Estimation of Information Sources Pertinence in Mediated Data Integration Systems

Wissem Labbadi  Jalel Akaichi
Wissem.Labbadi@yahoo.fr  Jalel.Akaichi@isg.rnu.tn

Computer Sciences Department  ISG University of Tunis
41 Avenue de la Liberté Bouchoucha, Bardo  TUNISIA

Abstract: Solutions to efficient fuzzy queries processing in decentralized environments were mostly proposed for a relational databases context. Moreover, most of related works process fuzzy queries without considering cooperation between distributed sources. To overcome these drawbacks, we propose a two-step method able to determine as efficiently as possible the set of rewritings that are likely to provide satisfactory answers to a conjunctive fuzzy query submitted to the mediated schema managing distributed and heterogeneous information sources. In the first step, an attempt is made to associate each query rewriting with histograms summarizing the distributions, in the query result, of the attributes to which fuzzy conditions are related. In the second step, histograms associated to large-scale distributed heterogeneous data sources are used to estimate the score of query rewritings. Score associated to each rewriting determines how closely the rewriting matches the user query and whether it is interesting to process this rewriting or not.

Keywords: conjunctive fuzzy query processing, mediated data integration systems, histogram-based query rewritings summaries, fuzzy query/summary matching, satisfactory answers.
1 Introduction

In web applications, user queries are expressed in terms of a fixed set of data sources. More precisely, to every query the appropriate information sources from which suitable answers should be retrieved are a priori known and specified. The problem of such systems is if these fixed sources are not available for a given reason or whether it is impossible to connect to them, the user query will not be satisfied and the application becomes useless. To overcome the limits of classical queries more flexible ways to access information are needed. The sources from which data are retrieved to answer a query are not a priori fixed, but, they are detected with a dynamic way according to their availabilities and pertinence degrees to answer the query. To do that, before answering a user query this latter is submitted to a mediator which determines the relevant sources among a collection of related data sources and provides transparent access to them. Such a system relieves the user from the burden of locating the data sources relevant to a query [10]. It also favors the extensibility of the number of data sources, so that adding new sources doesn't require changing the definition of the query and as a consequence the chance to answer such a query, even not in a completely way, when some of the sources are not available becomes higher. But, in front of the fast growth of data stored at large scale distributed sources, users are becoming not able to keep sufficient knowledge about databases contents and it's getting very difficult to find information unless one knows exactly where to get it from. Therefore, traditional techniques for querying data sources are pushed to their limits.

The idea suggested is to introduce some flexibility in the definition of user queries. To do that several solutions have been proposed among which the one based on fuzzy sets [1] where a user can incorporate in his query vague and linguistic terms, represented by fuzzy sets, to express his preferences and the mediated system takes care of finding the different combinations of sources which are applicants for providing satisfactory answers. But the constraint here is that such a system doesn't have any information about the quality of the results returned by these combinations.

A classical method to process a conjunctive fuzzy query by a mediator is to consider all the combinations of sources, retrieves from each one the satisfactory local answers, and then merges them with results from other combinations at a common site in order to provide the user with the overall best answers. But, in practice, among the identified sources there are many which are not containing any of the desired answers and so, querying these sources will incurs unnecessary cost especially when the number of the useless sources becomes large [2]. As a consequence, mediated systems need to surpass classical methods of processing queries submitted to their mediated schemas; hence the idea is to query integrated information sources through their summaries.

Let us mention that solutions to flexible querying of multiple data sources through their summaries were only proposed for a relational databases context [2, 3, 9].
Moreover, these related works process fuzzy queries without considering cooperation between distributed sources. To overcome these drawbacks, we propose in this work to extend the histogram-based approach presented in a decentralized relational environment to conjunctive fuzzy queries, which raises the problem of finding the different combinations of sources which can provide satisfactory answers. Since we are interested here to data integration systems in which the contents of the sources are described as views over the mediated schema, the problem of finding the different satisfying combinations of sources becomes the problem of finding, using a set of views, the different rewritings of the user query which refer to these combinations of sources.

The contribution of this paper can be summarized in a two-step method to determine as efficiently as possible the set of rewritings that are likely to provide satisfactory answers to a conjunctive fuzzy query submitted to the mediated schema. In the first step, an attempt is made to associate each query rewriting with histograms summarizing the distributions of the attributes of interest (attributes to which fuzzy conditions are related) in the query result. In the second step, histograms are used to estimate the score of query rewritings. Score associated to each rewriting determines how closely the rewriting matches the user query and whether it is interesting to process this rewriting or not.

