be used to query the
225 target database. The value of v indicates how many objects should be retrieved by each submitted query.
226 Variable S represents the set of objects in the sample database. Using these parameters, the algorithm
227 tries to compose the sample database with values obtained from the target database, using the queries
228 contained in the log.

229 In experiments performed, for which details will be presented in Section 5, p was set to 0.9 and v was
230 adjusted in order to make the sample database at most 10% the size of the target database. We call attention
231 to the convenience of such strategy since we do not need to have access to the internal details of how target
232 databases are implemented or structured.

```
1: GenerateSample( $p, v, L$ )
2: Let  $K$  be the  $p \times |L|$  most frequent queries from  $L$ 
3:  $S \leftarrow \emptyset$ 
4: for all  $K_i \in K$  do
5:   Submit  $K_i$  to be processed by the remote database
6:    $O_v \leftarrow v$  first objects retrieved by  $K_i$ 
7:    $S \leftarrow S \cup O_v$ 
```

Fig. 2. Sample database generation algorithm.

233 **4. Ranking queries and answers**

234 Given a set of candidate structured queries, we should be able to determine which are the most likely to
 235 be the queries intended by the user. In this section, we describe the Bayesian network model to determine
 236 the structured queries that best match the content of the database. Following, we present the model used to
 237 rank the objects returned as answers to the selected structured query. We start with a brief introduction on
 238 Bayesian networks and the basic model chosen for our solution.

239 *4.1. Bayesian networks*

240 In this work, Bayesian networks provide a formal framework for the query structuring and result
 241 ranking mechanisms. Our approach is based on the model proposed by Ribeiro-Neto and Muntz (1996),
 242 which takes an episte-mological view of the information retrieval problem, interpreting probabilities as
 243 degrees of belief devoid of experimentation. For this reason, they are also called *belief networks*.

244 Bayesian networks provide a graphical formalism for explicitly representing independencies among the
 245 variables of a joint probability distribution. The probability distribution is represented through a directed
 246 acyclic graph whose nodes represent the random variables of the distribution. Thus, two random variables,
 247 X and Y , are represented in a Bayesian network as two nodes in a directed graph. An edge directed from Y
 248 to X represents the influence of the node Y , the *parent* node, on the node X , the *child* node. The intensity of
 249 the influence of variable Y on variable X is quantified by the conditional probability $P(x|y)$, for every
 250 possible set of values (x, y) .

251 According to the model proposed in Ribeiro-Neto and Muntz (1996), queries, documents and keywords
 252 are seen as events. These events are not independent, since, for instance, the occurrence of a term will
 253 influence the occurrence of a document. Using a Bayesian network, we can model these events and their
 254 interdependencies. Considering the fact that both documents and queries are composed of keywords, we
 255 can model a document retrieval problem using the network in Fig. 3.

256 In this network, each node D_j models a document, the node Q models the user query, and the K_i nodes
 257 model the terms in the collection. The vector \vec{k} is used to refer to any of the possible states of the K_i root
 258 nodes.

259 The similarity between a document D_j and the query Q can be interpreted as the probability of document
 260 D_j occurring given that query Q has occurred. Thus, using Bayes' law and the rule of total probabilities, we
 261 compute the similarity $P(D_j|Q)$ as:

$$P(D_j|Q) = \eta \sum_{\vec{k}} P(D_j|\vec{k})P(Q|\vec{k})P(\vec{k}) \tag{1}$$

263 which is the generic expression for the ranking of a document D_j with regard to a query Q . To represent any
 264 of the traditional IR models, using the network in Fig. 3, we need only to define the probabilities
 265 $P(d_j|\vec{k})$, $P(q|\vec{k})$, and $P(\vec{k})$ appropriately, as demonstrated in Ribeiro-Neto and Muntz (1996).

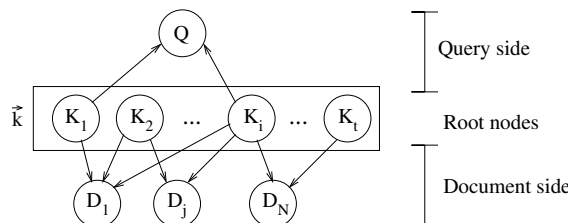


Fig. 3. Belief network for a query Q composed of the keywords K_1 and K_l .

266 Using a similar approach, we can build Bayesian network models to both select the structured query
 267 most likely to correspond to a given database, and to rank the final query results. In the following sections,
 268 we present two of such models.

269 4.2. Finding the best structured queries

270 Given an unstructured query, it most likely that several distinct structured queries can be derived from it.
 271 Thus we need a consistent way of ranking these according to their similarity to the target database, or in
 272 other words, their similarity to the application domain in question. This ranking of structured queries is
 273 accomplished through the use of the Bayesian network model shown in Fig. 4.

274 The network in Fig. 4 consists of a set of nodes, each representing a piece of information from the
 275 problem to be solved. To each node in the network is associated a binary random variable. This variable
 276 takes the value 1 to indicate that the corresponding information will be accounted for in the ranking
 277 computation. In this case, we say that the information was *observed*. Although the network can be easily
 278 expanded to model any database schema, for simplicity, here we show only two attributes, A_1 and A_2 . To
 279 simplify the notation, we define T_i as the set of all terms in the database that compose the values in the
 280 domain of attribute A_i . In this case, each value is considered as a string and the terms are the words in the
 281 string.

