Associative Query Answering via Query Feature Similarity*

Wesley W. Chu and Guogen Zhang
Computer Science Department
University of California, Los Angeles, CA 90095, USA

Abstract

Associative query answering provides additional relevant information to the queries that is not explicitly asked, but is of interest to the user. For a given query, associative information may be derived from past user query cases based on the user type and the query context. A case-based reasoning approach that matches query features is proposed. Query feature consists of the query topic, the output attribute list, and the selection constraints. The similarity of the query feature is defined and can be evaluated from the semantic model that is derived from the database schema. Query feature based associative attribute search is presented.

1 Introduction

Associative query answering [1, 2, 3] is to provide users with additional relevant information to the query that the user did not ask or does not know how to ask, and may be of interest to the users. For a given query, the associative information depends on the user's intention, interests, and query context. For example, for the query: "*List airports in Tunisia that can land a C-5 cargo plane*," the relevant information for a transportation planner may include the *railway facility* at the airports. But for a pilot, the interested information may be the *runway condition* and *weather condition*.

Attribute based association from case base has been used [3]. However, the derived associative information often is not domain and user specific. To remedy such shortcoming, we propose to use query feature that includes query topic as additional information to provide relevant information. As a result, the proposed approach yields more relevant information than that of the attribute based search approach [3], which is only based on the query output attribute list and selection conditions. Further, the domain knowledge from a semantic model is used to evaluate query feature similarity to improve the efficiency in searching associative information from case bases.

The paper is organized as follows. In section 2, we discuss our approach and present an overview of the associative query answering system. In section 3 we describe a semantic model based on the database schema used for topic identification and query similarity evaluation. Then in section 4 query feature and query similarity are discussed. The search and learning process is described in section 5.

*This research is supported in part by DARPA contracts No. F30602-94-C-0207 and No. N66001-97-C-8601.

2 Associative query answering systems

2.1 Approach

Due to the unlimited number of user queries possible for a relational database, it will be very costly to derive associative information by knowledge engineers [4]. Therefore, a case-based reasoning approach [3] is used to capture the domain and user-specific information for associative query answering.

Given a set of past queries, searching for associative attributes can be based on the frequencies of attribute groups that appeared in the queries. However, such approach does not provide context of co-occurrent attributes. Therefore we propose to use the information in the database schema and the query to analyze the user intention to guide query association.

For an SQL query, three types of associative attributes and expressions are identified: 1) *simple associative attributes*: attributes of relations in the current query; 2) *extended associative attributes*: attributes of relations introduced into the query by joins with the existing relations; and 3) *statistical associative information*: aggregate functions, e.g. `count` and `sum`, that are related to the main entity in the query. In this paper, we only focus on the first two types.

2.2 System overview

The associative query answering process is illustrated in Fig. 1. A semantic model is constructed from the database schema and domain knowledge. Each user can derive his own domain-specific semantic model from the general semantic model via user-specific knowledge. A case base stores past queries and the corresponding association cases. After an SQL query converts to a query feature vector, the case base is searched based on the similarity of the input query to queries in the case base. The matched associative attribute candidates are displayed to user for selection.

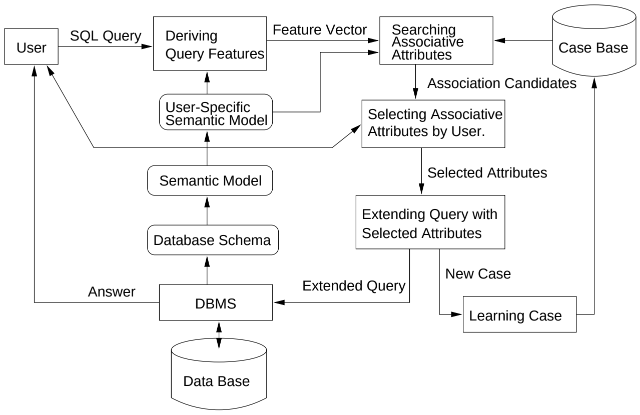

Figure 1: The associative query answering process by query feature matching.

The query is extended with the selected attributes and is sent to the DBMS for execution. Each query, with its association and user feedback, is recorded in the case base for future reasoning use.