The remainder of this paper is organized as follows. Section 2 describes related works. Section 3 presents the contribution of this paper as a solution to find the different combinations of sources which can provide satisfactory answers to a conjunctive fuzzy query. Finally, section 4 concludes, and outlines some of the future works in this area.

2 Related work

To the best of our knowledge, the problem of flexible querying of large number of information sources through their summaries has been dealt with first in [9] and [2] respectively in a context of k Nearest Neighbors queries and Top-k queries where the authors propose to use histograms to approximate the frequency distributions of values in the most queried attributes of data sources (relations) and based on them rank distributed relational databases with respect to a given query based on the estimated distance of the best matched tuple in each database. The distance is a measure of how well a tuple satisfies a query [2]. Having available distances of the best matched tuples in the different databases enables to rank the databases in ascending order of these distances. Then the databases are accessed in the order in which they are ranked, one at a time to select the k best matched tuples of the query. Two merge algorithms are proposed to determine which databases should be accessed from the ranked ones and what tuples from accessed databases should be returned. The first algorithm, Merge-1 was proposed in [9] and the second algorithm is MIN-2 and was proposed in [2].
In [3], authors extended the previous works to deal with the problem of querying distributed information sources, through fuzzy summaries, by means of fuzzy queries instead of top-k queries which constitute only a special case of fuzzy ones. Different approaches to fuzzy summaries which have been proposed by [4, 5, 6, 7, 8] were presented and it has been shown for each of these approaches how a fuzzy query can be matched against a fuzzy summary in order to assess the interestingness of a given data source with respect to the considered query. A fuzzy query matching algorithm was also proposed in [3] for the summary method based on histograms, which is used in [2] in a context of top-k query processing, providing an approximation of the average satisfaction degree of a fuzzy condition by a tuple of the relation (source) containing the summarized attribute to which the considered fuzzy condition is related.

In all the mentioned works [9, 2, 3], solutions to efficient flexible queries (k Nearest Neighbors queries, Top-k queries and fuzzy queries) processing in decentralized environments were only proposed for a relational databases context. These works didn't deal with the diversity of the distributed information sources and their heterogeneity while it is one of the main difficulties met by web users. Another drawback is that the fuzzy queries considered in [9, 2, 3] constitute a particular case where they are requiring no cooperation from distributed sources to be answered. That is, each source which is likely to contain the desired answers is considered able to answer the query in isolation from other sources. Note that, in general, to form a query answer users have to combine data from different sources. A challenge is to find the different combinations of sources which can provide satisfactory answers.

### 3 Estimation of the pertinence of sources combinations

In order to estimate the desirability of each distribution of values with respect to a given fuzzy condition in user queries, related works use the histograms summarizing the frequency distributions of values in the different sources containing the attribute of interest each time when they proceed to the fuzzy query/summary matching since the different distributions of values which they want to assess their desirability with respect to the fuzzy condition are those in the distributed sources containing the attribute of interest.

However, in general, the frequency distributions of the attributes of interest in the queries results are different than those in the sources, the case of the conjunctive queries. The change in the frequency distributions is returned to joining the sources descriptions when reformulating the user query, posed over the mediated schema, into queries that refer directly to sources schemas.

As a consequence, existing solutions based on histograms are not sufficient to determine the different combinations of sources which can provide satisfactory answers to conjunctive fuzzy queries.
The idea suggested is to estimate the construction for each conjunctive fuzzy query the histograms summarizing the distributions of values, in the query result returned by the different combinations of sources, to which fuzzy conditions of the considered query are related.

In this section, we present a method based on the uniform distribution for constructing the desired histograms and based on them to assess the interestingness of each sources combination with respect to the query without processing it against them. Very promising results are obtained using this method.

3.1 Running Example

Let us consider a set of centers giving trainings in computer sciences. Each center maintains a database accessible to the public through a web site. Each database lists all the courses taught in the corresponding center and provides information about these courses and trainers who taught them. Let us consider also a conjunctive fuzzy query submitted to a mediated system integrating the heterogeneous training centers databases.