282 In the network of Fig. 4, the database is represented by node O . Each node A_i represents an attribute in
 283 the database. Each node a_{ij} represents a term in T_i . Each node Q_i represents a structured query to be ranked.
 284 Each node Q_{ij} represents the portion of the structured query Q_i that corresponds to the attribute A_j . Vectors
 285 \vec{a}_1 and \vec{a}_2 represent a possible state of the variables associated to the nodes a_{1i} and a_{2i} , respectively.

286 We can think of this network as modeling, for instance, a database on books, with attributes $A_1 = \text{Title}$
 287 and $A_2 = \text{Author}$. In this case, node A_{11} represents the title of a stored book, like “I, Robot”, where the term
 288 “I” is represented by node a_{11} and the term “Robot” is represented by node a_{12} . In a similar fashion, node
 289 Q_2 represents the structured query $Q_2 = \{\langle \text{Title}, \text{“I, Robot”} \rangle, \langle \text{Author}, \text{“Asimov”} \rangle\}$, where Q_{21} is the part
 290 referring to the Title attribute, Q_{22} the part referring to the Author attribute. Node a_{22} is the term “Asimov”.

291 The similarity of a structured query Q_i with the database O can be seen as the probability of observing
 292 Q_i , given that the database O was observed, $P(Q_i|O)$. Examining the network in Fig. 4, we can derive the
 293 equation:

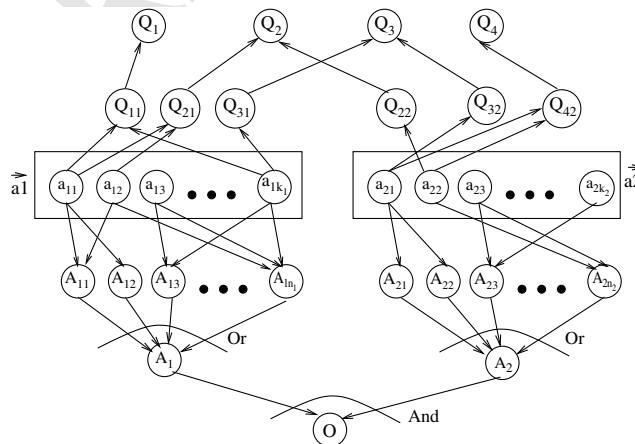


Fig. 4. Bayesian network model for query structuring.

$$P(Q_i|O) = \alpha \times \sum_{\vec{a}_1, \vec{a}_2} P(Q_i|\vec{a}_1, \vec{a}_2) \times P(O|\vec{a}_1, \vec{a}_2) \times P(\vec{a}_1, \vec{a}_2) \quad (2)$$

295 where α is a normalizing constant, as detailed in Pearl (1988).

296 To compute the probability of observing the structured query Q_i , given the state of all its attributes, we
297 consider that the query node Q_i should be active only when all the attribute nodes are active. However, we
298 want to penalize queries that do not contain all the terms specified by the user. This translates to the
299 following equation:

$$P(Q_i|Q_{i1}, Q_{i2}) = \begin{cases} \frac{C^{K_i}}{C^{K_U}} & \text{iff } Q_{i1} = 1 \wedge Q_{i2} = 1 \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

301 where K_i is the number of terms in the candidate structured query, K_U is the number of terms in the
302 unstructured query given by the user, and $C \geq 1$ is a constant, empirically chosen.

303 The probability of observing the database $P(O|A_1, A_2)$ is defined as a conjunction. We say that the
304 database is observed if all the attributes are observed, i.e.,

$$P(O|A_1, A_2) = \begin{cases} 1 & \text{iff } A_{i1} = 1 \wedge A_{i2} = 1 \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

306 Given these two definitions and the fact that \vec{a}_1 and \vec{a}_2 are independent, Eq. (2) can be rewritten as:

$$P(Q_i|O) = \alpha \times \sum_{\vec{a}_1, \vec{a}_2} P(Q_i|Q_{i1}, Q_{i2}) \times P(Q_{i1}|\vec{a}_1) \times P(Q_{i2}|\vec{a}_2) \times P(A_1|\vec{a}_1) \times P(A_2|\vec{a}_2) \times P(\vec{a}_1) \times P(\vec{a}_2) \quad (5)$$

308 Eq. (5) is the general equation for ranking a structured query. The remaining conditional probabilities can
309 now be defined according to the values stored in the database.

310 The probability of observing the part of query Q_i assigned to an attribute A_j , given that a set of terms,
311 indicated by \vec{a}_j , was observed is now defined as:

$$P(Q_{ij}|\vec{a}_j) = \begin{cases} 1 & \text{if } \forall k, g_k(\vec{a}_j) = 1 \text{ iff } t_{jk} \text{ occurs in } Q_{ij} \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

313 where $g_k(\vec{a}_j)$ gives the value of the k th variable of the vector \vec{a}_j and t_{jk} is the k th term in T_j . This equation
314 guarantees that just the states in which the only active terms are those of the query attribute Q_{ij} are taken
315 into consideration.