For a given query, the system first locates similar query cases with the same topic. If such query cases exist, the associative attributes are used as the association. Otherwise, query cases with related topics are searched and the query feature similarity is used to guide the search process.

3 Database schema and semantic model

In relational databases, relations are constructs used for representing entities and relationships [5]. To explicitly express the semantics in the relations, we construct a semantic (EER) model based on the database schema [6], user-defined relationships, and mapping from its constructs into the relations in the database.

The semantic model is represented as a directed graph called semantic graph. The nodes in the graph are entities and complex relationships that have corresponding relations in the database. The edges are links (i.e. joins) among relations. The user can define additional nodes and edges for his views in his own model. The semantic graph is further divided into a set of overlapping query contexts, each of which is a connected component of the user's semantic graph, and reflect the user's interests on the database when performing a certain task. A query topic is a connected component of a query context. A query is a constraint on a topic, i.e. a selection on instances of a topic, with an output attribute list. Fig. 2 shows a context of the semantic model about aircrafts and airports for a transportation database, where "landing" is a user-defined link.

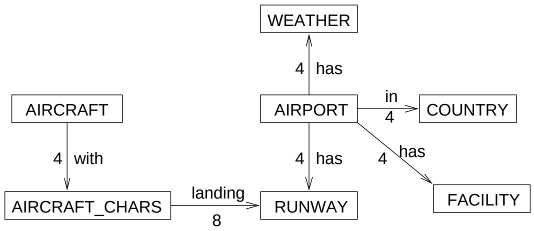

Figure 2: A query context about AIRCRAFT and AIRPORT for a transportation database.

To compare the similarity of query topics, we assign weights for the nodes and edges in each query context. Weights are used to measure the relative importance of the semantic contents of the links in the graph. An equal unity weight is assigned to all nodes. Weights for the edges are based on their semantic contents, or the context specificity of the links. For example, in Fig. 2 "landing" is a more specific link compared to other links, thus a larger weight is assigned. The weight for a path is the sum of weights of the nodes and edges on the path. For any two nodes, if one path connecting them has larger weight than another, the larger one is considered containing more specific semantics than the smaller one [7, 8]. Let queries Q_1 and Q_2 contain a common path with weight w_a, Q_1 and Q_3 also have a common path with weight w_b. If $w_a > w_b$, then the topic of Q_1 is more similar to Q_2 than Q_3. The same links may have different relative importance under different contexts, and the weight assignment is user and context sensitive.

4 Query feature and similarity

4.1 Query feature vector

A query can be characterized with the following components:

1. a topic graph, T, defining the topic of the query;
2. an output attribute list, O, defining the queried aspect on the topic;
3. a set of selection conditions, C.

We shall denote $Q = \langle T, O, C \rangle$ as the *feature vector* of a query.

For an SQL query with a single SELECT statement, a feature vector can be constructed. Fig. 3 shows the SQL query and its feature vector for the example stated in the introduction.

4.2 Topic relationships among queries

For two queries $Q_1 = \langle T_1, O_1, C_1 \rangle$ and $Q_2 = \langle T_2, O_2, C_2 \rangle$, the relationships of their topics can be used to derive the associative information.

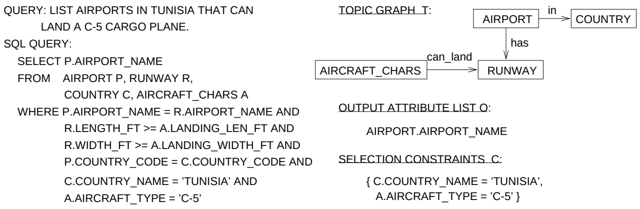

Figure 3: Feature vector for the example query.

1. *Same topic*: $T_1 = T_2$. If $O_2 - O_1 \neq \emptyset$, then $O_2 - O_1$ may be used as associative attributes for Q_1.
2. *Sub-topic*: $T_1 \subset T_2$. T_1 is a subtopic of T_2, and T_2 is a super topic of T_1. T_1 can be extended to T_2 and $O_2 - O_1$ can be used as possible associative attributes for Q_1.
3. *Overlapped topics*: $T_1 \not\subset T_2$ and $T_1 \not\supset T_2$, but $T_1 \cap T_2 \neq \emptyset$. Similar to the sub-topic relationship, $T_2 - T_1$ can be used to extend T_1, and $O_2 - O_1$ are possible associative attributes for Q_1.
4. *Unrelated topics*: $T_1 \cap T_2 = \emptyset$. The topics of Q_1 and Q_2 are not directly related. No associative information can be derived from an unrelated query.