In a training center schema, a course is taught during a quarter to a unique level classes among three possible ones (level 1, level 2 and level 3). Each course has a title and a registration_cost which is a sum of money paid by students who registered in this course. The Teaches relation lists all the courses being taught by trainers during the different quarters. An evaluation value is a note associated to a course taught by a given trainer during a given quarter.

\[
\text{Course (course_id, title, class_level, registration_cost)}
\]
\[
\text{Trainer (trainer_id, name, area)}
\]
\[
\text{Teaches (trainer_id, course_id, quarter, evaluation).}
\]

The mediated schema exposed to the user is the training center schema except that the relations Teaches and Course have an additional attribute "t_center" identifying the training center at which a course is being taught:

\[
\text{Course (course_id, title, class_level, registration_cost, t_center)}
\]
\[
\text{Teaches (trainer_id, course_id, quarter, evaluation, t_center).}
\]

To illustrate the solution, suppose we have the following two data sources. The first source s1 lists all the courses taught in a given training center and their registration costs. This source is described in the mediator by the following view definition:

\[
\text{create view v1 as}
\text{select course_id, title, class_level, registration_cost}
\text{from Course.}
\]

The second source s2 lists among the courses provided by source s1 those being taught at the same training center with their evaluations, and is described by the following view definition:
create view v2 as
    select course_id, trainer_id, quarter, evaluation
    from Teaches.

If a user wants to know the courses having not expensive registration costs and having good evaluations, he submits to the data integration system the following query:

    select title, evaluation, t_center
    from Course, Teaches
    where Course.course_id = Teaches.course_id and
        registration_cost is NOT EXPENSIVE and evaluation is GOOD.

The data integration system is able to answer the previous query by joining the two sources s1 and s2. These sources are determined by the system after reformulating the query, posed over the mediated schema, into one that uses the views v1 and v2 because the view v1 (respectively v2) mentions the relation Course (resp. Teaches) which is mentioned by the query and also selects the attribute title (resp. evaluation) which is selected by the query. (For more information about usability of a view to answer a query see [10]). This query is the following:

    select title, evaluation
    from v1, v2
    where v1.course_id = v2.course_id and
        registration_cost is NOT EXPENSIVE and evaluation is GOOD.

But, the data integration system doesn't have any information about the quality of the result returned by the query that refers directly to the schemas of the sources s1 and s2 as well results returned by the queries that refer to the schemas in the other relevant sources that are integrated in the system. Consequently, the data integration system needs a more efficient way to answer a fuzzy query posed in terms of its mediated schema; hence the idea of querying the integrated data sources through their summaries.

As it's mentioned before, the paper [3] studied different approaches to data sources summaries and it shows by applying each one over an example that the non fuzzy method based on histograms, which is used in [2] in a context of top-k query processing, is very efficient to focus on the "promising" sources only.

Let us consider again the two sources s1 and s2. Let H1 be a histogram describing the distribution of the registration cost values present in s1 and let H2 be a histogram describing the distribution of the evaluation values present in s2. Since there are several offerings of the same course in a given training center, each time an evaluation is associated to this course. Hence, such a course may have many evaluations. For constraints imposed by the solution in this paper, the evaluation values described by a histogram are grouped by course_id and aggregated on average so that each course must have a single evaluation value.
Starting from the join condition \( v1.course_id = v2.course_id \) applied to the views \( v1 \) and \( v2 \) in the query referring to the sources \( s1 \) and \( s2 \), and from the histograms \( H1 \) and \( H2 \), we can understand that the query result doesn't contain all the courses offered at the training center because only five courses from the view \( v1 \) will be joined with those from \( v2 \) since \( H1 \) describes the distribution of the registration cost values of 10 courses when \( H2 \) describes the distribution of the evaluation values only for 5 courses among those present in \( s1 \).

As a consequence, to assess the ability of the combination of the two sources \( s1 \) and \( s2 \) to provide satisfactory answers to the query, we don't have to include the not joined courses in the satisfaction computing process since they are not included into the query answers. We can see according to the tables TABLE.1 and TABLE.2 describing respectively the content of the sources \( s1 \) and \( s2 \) that the courses \textit{Linux Operating Systems}, \textit{Artificial Intelligence}, \textit{UML}, \textit{Operating Systems Foundation} and \textit{Multimedia} are not considered by the query and the fact of incorporating them in the satisfaction computing process will deceive the average satisfaction value estimated for the query result returned by the combination of \( s1 \) and \( s2 \).
<table>
<thead>
<tr>
<th>Course_id</th>
<th>Title</th>
<th>Registration_cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>c1</td>
<td>Database systems</td>
<td>300</td>
</tr>
<tr>
<td>c2</td>
<td>Computing Networks</td>
<td>250</td>
</tr>
<tr>
<td>c3</td>
<td>UML</td>
<td>120</td>
</tr>
<tr>
<td>c4</td>
<td>Multimedia</td>
<td>80</td>
</tr>
<tr>
<td>c5</td>
<td>Operating Systems Foundation</td>
<td>200</td>
</tr>
<tr>
<td>c6</td>
<td>Algorithm and Programming</td>
<td>150</td>
</tr>
<tr>
<td>c7</td>
<td>XML Technology</td>
<td>100</td>
</tr>
<tr>
<td>c8</td>
<td>Linux Operating systems</td>
<td>230</td>
</tr>
<tr>
<td>c9</td>
<td>Information Security</td>
<td>220</td>
</tr>
<tr>
<td>c10</td>
<td>Artificial Intelligence</td>
<td>270</td>
</tr>
</tbody>
</table>