316 The probability of observing the attribute A_i , given the terms indicated by \vec{a}_i , is defined as a disjunction
317 of all probabilities for each possible value of A_i ,

$$P(A_i|\vec{a}_i) = 1 - \prod_{1 \leq j \leq n_i} 1 - P(A_{ij}|\vec{a}_i) \quad (7)$$

319 where n_i is the number of values in the database for attribute A_i . If we consider that $P(A_{ij}|\vec{a}_i)$ measures the
320 similarity between the value A_{ij} and the terms indicated by \vec{a}_i , then Eq. (7) tells us that, if the terms in \vec{a}_i fit
321 exactly one of the values A_{ij} , the final probability is 1. If not, the final probability will be higher the more
322 similar the terms in \vec{a}_i , are to the values in A_{ij} .

323 To define the probability $P(A_{ij}|\vec{a}_i)$ of observing a value A_{ij} , given a set of terms indicated by \vec{a}_i , we use the
324 cosine measure, as defined for the vector space model in IR systems (Salton & McGill, 1983). The value A_{ij}
325 for attribute A_i is seen as a vector of $|T_i|$ terms. To each term t_k in A_{ij} , we assign a weight w_{ik} that reflects the
326 importance of the term for attribute A_i , in the database, i.e.,

$$w_{ik} = \begin{cases} \frac{\log(1+f_{ik})}{\log(1+n_i)} & \text{if } t_k \text{ occurs in } A_{ij} \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

328 where f_{ki} is the number of occurrences of term t_k in the values of the attribute A_i , and n_i is the total number
 329 of values for attribute A_i , in the database. The values for f_{ki} and n_i are computed directly from the sample
 330 database. Notice that we do not consider the traditional *idf* normalization since this would penalize the
 331 most frequent terms, which are the most important for our approach.

332 The probability of observing A_{ij} is, therefore, defined as the cosine of the angle between vector A_{ij} and
 333 vector \vec{a}_i , i.e.,

$$P(A_{ij}|\vec{a}_i) = \cos(A_{ij}, \vec{a}_i) = \frac{\sum_{t_k \in T_i} w_{ik} g_k(\vec{a}_i)}{\sqrt{\sum_{t_k \in T_i} w_{ik}^2}} \quad (9)$$

335 Finally, since we have no a priori preference for any set of terms, we define the probability of vector \vec{a}_i as a
 336 constant $P(\vec{a}_i) = \frac{1}{2^{|T_i|}}$.

337 In sum, the probability $P(Q_i|O)$ is a disjunction of the similarities between the attribute values in the
 338 database and the values assigned to the respective attributes in the structured query. This is translated by
 339 the equation:

$$P(Q_i|O) = \eta \times \frac{C^{K_{Q_i}}}{C^{K_U}} \times \left[1 - \prod_{1 \leq j \leq n_1} 1 - \cos(A_{1j}, \vec{a}_1) \right] \times \left[1 - \prod_{1 \leq j \leq n_2} 1 - \cos(A_{2j}, \vec{a}_2) \right] \quad (10)$$

341 where \vec{a}_1 and \vec{a}_2 are the states where only the query terms referring to attributes A_1 and A_2 , respectively, are
 342 active, n_1 and n_2 are the total number of values for attributes A_1 and A_2 in the database, and η accounts for
 343 the constants α , $P(\vec{a}_1)$, and $P(\vec{a}_2)$.

344 We can now rank all the structured queries by computing $P(Q_i|O)$ for each of them. The user can then
 345 select one query for processing among the top ranked ones, or the system can simply process the first query.

346 It should be noted that, since we are only interested in the probability $P(Q_i|O)$, the actual network does
 347 not need to be implemented. Instead, Eq. (10) can be directly used. In the future, possible changes to the
 348 model should be considered to allow the use of inference algorithms. In this work, however, Bayesian
 349 networks were used as a modeling tool, useful in providing an intuitive model of the problem to be solved
 350 and guaranteeing the coherence and formal soundness, not only of the model, but also of future changes
 351 that may be applied.

352 4.3. Finding the best answers

353 Once a structured query is selected, either by the user or by the system itself, it is submitted to the target
 354 database. The set of objects returned can then be ranked according to the probability of satisfying the user
 355 needs. To this effect, the structured query and the returned objects are modeled using the Bayesian network
 356 shown in Fig. 5. This model is similar to those presented in Calado, Cristo, Moura, Nivio Ziviani, and
 357 Gonçalves (2003) and Ribeiro-Neto and Muntz (1996) for ranking documents according to different
 358 sources of evidence. In this work, however, the model is used to rank database objects according to the
 359 content of its attributes. Further, a new level is added to the network to represent attributes composed of
 360 several values.

361 As in the previous section, although the network can be easily expanded to model any database schema,
 362 for simplicity we show only two attributes, A_1 and A_2 . To simplify the notation, we define T_i as the set of all
 363 terms in the returned objects that compose the values in the domain of the attribute A_i . When ranking the
 364 returned objects special attention needs to be paid to value lists, i.e., attributes that are constituted by a
 365 list of values, as explained in Section 3.1. To exemplify how this type of attribute is treated, in the network
 366 we represent attribute A_2 as a value list attribute.

