Distance-Based Fuzzy Relations in Flexible Query Answering Systems: Overview and Experiences

Ulrich Bodenhofer¹, Peter Bogdanowicz², Gerhard Lanzerstorfer³, and Josef Küng⁴

¹ Software Competence Center Hagenberg, A-4232 Hagenberg, Austria
ulrich.bodenhofer@scch.at

² Infomatic, A-4614 Marchtrenk, Austria
office@infomatic.at

³ ING DiBa Direktbank Austria, A-4060 Leonding, Austria
gerhard.lanzerstorfer@ing-diba.at

⁴ Institute for Applied Knowledge Processing
Johannes Kepler University, A-4040 Linz, Austria
jkueng@faw.uni-linz.ac.at

Abstract. This paper provides a brief overview of OVQS, a framework for flexible query answering systems based on distance-based fuzzy relations. The necessary theoretical and methodological background is given. The concepts are illustrated by means of a practical case study.

1 Introduction

The use of classical binary logic for data retrieval poses severe limitations. Firstly, real-world data, in particular, numeric data, are often perturbed by noise or other errors. This may result in unstable behavior in the sense that minimal variations of the data can change the result of a query dramatically. Secondly, no structural information is available about how close a rejected record was to the fulfillment of a query. This loss of information is particularly harmful if the user would still be interested in potentially close records if the query gives an empty result. These two demands have created an own discipline that is concerned with how query interfaces can be extended such that a flexible interpretation of queries is possible (see e.g. [5, 6, 12] for recent overviews and further references to literature)—in particular, with the motivation to suggest alternatives which are close to matching the criteria in case that a query gives an empty result. This area is often referred to as “flexible querying”.

The given paper provides an overview of OVQS, a framework for flexible querying based on fuzzy relations. OVQS uses fuzzy equivalence relations to represent gradual similarity and, consequently, fuzzy orderings to allow flexible interpretation of ordinal queries. In order to circumvent prohibitive storage requirements for representing fuzzy relations, OVQS adopts the basic approach of the established Vague Query System (VQS), that is, to model gradual similarity
by means of distances. The seamless integration of distances in the framework of fuzzy relations is achieved by means of existing results from the theory of fuzzy relations. Furthermore, this paper presents a practical case study from a student project, where the ideas behind OVQS were used to implement a prototype of a flexible query answering system for a database containing second-hand cars.

2 Theoretical and Methodological Background

The use of fuzzy relations in flexible querying has one significant shortcoming in terms of practical feasibility: fuzzy relations, in their most general form, need to be represented by means of tables containing the degrees of relationship between the records, which requires large amounts of storage. A pragmatic, yet effective, approach that makes efficient use of the resources available is to use distances for representing the gradual similarity of records. A well-developed concept that has been established outside the framework of fuzzy relations is the so-called Vague Query System (VQS) [11]. The basic idea behind VQS is to transform even non-numeric data into a numeric representation. This so-called Numeric Coordinate Representation (NCR) serves as the means to compute the similarity between records by using ordinary (e.g. Euclidean) distances, thus overcoming the need for large similarity tables.

As highlighted in [3], VQS has two shortcomings: (1) distances are automatically normalized on the basis of actual data, thus distances are difficult to compare for different attributes; (2) a flexible interpretation of ordinal queries like “at least”, “at most”, etc. is not straightforward. The key to enriching VQS by ordinal constructs and overcoming the comparability issue is to redraft and extend VQS in the framework of fuzzy relations. We start from the well-known concept of a fuzzy equivalence relation, which is a straightforward choice for modeling gradual similarity [4, 14]. In the remaining part of the paper, we make use of triangular norms (t-norms) as generalized models of conjunction [10].

Definition 1. A binary fuzzy relation $E : X^2 \rightarrow [0, 1]$ is called fuzzy equivalence relation with respect to a t-norm $T$, for brevity $T$-equivalence, if and only if the following three axioms are fulfilled for all $x, y, z \in X$:

(i) Reflexivity: $E(x, x) = 1$
(ii) Symmetry: $E(x, y) = E(y, x)$
(iii) $T$-transitivity: $T(E(x, y), E(y, z)) \leq E(x, z)$

The question arises how to transform distances into a fuzzy equivalence relation in a meaningful way. For this purpose, a well-established result is available if the t-norm $T$ under consideration is continuous Archimedean$^1$ [10].