TABLE.1  Content of source s1

For this reason, an attempt is made in this paper to assess each rewriting relatively to a fuzzy query by summarizing the distribution of values present in the attributes of interest and which are related only to the retrieved answers. We present a two-step method to construct the histograms summarizing the desired distributions.

In the first step, an attempt is made to estimate the size of the result returned by the rewriting to answer the query. In the second step, a solution based on the uniform distribution of the attributes of interest in the individual sources is proposed to construct the desired histograms.

The first step determines the size of the results returned by all the possible query rewritings without processing them against the integrated data sources. So, in order to estimate the result size of a given query rewriting, this step is based on histograms used to approximate the frequency distributions of values in the attributes of interest present in the relations referred by this rewriting. A uniform distribution is assumed within each interval of the histograms. That is, the frequency of a value in an interval is approximated by the average of the frequencies of all values in the interval [12].

Before presenting any results on the estimation of query results size, we introduce some notation regarding a histogram $H$, with $m$ intervals, approximating the frequency distribution of an attribute $A$:

$H[I_k]$  the number of tuples whose $A$-value is in the interval $I_k$. 

<table>
<thead>
<tr>
<th>Course_id</th>
<th>Evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td>c1</td>
<td>18</td>
</tr>
<tr>
<td>c6</td>
<td>12</td>
</tr>
<tr>
<td>c2</td>
<td>17</td>
</tr>
<tr>
<td>c9</td>
<td>10</td>
</tr>
<tr>
<td>c7</td>
<td>8</td>
</tr>
</tbody>
</table>

TABLE.2  Content of source s2
V (I_k, A)  the number of distinct values the attribute A has in the interval I_k (assumed to be known for each interval of each histogram).

3.2 Histogram Effectiveness for Estimation Problems

3.2.1 Result Size of Selection Queries

The result size of a specific selection query (σ_{A="a"}) applied on an attribute A (or combination of attributes) of interest to which a histogram H is maintained to approximate its frequency distribution, is estimated by the following formula where the constant "a" belongs to the interval I_k:

\[ T (\sigma_{A="a"}) = H [I_k] / V (I_k, A) \]

This formula is still the best estimate on the average, even if the values in the interval I_k are not uniformly distributed (Zipfian or other distribution), but as long as the constant "a" in the selection condition is chosen randomly [13].

3.2.2 Result Size of Equality Join Queries

Starting from an equality join query Q applied to the sources s1 and s2 on the attribute A, we would like to determine the size of the query Q without processing Q against s1 and s2. We use for this purpose the histograms H1 and H2 approximating the frequency distributions of the attribute A respectively in the sources s1 and s2 to return the closest approximation to the result size of Q. Here we assume the same partition of the domain of A into fixed sized intervals I_1, ... I_m in the two sources s1 and s2 and we also assume a uniform distribution within each interval of each histogram.

The idea is to compute the result size of joining the values of each interval I_k (k=1 ... m) of H1 with those belonging to the interval I_k of H2, and then the different sizes are summed in order to obtain the size of the query result.

\[ T (Q) = \sum_{k=1}^{m} H1 [I_k] * H2 [I_k] / \max (V (I_k, s1(A)), V (I_k, s2(A))) \]

3.2.3 Distribution of the attribute "registration_cost" in the query result

Let us denote H_{res} the histogram approximating the distribution of the attribute registration_cost in the query result. The idea is to compute, for each interval I_k (k=1 ... 3), the number of tuples H_{res}[k] whose registration_cost values is in I_k.