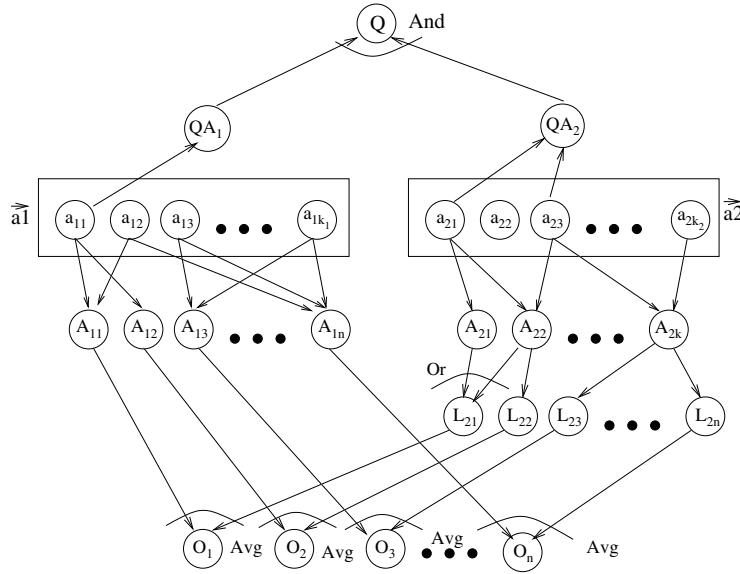


Fig. 5. Bayesian network model for result ranking.

367 In the network in Fig. 5, node Q represents the structured query, submitted for processing. Each node
 368 QA_i represents the part of the structured query Q relative to the attribute A_i . Each node a_{ij} represents a term
 369 in T_i . Each node A_{ij} represents the value of the attribute A_i in object O_j . Since attribute A_2 is a list attribute,
 370 we add one more level to the network. Each node L_{2j} represents the list of values assigned to the attribute A_2
 371 in object O_j . Finally, each node O_j represents a returned object. As before, with each node in the network is
 372 associated a random variable that takes the value 1 to indicate that information regarding the respective
 373 node was observed. Vectors \vec{a}_1 and \vec{a}_2 represent a possible state of the variables associated with nodes a_{1i}
 374 and a_{2i} , respectively.

375 An object O_j is ranked according to the probability of satisfying the structured query Q , i.e., the
 376 probability of observing O_j , given that Q was observed, $P(O_j|Q)$. Examining the network in Fig. 5, we have
 377 that:

$$P(O_j|Q) = \alpha \times \sum_{\vec{a}_1, \vec{a}_2} P(Q|\vec{a}_1, \vec{a}_2) \times P(O_j|\vec{a}_1, \vec{a}_2) \times P(\vec{a}_1, \vec{a}_2) \quad (11)$$

379 where α is a normalizing constant.

380 The probability $P(Q|QA_1, QA_2)$ is defined as a conjunction, i.e.,

$$P(Q|QA_1, QA_2) = \begin{cases} 1 & \text{iff } QA_1 = 1 \wedge QA_2 = 1 \\ 0 & \text{otherwise} \end{cases} \quad (12)$$

382 For $P(O_j|A_{1j}, L_{2j})$, we want the probability of observing the object O_j to increase as the number of its
 383 attributes that are observed increases. Thus, we define

$$P(O_j|A_{1j}, L_{2j}) = \text{no. of active parents} / \text{total no. of parents} \quad (13)$$

385 where the word *parents* refers to the parent nodes of node O_j .

386 These definitions and the fact that \vec{a}_1 and \vec{a}_2 are independent allow us to rewrite Eq. (11) as:

$$P(O_j|Q) = \alpha \times \sum_{\vec{a}_1, \vec{a}_2} P(QA_1|\vec{a}_1) \times P(QA_2|\vec{a}_2) \times \frac{P(A_{1j}|\vec{a}_1) + P(L_{2j}|\vec{a}_2)}{2} \times P(\vec{a}_1) \times P(\vec{a}_2) \quad (14)$$

388 Notice that the average between $P(A_{1j}|\vec{a}_1)$ and $P(L_{2j}|\vec{a}_2)$, originating from the definition of $P(O_j|A_{1j}, L_{2j})$,
 389 allows us to accumulate the evidences associated with each attribute for the final probability of the object.
 390 The remaining conditional probabilities can now be defined. We define a list as a disjunction of its
 391 values. Therefore, the probability for the list attribute is given by:

$$P(L_{2j}|\vec{a}_2) = 1 - \prod_{\forall k \in \mathcal{V}} (1 - P(A_{2k}|\vec{a}_2)) \quad (15)$$

393 where \mathcal{V} is the set of indices for the values in the list represented by L_{2j} . Eq. (15) determines that, for a value
 394 list to be observed, it is enough that one of its values is observed.

395 The probability of observing the part of Q relative to an attribute A_i , given that the set of terms indicated
 396 by \vec{a}_i was observed is defined as:

$$P(QA_i|\vec{a}_i) = \begin{cases} 1 & \text{if } \forall k, g_k(\vec{a}_i) = 1 \text{ iff } t_{ik} \text{ occurs in } QA_i \\ 0 & \text{otherwise} \end{cases} \quad (16)$$

398 where $g_k(\vec{a}_i)$ indicates the value of the k th variable of the vector \vec{a}_i , and t_{ik} is the k th term in T_i . This equation
 399 guarantees that only the states where the only active terms are those of the query Q will be taken into
 400 consideration.