$^1$ Simplistically, this means that $T$ is continuous and fulfills $T(x, x) < x$ for all $x \in [0, 1]$; such a t-norm can always be represented by means of a so-called additive generator, i.e. a continuous and strictly decreasing bijection $f : [0, 1] \rightarrow [0, \infty]$, such that the representation $T(x, y) = f^{-1}(\min(f(x) + f(y), f(0)))$ holds.
Theorem 1. [8] Consider a continuous Archimedean t-norm T with additive generator f, a pseudo-metric d : X² → [0, ∞], and a real constant C > 0. Then the following mapping is a T-equivalence:

\[ E_{d,C}(x, y) = f^{-1}\left( \min\left( \frac{1}{C} \cdot d(x, y), f(0) \right) \right) \] (1)

By means of Theorem 1, we achieve a perfect synergy: we are able to formulate gradual similarity in the framework of fuzzy relations in a well-founded way, still being able to use distances as the basis for calculating similarity. Thus, we do not need similarity tables to represent gradual similarity. For non-numeric attributes, VQS’s NCR approach is still usable.

There is a well-developed theory of fuzzy orderings that integrates seamlessly with the theory of fuzzy equivalence relations [1, 2, 9]. Thus, this class of fuzzy relations is a natural choice to achieve a flexible interpretation of ordinal queries.

Definition 2. A fuzzy relation \( L : X^2 \rightarrow [0, 1] \) is called fuzzy ordering with respect to a t-norm T and a T-equivalence E, for brevity \( T\)-E-ordering, if and only if it is T-transitive and fulfills the following two axioms for all \( x, y \in X \):

(i) \( E\)-Reflexivity: \( E(x, y) \leq L(x, y) \)
(ii) \( T\)-E-antisymmetry: \( T(L(x, y), L(y, x)) \leq E(x, y) \)

A \( T\)-E-ordering \( L \) is called strongly complete if \( \max(L(x, y), L(y, x)) = 1 \) for all \( x, y \in X \).

Theorem 2. [1] Consider a fuzzy relation \( L \) on a linearly ordered domain \( X \) and a T-equivalence \( E \) on \( X \). Further assume that the linear ordering and \( E \) are compatible in the sense that \( E(x, z) \leq \min(E(x, y), E(y, z)) \) for all linearly ordered three-element chains \( x \leq y \leq z \). Then the following fuzzy relation is a strongly complete \( T\)-E-ordering:

\[
L(x, y) = \begin{cases} 
1 & \text{if } x \leq y \\
E(x, y) & \text{otherwise}
\end{cases}
\]

Theorem 2 particularly implies that the “combination” of a crisp linear ordering and a compatible fuzzy equivalence relation has a clear theoretical interpretation as a vague concept of ordering (a “linear ordering with imprecision”) [2]. The only question remaining is how a fuzzy equivalence relation can be constructed from a (pseudo-)metric such that compatibility with a given crisp ordering is fulfilled. It is not difficult to prove that, if

\[ x \leq y \leq z \Rightarrow d(x, z) \geq \max(d(x, y), d(y, z)) \] (2)

holds for all \( x, y, z \in X \), the fuzzy equivalence relation \( E_{d,C} \) defined as in (1) is compatible with \( \leq \) such that Theorem 2 can be applied [2].
3 An Overview of OVQS

Like VQS, OVQS is designed as a proxy between the user and a SQL-capable relational database. OVQL, the language of OVQS, extends SQL by conditions that can be interpreted in a flexible way as follows:

\[
\text{VQLExpression} := \text{"SELECT FROM" \ DataSource \ "WHERE" \ Conditions \ "INTO" \ destinationTableName;}
\]

\[
\text{DataSource} := ([\text{ownerName}\.\text{"\}]\text{rootTableName}) \ |
\text{([ownerName}\.\text{"\}]\text{rootViewName}) \ |
\text{\("\text{sqlSelectStatement}\"\)};
\]

\[
\text{Conditions} := \text{Condition} \ {\{"\text{AND}\" \ Condition\}};
\]