- H_{res}[1] = \frac{(V(I_1, registration_cost) \times \text{query result size})}{\sum_{k=1}^{3} V(I_k, registration_cost)} = \frac{(2 \times 5)}{(2 + 3 + 5)} = \frac{10}{10} = 1.
- H_{res}[2] = \frac{(V(I_2, registration_cost) \times \text{query result size})}{\sum_{k=1}^{3} V(I_k, registration_cost)} = \frac{(3 \times 5)}{(2 + 3 + 5)} = \frac{15}{10} = 1.5 \approx 2.
- H_{res}[3] = \frac{(V(I_3, registration_cost) \times \text{query result size})}{\sum_{k=1}^{3} V(I_k, registration_cost)} = \frac{(5 \times 5)}{(2 + 3 + 5)} = \frac{25}{10} = 2.5 \approx 3.
The histogram estimating the frequency distribution of the attribute registration_cost in the query result is the following:

3.2.4 Distribution of the attribute "evaluation" in the query result

Since the attribute course_id in the relation Teaches is a foreign key referring to the primary key (course_id) in the relation Course and each course occurs in the relation Teaches only once (courses are grouped by course_id), then the query result is formed by the same courses present in the relation Teaches. As a consequence, the frequency distribution of the attribute evaluation in the query result is the same that the one in the relation Teaches and so the histogram summarizing the distribution of the attribute evaluation is similar to the one summarizing the frequency distribution of the attribute evaluation in the relation Teaches.

The histogram estimating the frequency distribution of the attribute evaluation in the query result is the following:

Having available the desired histograms for the different query rewritings enables to determine how closely each rewriting matches the fuzzy conditions specified in the user query and hence, to keep only those which can provide satisfactory answers.
3.3 Fuzzy Query /Histogram-Based Summary Matching

In this section, we show how a fuzzy query, posed over the mediated schema, can be matched against the histogram based approximated distribution (s) of the attribute (s) of interest in the result of each query rewriting in order to find the different combinations of sources which can provide satisfactory answers to the considered query.

The idea is to compute the satisfaction value of each fuzzy predicate by the query rewriting result. We use for this purpose the algorithm used in [3] in a relational context to approximate the average satisfaction degree of a fuzzy predicate by a tuple of relational data. Then, the different average satisfaction degrees can be aggregated in order to have a global vision of the query rewriting.

**Example 2.** Continuing on with the query rewriting referring to the sources s1 and s2. Let us consider again the two histograms H1 and H2 and the two fuzzy sets "NOT EXPENSIVE" and "GOOD" defined by the following fuzzy membership functions:

![Histograms](image)

The algorithm provides the average satisfaction degrees respectively of the two fuzzy predicates "NOT EXPENSIVE" and "GOOD":

\[
((1 \ast 1) + (0.495 \ast 2) + (0 \ast 3))/6 = 1.99/6 \approx 0.33.
\]

\[
((0.35 \ast 2) + (0.89 \ast 3))/5 = 3.37/5 \approx 0.67.
\]

The actual average satisfaction degrees respectively of the fuzzy predicates "NOT EXPENSIVE" and "GOOD" equals 1.5/5 = 0.3 and 4.14/5 \approx 0.82. In this particular case, the degrees computed (0.33) and (0.67) are relatively good approximations.

4 Conclusion and future works

In this paper, we presented a two-step solution for finding the different combinations of sources which can provide satisfactory answers to conjunctive fuzzy queries. In the first step, an attempt is made to associate each combination with histograms to estimate the score of the combination of sources with respect to the query.
For every combination, we used histograms to approximate data distributions in individual sources and based on them estimate query result sizes and construct histograms estimating data distributions in the result returned by the considered combination.

Here, we considered that histograms associated to data sources are making the uniform distribution assumption. Such a histogram is called *trivial* and this assumption, however, rarely holds in real data and estimates based on it usually have large errors [14, 15]. As a perspective, we plan to use classes of optimal histograms (those with least error in their estimates) rather than trivial ones and study the effectiveness of these optimal histograms. There has been considerable works done on identifying classes of optimal histograms for the estimation problems. Yannis Ioannidis and Stavros Christodoulakis presented in [16] several results which showed that the class of *serial* histograms is close to optimal, and effective in estimating sizes and value distributions in query results. In [17], it has been shown that estimations of histograms belonging to the class of *equi-width* histograms are often better than trivial ones. Piatetsky-Shapiro and Connell introduced in [18] the class of *equi-depth* (or *equi-height*) histograms and showed that equi-width histograms have a much higher worst-case and average error for a variety of selection queries than equi-depth histograms.

**References**