401 As for query structuring, the probability of observing the value A_{ij} given that the terms indicated by \vec{a}_i ,
 402 were observed, $P(A_{ij}|\vec{a}_i)$, is defined using the cosine measure. The value A_{ij} of the object O_j is seen as a
 403 vector of $|T_i|$ terms. This time, however, we are only interested in determining whether each term t_k occurs,
 404 or not, in the value A_{ij} . Therefore, we define the term weight w_{ik} simply as:

$$w_{ik} = \begin{cases} 1 & \text{if } t_k \text{ occurs in } A_{ij} \\ 0 & \text{otherwise} \end{cases} \quad (17)$$

406 The probability of A_{ij} given \vec{a}_i is defined as:

$$P(A_{ij}|\vec{a}_i) = \cos(A_{ij}, \vec{a}_i) = \frac{\sum_{t_k \in T_i} w_{ik} g_k(\vec{a}_i)}{\sqrt{\sum_{t_k \in T_i} w_{ik}^2}} \quad (18)$$

408 As in Section 4.2, since we have no preference for any particular set of terms, we define the probability
 409 associated with the vector \vec{a}_i , as a constant, i.e.,

$$P(\vec{a}_i) = \frac{1}{2^{|T_i|}}$$

411 In conclusion, the probability $P(O_j|Q)$ is the average of the similarities between the attribute values in
 412 object O_j and the values for the respective attributes in the structured query, i.e.,

$$P(O_j|Q) = \frac{\eta}{2} \times \left[\cos(A_{1j}, \vec{a}_1) + \left[1 - \prod_{\forall k \in \mathcal{V}} (1 - \cos(A_{2k}, \vec{a}_2)) \right] \right] \quad (19)$$

414 where each \vec{a}_i is the state where the only active terms are those in the query part referring to the attribute A_i ,
 415 η summarizes the constants α and $P(\vec{a}_i)$, and \mathcal{V} is the set indexes for the values in the list for attribute A_2 .

416 Once we compute the values of $P(O_j|Q)$ for all the returned objects, they can be presented to the user as a
 417 ranked list. As for the network in Section 4.2, we note that this network does not need to be implemented,
 418 since Eq. (19) can be used directly.

419 **5. Experiments**

420 To validate our proposal, several experiments were performed. Before discussing results, we first describe
421 the experimental setup, including the test databases, test queries and evaluation criteria.

422 *5.1. Experimental setup*

423 *5.1.1. Test databases*

424 Experiments were run using seven distinct databases, with application domains Book, Hotel, Cruiser, CD,
425 Movie, Hardware, and Game (computer games). Table 1 shows the attributes composing each object in the
426 experimental target databases. All databases were built by extracting information from seven Web sites:
427 Barnes&Noble.com, HotelDiscount!com, Yahoo! Travel, Music Data Base, The Internet Movie Database,
428 Price Watch, and Mame DK. Data was extracted using the DEByE tool, described in Laender et al. (2002).

429 Table 2 shows, for each database, the total number of objects, the number of objects used to populate
430 the sample database, and the number of queries submitted during the experiment. Through preliminary
431 experiments, we were able to conclude that the best results were achieved by setting the value of p in the
432 sample database generation algorithm to 0.9 (see Section 3.3) and using about 10% of the objects in the
433 target database to build the sample database. We note that, by setting p to 0.9, we are using 90% of total
434 number of queries in the log, which corresponds to about 50% of the distinct queries.

Table 1
Structure of the objects in each database

Database	Attributes	Type
Book	Title, Author, CoverType, Date, Price	Atomic
Hotel	Name, Address AreaActivities, PropertyAmenities	Atomic List
Cruiser	Title, Price, CruiseLine, ShipName PortsOfCall	Atomic List
CD	Title, Artist, CatNumber, Label, PlayingTime, Style, Language, Country	Atomic
Movie	Title, Writer, Director, Genre, Year Cast	Atomic List
Hardware	Brand, Product, Description, Price, ShipDate, Dealer Phone	Atomic List
Game	Name, Manufacturer, Year, GameType, Status	Atomic

Table 2
Statistics of all databases

Database	# Objects	# Samples	# Queries
Book	38,086	380	40
Hotel	2362	230	39
Cruiser	890	90	40
CD	14,210	1420	28
Movie	100,000	5000	39
Hardware	6966	690	40
Game	3360	330	40

435 This is a fundamental difference from the work presented in Calado et al. (2002), where the whole target
436 database was assumed available to the query structuring system.

437 Since query logs were not available, artificial logs were created. The log creation process consisted of
438 taking a set of random objects from the target database, selecting a random number of attribute-value pairs
439 from each object, and building a structured query with the selected attribute-value pairs. Although arti-
440 ficially constructed, this set of queries tries to follow a realistic setting. Thus, the final distribution of queries
441 in the generated log follows Zipf's law, a common distribution in search engine queries. Further, the
442 average number of fields per query is 1.96 and the average number of terms per query is 3, since it is known
443 that Web users tend to use short and imprecise queries (Spink, Wolfram, Jansen, & Saracevic, 2001). The
444 logs contain a total of 2509 distinct queries, of which 1319 were used to build the sample database and 266
445 were used for testing the system.

446 5.1.2. Test queries

447 For each database D , a random set of structured queries was selected from the corresponding log, to be
448 used as test queries. To each test query Q_i , for a database D with schema S_D , we assigned an unstructured
449 query U_i , defined as $U_i = \{t \mid t \text{ is a term in } v_j \text{ and } \langle A_j, v_j \rangle \in Q_i\}$, where $A_j \in S_D$. The query U_i is, thus, the set
450 of all terms that compose the attribute values of Q_i . For instance, to the structured query
451 $Q = \{\langle \text{Title, "I, Robot"} \rangle, \langle \text{Author, "Isaac Asimov"} \rangle\}$, corresponds the unstructured query $U = \{\text{"I",}$
452 $\text{"Robot", "Isaac", "Asimov"}\}$.