4.3 Query similarity

Query similarity is based on three components of the feature vector: query topic, output attribute list, and constraints. Their relative importance is given by their corresponding weights: w_t, w_o, and w_c, with $w_t + w_o + w_c = 1$. The weights are user and context dependent.

For an input query $Q_1 = \langle T_1, O_1, C_1 \rangle$ and a query in the case base $Q_2 = \langle T_2, O_2, C_2 \rangle$, the similarity of Q_1 and Q_2 is defined as:

$$\begin{aligned}\theta(Q_1, Q_2) = \quad & w_t \times \theta_t(Q_1, Q_2) + \\ & w_o \times \theta_o(Q_1, Q_2) + \\ & w_c \times \theta_c(Q_1, Q_2)\end{aligned} \tag{1}$$

where $\theta_t()$, $\theta_o()$, and $\theta_c()$ are the similarity functions for the topics, output attribute lists, and selection constraints, respectively.

The *topic similarity* is based on the size of the common topic components relative to the size of both topics together as follows:

$$\theta_t(Q_1, Q_2) = \frac{|T_1 \cap T_2|}{|T_1 \cup T_2|} \tag{2}$$

where $|T|$ is the size of the topic, defined as the sum of weights of all nodes and edges. For instance, if the two topics are the same, then their topic similarity is 1. If a topic is a sub-topic of another, their similarity is the ratio of the size of the sub-topic to that of the super topic. Non-related topics have similarity of 0.

The *similarity of output attribute lists* is defined as:

$$\theta_o(Q_1, Q_2) = \frac{|O_1 \cap O_2|}{|O_1|} \tag{3}$$

which is based on the number of attributes in O_1 contained in O_2, and reflects the proximity of the queried aspect of Q_2 to that of Q_1.

The *similarity of selection constraints* is based on two factors: the number of the identical condition patterns and the similarity of values for these condition patterns. To evaluate the similarity, we divide C into a mapping S and a tuple of constants V. When the mapping is supplied with the constant values as the arguments, the set of selection conditions is generated. For example, if we have two selection conditions:

C.COUNTRY_NAME= 'TUNISIA',
A.AIRCRAFT_TYPE = 'C-5'

then we have $S = \{C.COUNTRY\_NAME = \$1, A.AIRCRAFT\_TYPE = \$2\}$, and $V = ('TUNISIA', 'C-5')$, where \$1 and \$2 refer to corresponding components of the tuple. $S(V)$ is the above selection conditions. We have

$$\theta_c(Q_1, Q_2) = \theta_s(Q_1, Q_2) \times \theta_v(Q_1, Q_2) \tag{4}$$

where $\theta_s(Q_1, Q_2)$ is the similarity of mappings, defined as:

$$\theta_s(Q_1, Q_2) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|} \tag{5}$$

and $\theta_v(Q_1, Q_2)$ is the similarity of constraints values for the common conditions. Assume there are k common conditions and the projections of tuples V_1 and V_2 onto these common conditions are represented as sets $D_{i1}, \ldots, D_{ik}$ for $i = 1, 2$, $\theta_v(Q_1, Q_2)$ is defined as:

$$\theta_v(Q_1, Q_2) = \frac{1}{k} \times \sum_{j=1}^{k} (1 - RE(D_{1j}, D_{2j})). \tag{6}$$

where $RE()$ is the relaxation error [9] for two sets of values, which is the average pair-wise distance of elements from each set. The $RE()$ values are provided from a tree-structured classification hierarchy TAH [10, 11] and is normalized into $[0, 1]$. As exceptions to equation 4, if $|S_1 \cup S_2| = 0$ (i.e. no selection conditions in both queries), then $\theta_c(Q_1, Q_2) = 1$. And if one of them is empty, say S_1, then the S_1 is treated as equal to S_2, with constants being the projections of the attributes from the database.