453 In our experiments, the query Q_i was taken as the correct structured formulation of the unstructured
454 query U_i . To evaluate the system, we submit query U_i to the structuring mechanism, and examine how
455 close the returned structured queries were to query Q_i . Ideally, the system should be able to place a candi-
456 date query equal to Q_i at the top of the ranking. Each query was submitted to only one database.

457 From the selected test queries, about 20% were not used in the construction of the sample database. For
458 this reason, some of the unstructured queries contained terms that were not known to the system. These
459 terms were discarded and the resulting incomplete queries received no special treatment, being included in
460 the evaluation as all other generated queries. This reflects a realist Web environment, where users may enter
461 keywords that are not present in the sample database. We note that, in a real Web environment, the sample
462 database could be further augmented by adding these new terms, as they occur in the users' queries. This
463 approach, however, was not explored in this work.

464 5.1.3. Evaluation criteria

465 To evaluate the structuring algorithm and ranking model performance, three criteria were used: (1)
466 *Application Domain Correctness*, i.e., if the algorithm was able to generate the query for the correct
467 application domain and rank it at top; (2) *Query Correctness*, i.e., if the correct query is among the top
468 queries returned for a given application domain; and (3) *Quality of Returned Objects*, i.e., if the returned
469 objects are relevant regarding the user query.

470 Given an unstructured query U_i , the Quality of Returned Results criteria is measured by comparing the
471 set of objects returned by the system when U_i is processed to a set of objects known to be the correct
472 answers for U_i . Usually, this is accomplished by letting a user evaluate each result individually, indicating
473 those that he considers relevant. In this work, however, we assume that the correct results for query U_i are
474 those returned by the target database when the corresponding correctly structured query Q_i is submitted.
475 Thus, to determine the set of correct answers to U_i , the corresponding structured query Q_i is submitted to
476 the target database and the set of objects returned RU_i is taken. This set is then used to evaluate how many
477 relevant objects the system is able to retrieve when the unstructured query U_i is processed.

478 The most commonly used measures to determine the quality of a ranked set of objects are precision and
479 recall figures (Baeza-Yates & Ribeiro-Neto, 1999). These measures are defined as follows. For a given query
480 U_i , let RU_i be the set of correct objects and let SU_i be the total set of objects returned by the system.

481 Precision is defined as $p = |SU_i \cap RU_i|/|SU_i|$, which is the fraction of the set of objects retrieved that is
 482 correct. Recall is defined as $r = |SU_i \cap RU_i|/|RU_i|$, which is the fraction of correct objects that was retrieved.

483 Usually, we are interested in the values of precision taken at 11 standard recall points, 0%, 10%, 20%,
 484 30%, 40%, 50%, 60%, 70%, 80%, 90%, and 100%. This precision–recall curve gives us the quality of the
 485 ranked set of answers, as we descend along the ranking. Precision at 0% is defined as the precision when we
 486 the first relevant object in the ranked answer set is found. In our experiments, the precision and recall values
 487 can be seen as correlation measures between the results of the manually structured queries and the queries
 488 structured by our algorithm.

489 5.2. Results

490 Table 3 shows the results for application domain correctness when we consider the three first queries in
 491 the ranking of the automatically structured queries. We observe that the system was able to rank the query
 492 for the correct domain at the top 88% of the time. About 93% of the time, the correct domain is placed
 493 among the three highest ranked queries.

494 Evaluation of query correctness was performed using two measures: percentage of correct queries (CQ)
 495 and percentage of correct attributes (CA). Table 4 shows the values for each database. As expected, the best
 496 results were achieved in databases where the attribute values are more regular, like Game, or Cruiser, with
 497 about 90% and 80% of correct queries ranked first, respectively. For databases in which attribute values are
 498 composed of free text with a less controlled pattern, correctness values were lower. For instance, in the
 499 Hardware domain, whose objects contain a *description* attribute, or in the Movie domain, in which some
 500 attributes, like writer and director, have very similar values, only 50% and 59% of the first ranked queries,
 501 respectively, were correctly structured.

502 Results of domain correctness and query correctness allow us to draw two conclusions. First, even when
 503 the system is not able to accurately rank the correct structured query, in most cases, this query will be
 504 among the three highest ranked. Thus, in most cases, the user would need to look only at three queries
 505 before choosing the appropriate one. Second, it is interesting to note that, even when the correct query was
 506 not found, many of the attributes for the highest ranked queries were correct. Thus, relevant items can still
 507 be obtained by submitting the query to the target database, as the following results indicate.

508 For the quality of returned objects criterion, results are shown in the precision–recall graph of Fig. 6.
 509 Curves labeled “first”, “second”, and “third” correspond to the precision–recall in the set of objects ob-
 510 tained by submitting the top three ranked structured queries. The curve labeled “user” represents the results
 511 of submitting the structured query chosen by a user from the top three ranked queries.

512 At low recall values, by submitting the highest ranked query, the system was able to obtain results with a
 513 precision consistently above 70%, i.e., less than 30% of the top ranked returned objects were incorrect. The
 514 second and third ranked queries, however, achieved a much worse performance, being unable to return
 515 more than about 40% and 25% correct results at low recall, respectively. With a small user intervention,
 516 however, results are much improved, getting precision values close to 90%, when all relevant objects are
 517 retrieved.