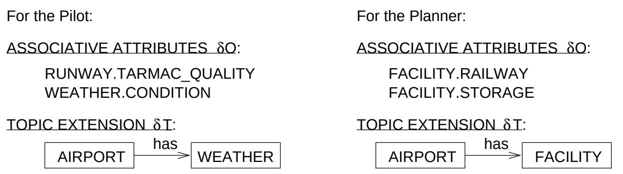

Figure 4: Possible associations for the example query.

5 Searching associative attributes from case base

Given an input query $Q = \langle T, O, C \rangle$, an extended query with associative information is $Q' = \langle T \cup \delta T, O \cup \delta O, C \rangle$, where δO is a set of the associative attribute candidates and δT is the necessary topic extension to derive these associative attributes. The associative extension denoted as $\langle \delta O, \delta T \rangle$ should not change the user's constraints on the query topic. Our goal is to determine $\langle \delta O, \delta T \rangle$ for the query Q. Fig. 4 shows the associations for two user types for the query example, and the corresponding query context and feature vector are shown in Fig. 2 and 3.

5.1 Search process

Searching of case base is done for each user type and context. Each query case in the case base is represented by its feature vector, and a log of past associative extensions of form $\langle \delta O, \delta T \rangle$ with positive user feedback $\langle \delta O_p, \delta T_p \rangle$ when each time the case is applied, where $\delta O_p \subseteq \delta O$ and $\delta T_p \subseteq \delta T$ and they are user selected attributes and corresponding topic extension. Thus, $\delta O - \delta O_p$ and $\delta T - \delta T_p$ are the negative feedback. Each case is time-stamped for ordering purposes.

The case base is organized based on the query topics at the first level, the output attribute lists at the second level, and then selection constraints at the third level. The cases are classified into groups based on these levels. The relationships among topics are linked to facilitate the search.

Given an input query $Q = \langle T, O, C \rangle$, the search process is outlined as follows:

1. If there is a set of cases with the same topic as T, use the *same topic matching*, in which search for associative attributes is limited within this case group. If the same topic matching is successful, the search is completed;
2. Otherwise, use *related topic matching*, where search is first expanded into the sub-topics and super topics, and then to the overlapped topics if necessary.

The success of the search process is based on the number of relevant associative attributes required, which can be determined either by system default heuristics or by the user.

The associative candidates are displayed to the user for selection. The selected associative attributes (positive feedbacks) are extended to form a new query. The associative extension is recorded into the case base. The details of the search will be discussed in the following sections.

5.2 Same topic matching

The set of query cases with the same topic as the input query are searched. The single most similar query case is used at first and its log is retrieved for evaluation. The search process terminates if the associative attributes provide sufficient relevant information. Otherwise, less similar query cases are considered for association, and the process is repeated until sufficient relevant associative information is obtained or all the cases with the same topic are exhausted.

The usefulness evaluation for an attribute is based on the similarity of the query and usefulness weight of the attribute, which is calculated from the feedback history. Two different schemes based on history importance are proposed for usefulness weight calculation. One is to give all the cases in the history the same importance, which we call *equal importance* (EI) scheme. The other, called *recently used* (RU) scheme places more importance on recent cases. The former is suitable for a stable database and users with stable behavior, and the latter more suitable for a database incurring many updates or users with transient behaviors.

For the EI scheme, the weight assigned to an associative attribute can be calculated as follows:

$$W_{EI} = \frac{1 + N_p}{d + N_p + N_n} \tag{7}$$

where N_p and N_n are the positive and negative counters respectively, and $d \geq 1$ is used to control the effect of the first few feedbacks.

For the RU scheme, all the feedbacks are ordered based on the time sequence, and N_p, N_n, R_p, and R_n are the total positive and total negative feedbacks, and recent positive and negative feedbacks, respectively. The recent feedbacks are counted based on the most recent turning points in the feedback sequence. The weight is calculated as:

$$W_{RU} = (1 - \alpha) \times \frac{N_p}{N_p + N_n} + \alpha \times \frac{R_p}{R_p + R_n} \tag{8}$$

where $0.5 < \alpha \leq 1$ is used to control the relative significance of the recent feedbacks.

The relevancy of an associative attribute from the same query cases is the usefulness weight of the attribute from equation 7 or 8, times the similarity of the queries from equation 1. The total usefulness is the weighted average from each group by the number of cases. If the total usefulness is greater than a certain threshold (e.g. 0.3), then it is a useful associative attribute. The threshold value depends on context, user profile, and should be estimated experimentally.

5.3 Related topic matching

If the association search on the same topic query cases does not provide sufficient information, then search is expanded to the sub-topic and super topic groups, and to the overlapped topics.

For the set of queries $\{Q_i = \langle T_i, O_i, C_i \rangle | i = 1, \ldots, n\}$, where n is the number of cases under evaluation, each attribute in O_i is assigned a weight in accordance with query similarity (equation 1). All the attributes from the searched queries are merged. The weight for an attribute $o \in \cup_i O_i$, $W(o)$, is:

$$W(o) = \sum_{i=1}^{n} m(o, O_i) * \theta(Q, Q_i), \tag{9}$$

where the membership function $m(o, O_i)$ is 1 if $o \in O_i$, and 0 otherwise. The list is sorted based on the attribute weights.

For query cases with super topics, both simple associative attributes and extended associative attributes can appear in the result. For each extended attribute, the corresponding topic is used for deriving δT, which is the path in the super topic from the node containing the attribute to a node in the input query topic graph.

6 Conclusions

A query feature based associative query answering from case base is proposed in the paper. A query feature consists of its topic, its output attribute list, and the selection constraints. For a given query, the topic similarity of cases is evaluated from a user-specific semantic model based on the database schema, user type, and context. A case base that records the past user queries and association cases is used for searching for associative attributes, which is based on the user type, context, and the query features. Cases with a same topic are searched first, then followed by searching the related cases with sub-topics, super topics, or overlapped topics if the same topic searching does not provide any relevant information. We are currently in the process of implementing the proposed approach on the CoBase system [9]. We plan to evaluate the improvement of using the proposed query feature approach as compared with the attribute linking approach which searches past cases that have a same attribute either from the output attribute list or the selection constraints [3].

Acknowledgments

The authors would like to thank Chih-Cheng Hsu of UCLA for his helpful comments during the writing of this paper.

References

[1] F. Cuppens and R. Demolombe, Extending answer to neighbor entities in a cooperative answering context. *Decision Support System*, pp. 1-11, 1991.

[2] W. W. Chu and Q. Chen, Neighborhood and Associative Query Answering. *Intelligent Information Systems*, 1(3-4):355-382, 1992.

[3] G. Fouqué, W. W. Chu, and H. Yau, A Case-Based Reasoning Approach for Associative Query Answering, in *Proc. of the 8th SMIS*, Oct., 1994, NC.

[4] J. F. Allen and C. R. Perrault, Analyzing Intention in Utterances, *Artificial Intelligence*, 15: 143-178, 1980.

[5] J. D. Ullman, *Principles of database and knowledge-base systems*, Vol.II, Computer Science Press, 1988.

[6] R. H. L. Chiang, T. M. Barron, and V. C. Storey, Reverse engineering of relational databases: Extraction of an EER model from a relational database, *Data & Knowledge Engineering*, 12:107-142, 1994.

[7] Y. E. Ioannidis and Y. Lashkari, Incomplete path expressions and their disambiguation, SIGMOD'94, pp.138-149.

[8] J. A. Wald and P. G. Sorenson, Resolving the query inference problem using Steiner trees, *ACM Transactions on Database Systems*, Vol.9, No. 3, pp. 348-368, Sept. 1984.

[9] W. W. Chu, H. Yang, K. Chiang, M. Minock, G. Chow, C. Larson, CoBase: A Scalable and Extensible Cooperative Information System, *Journal of Intelligent Information Systems*, 1996.

[10] W. W. Chu and K. Chiang, Abstraction of high level concepts from numerical values in databases. In *Proceedings of AAAI Workshop on Knowledge Discovery in Databases*, Seattle, 1994.

[11] M. Merzbacher and W. W. Chu, Pattern-based clustering for database attribute values. In *Proceedings of AAAI Workshop on Knowledge Discovery in Databases*, Washington, D.C., 1993.