Table 3
Application domain correctness for all queries

Ordering position	Domain correctness rate (%)
First	88.3
First and second	91.7
First, second and third	92.9

Table 4
Percentage of correct queries (CQ) and correct attributes (CA) for each database

Database		First	First and second	First, second, and third
Hotel	CQ (%)	76.9	82.1	84.6
	CA (%)	87.1	90.1	91.1
Cruiser	CQ (%)	80.0	92.5	92.5
	CA (%)	89.4	95.4	95.4
Book	CQ (%)	77.5	80.0	80.0
	CA (%)	80.0	81.6	81.6
Hardware	CQ (%)	50.0	50.0	57.5
	CA (%)	63.6	68.2	74.4
Game	CQ (%)	90.0	92.5	95.0
	CA (%)	90.1	91.1	92.1
Movie	CQ (%)	59.0	71.8	87.2
	CA (%)	76.2	85.4	91.5
CD	CQ (%)	67.9	92.9	92.9
	CA (%)	80.6	95.2	95.2
Total	CQ (%)	71.8	79.7	83.8
	CA (%)	80.9	86.2	88.5

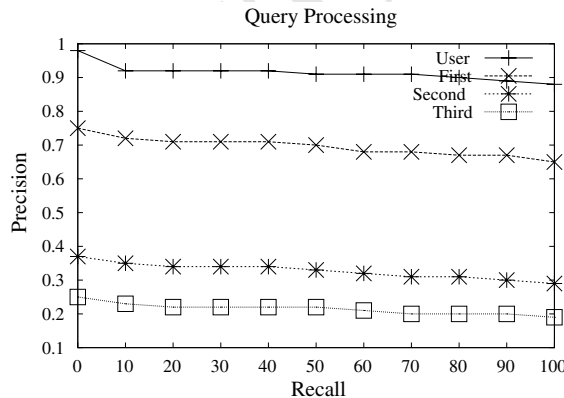


Fig. 6. Precision–recall values for the results of the automatically structured queries, averaged over all databases.

518 **6. Conclusions and future work**

519 In this work, we have presented a novel approach for querying Web databases using unstructured
 520 queries, i.e., queries constituted only of keywords. The approach is based on a Bayesian network model,
 521 which combines evidence taken from object attributes to provide structure for the keyword-based user
 522 query. Once the structured query is determined, it is submitted to one or more databases. Since the returned
 523 list of results can be quite extensive, we also present a Bayesian network model to rank the results according
 524 to their probability of satisfying the user’s needs.

Our query structuring and result ranking models provide a framework for querying Web databases that: (1) allows the formulation of queries using a very simple interface—one single text search box, (2) allows the use of a single interface to query any number of different databases, and (3) provides the results as a ranked set, in which the objects most likely to answer the user's query are shown first. These qualities make our proposal ideal for querying Web databases such as the ones available from on-line stores or digital libraries.

To test our approach, we implemented a prototype Web search system, capable of querying several different databases with a single interface. Experimental results using seven different databases show that, for each unstructured query submitted, our system was able to show the user the correct query among the first three proposed, from 58% to 95% of the time. Further, when the user selects one of these three top queries for processing, the ranked answers present average precision figures of about 90%.

As future work, we are currently considering the following problems. First, although results strongly suggest that our proposal is indeed effective, further tests should be performed with human users. Some preliminary experiments were already performed, showing that query structuring is indeed useful to real users and can increase their degree of satisfaction with the system.

Second, Web databases are usually flat, in the sense that they do not have a deep hierarchy of attributes for its objects. However, other types of databases may have more complex structures. The proposed Bayesian network model should, therefore, be extended to allow different types of attribute organization, thus making it more general and able to process different types of databases.

Third, additions to the query language should be provided in order to increase its expressiveness. This can be accomplished, for instance, by the introduction of numeric operators, or by allowing the user to restrict certain terms to certain attributes. Also, information in the sample database can be used to provide query expansion, thus allowing the improvement of incomplete or vague user queries. This extension will also imply changes in the proposed Bayesian network models.

Finally, adaptations of the models should be considered to facilitate the use of belief propagation algorithms. This would allow a more flexible use of the network, to be explored in the future. For instance, the model could include ways of automatically determining the size of the sample database to be used, or the best set of queries to be used in its generation.

Acknowledgements

This work was supported in part by the I3DL project—grant 680154/01-9, the GERINDO project—grant MCT/CNPq/CT-INFO 552.087/02-5, and by individual grants MCT/FCT SFRH/BD/4662/2001 (Pável Calado) and CNPq 304890/02-5 (Alberto H.F. Laender).

References

- Acid, S., de Campos, L. M., Fernández-Luna, J. M., & Huete, J. F. (2003). An information retrieval model based on simple Bayesian networks. *International Journal of Intelligent Systems*, 18(2), 251–265.
- Agrawal, S., Chaudhuri, S., & Das, G. (2002). DBXplorer: A system for keyword-based search over relational databases. In *Proceedings of the 18th international conference on data engineering* (pp. 5–16). San Jose, CA, USA: IEEE Computer Society.
- Baeza-Yates, R., & Ribeiro-Neto, B. (1999). *Modern information retrieval*. New York, USA: Addison Wesley.
- Bruno, N., Gravano, L., & Marian, A. (2002). Evaluating top-k queries over Web-accessible databases. In *Proceedings of the 18th international conference on data engineering* (pp. 369–382). San Jose, CA, USA: IEEE Computer Society.
- Calado, P., Cristo, M., Moura, E., Nivio Ziviani, B. R.-N., & Gonçalves, M. A. (2003). Combining link-based and content-based methods for Web document classification. In *Proceedings of the 12th international conference on information and knowledge management* (pp. 394–401). New Orleans, LA, USA: ACM Press.

- 567 Calado, P., da Silva, A. S., Vieira, R. C., Laender, A. H. F., & Ribeiro-Neto, B. A. (2002). Searching Web databases by structuring
568 keyword-based queries. In *Proceedings of the 11th international conference on information and knowledge management* (pp. 26–33).
569 McLean, VA, USA: ACM Press.
- 570 Callan, J. P. (1996). Document filtering with inference networks. In *Proceedings of the 19th annual international ACM SIGIR conference*
571 *on research and development in information retrieval* (pp. 262–269). Zurich, Switzerland: ACM Press.
- 572 Chaudhuri, S., & Gravano, L. (1999). Evaluating top-k selection queries. In *Proceedings of 25th international conference on very large*
573 *data bases VLDB'99* (pp. 397–410). Edinburgh, UK: Morgan Kaufman.
- 574 Cohen, W. W. (1999). Reasoning about textual similarity in a Web-based information access system. *Autonomous Agents and Multi-*
575 *Agent Systems*, 2(1), 65–86.
- 576 Dar, S., Entin, G., Geva, S., & Palmon, E. (1998). DTL's DataSpot: Database exploration using plain language. In *Proceedings of 24th*
577 *international conference on very large data bases VLDB'98* (pp. 645–649). New York, USA: Morgan Kaufman.
- 578 de Campos, L. M., Fernández-Luna, J. M., & Huete, J. F. (1998). Query expansion in information retrieval systems using a Bayesian
579 network-based thesaurus. In *Proceedings of the 14th annual conference on uncertainty in artificial intelligence UAI'98* (pp. 53–60).
580 San Francisco, USA: Morgan Kaufmann.
- 581 Doorenbos, R. B., Etzioni, O., & Weld, D. S. (1997). A scalable comparison-shopping agent for the World-Wide Web. In *Proceedings*
582 *of the first international conference on autonomous agents* (pp. 39–48). CA, USA: Marina del Rey.
- 583 Dumais, S., Platt, J., Heckerman, D., & Sahami, M. (1998). Inductive learning algorithms and representations for text categorization.
584 In *Proceedings of the 7th international conference on information and knowledge management* (pp. 148–155). Bethesda, USA: ACM
585 Press.
- 586 Florescu, D., Kossmann, D., & Manolescu, I. (2000). Integrating Keyword Search into XML Query Processing. *WWW9/Computer*
587 *Networks*, 33(1–6), 119–135.
- 588 Goldman, R., Shivakumar, N., Venkatasubramanian, S., & Garcia-Molina, H. (1998). Proximity search in databases. In *Proceedings*
589 *of 24th international conference on very large data bases VLDB'98* (pp. 26–37). New York, USA: Morgan Kaufman.
- 590 Haines, D., & Croft, W. (1993). Relevance feedback and inference networks. In *Proceedings of the 16th annual international ACM*
591 *SIGIR conference on research and development in information retrieval* (pp. 2–11). Pittsburgh, USA: ACM Press.
- 592 Laender, A. H. P., Ribeiro-Neto, B., & da Silva, A. S. (2002). DEByE—data extraction by example. *Data and Knowledge Engineering*,
593 40(2), 121–154.
- 594 Liu, L., Pu, C., & Han, W. (2000). XWRAP: An XML-enabled wrapper construction system for Web information sources. In:
595 *Proceedings of the 16th international conference on data engineering*, San Diego, CA, USA, pp. 611–621.
- 596 Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: networks of plausible inference* (2nd ed.). San Mateo, USA: Morgan
597 Kaufman.
- 598 Ribeiro-Neto, B., & Muntz, R. (1996). A belief network model for IR. In *Proceedings of the 19th annual international ACM SIGIR*
599 *conference on research and development in information retrieval* (pp. 253–260). Zurich, Switzerland: ACM Press.
- 600 Salton, G., & McGill, M. J. (1983). *Introduction to modern information retrieval*. New York, USA: McGraw-Hill.
- 601 Shin, D., Nam, S., & Kim, M. (1997). Hypertext construction using statistical and semantic similarity. In *Proceedings of the 2nd ACM*
602 *international conference on digital libraries* (pp. 57–63). Philadelphia, USA: ACM Press.
- 603 Spink, A., Wolfram, D., Jansen, B. J., & Saracevic, T. (2001). Searching the Web: The public and their queries. *Journal of the American*
604 *Society for Information Science and Technology*, 52(3), 226–234.
- 605 Swartout, W., & Tate, A. (1999). Ontologies. *IEEE Intelligent Systems*, 14(1), 18–19.
- 606 Turtle, H. R., & Croft, W. B. (1990). Inference networks for document retrieval. In *Proceedings of the 13th annual international ACM*
607 *SIGIR conference on research and development in information retrieval* (pp. 1–24). Brussels, Belgium: ACM Press.