\[
\text{Condition} := \text{NonNumericCond \ ParameterExpression} \ |
\text{NumericCond \ ParameterExpression};
\]

\[
\text{NonNumericCond} := \text{columnName \ "IS" \ alphaNumericValue;}
\]

\[
\text{NumericCond} := \text{columnName \ "IS" \ numericValue} \ |
\text{columnName \ "IS AT LEAST" \ numericValue} \ |
\text{columnName \ "IS AT MOST" \ numericValue} \ |
\text{columnName \ "IS WITHIN \ (" \ numericValue \ "," \ numericValue \ ")\"};
\]

\[
\text{ParameterExpression} := \text{["TOLERATE UP TO" \ numericValue]} \ |
\text{["WEIGHTED BY" \ numericValue]};
\]

As obvious from the syntax, there is an explicit distinction between numeric and non-numeric attributes. Like in VQS, we may assume that there is an underlying NCR for all non-numeric attributes. We are hence able to compute Euclidean distances for all attributes. Moreover, assume that there is a default radius of interest for each attribute that may be overridden with the optional "TOLERATE UP TO" parameters.

For defining the corresponding semantics, assume that we are given a continuous Archimedean t-norm \(T\) with additive generator \(f\) (if \(f(0) < \infty\), we assume without any loss of generality that \(f(0) = 1\)). Then, for a given non-numeric column, a condition "\(x \text{ IS q}\)" is evaluated in the following way: for a concrete value \(x_0\), the degree to which \(x_0\) fulfills the condition is computed as

\[
E_{d,C}(x_0, q) = f^{-1}\left(\min\left(\frac{1}{C} \cdot d(x_0, q), f(0)\right)\right),
\]

where \(d\) is a metric for the column under investigation which is constructed using an NCR. The parameter \(C\) is the radius of interest defined for the respective column.

For a numeric attribute, we are able to define the semantics of the four atomic conditions as

\[
t("x \text{ IS q}\" \mid \text{x}_0) = E_{C}(x_0, q)
\]

\[
t("x \text{ IS AT LEAST q}\" \mid \text{x}_0) = L_{C}(q, x_0)
\]

\[
t("x \text{ IS AT MOST q}\" \mid \text{x}_0) = L_{C}(x_0, q)
\]

\[
t("x \text{ IS WITHIN (a, b)\" \mid \text{x}_0) = \min\left(L_{C}(a, x_0), L_{C}(x_0, b)\right)
\]

18
with

\[ EC(x, y) = f^{-1}\left(\min\left(\frac{1}{2} \cdot |x - y|, f(0)\right)\right) \]

and \( LC(x, y) = \begin{cases} 1 & \text{if } x \leq y, \\ EC(x, y) & \text{otherwise.} \end{cases} \)

The question remains how the semantics of the “AND” connective is modeled, i.e. how the degrees of fulfillment of multiple atomic conditions are aggregated. Assume that a query consists of \( n \) atomic conditions and that the degrees to which a given record fulfills the \( i \)-th condition is \( t_i \). Further assume that the default weights for all attributes are equal to 1. Using the optional “WEIGHTED BY” parameter, the user can assign individual degrees of importance to atomic conditions. With a weight \( \tilde{w}_i > 1 \), he/she can strengthen the importance of the \( i \)-th condition. With a weight \( \tilde{w}_i < 1 \), he/she weakens the importance of the condition. Then a pseudo-arithmetic mean with respect to the additive generator \( f \) is used to compute the overall degree of fulfillment (detailed argumentation why this is justified and advisable can be found in [3, 13]):

\[ A_w(t_1, \ldots, t_n) = f^{-1}\left(\min(f(0), \sum_{i=1}^{n} w_i \cdot f(t_i))\right) \quad \text{with} \quad w_i = \frac{\tilde{w}_i}{\sum_{i=1}^{n} \tilde{w}_i}. \quad (8) \]

This total degree of fulfillment is computed for each record in the table under consideration. Finally, the degrees of fulfillment are sorted and the list of records is presented to the user in descending order (better fitting records first).

One degree of freedom is still open—the choice of the t-norm \( T \). Continuous Archimedean t-norms are either strict (i.e. \( f(0) = \infty \)) or nilpotent (i.e. \( f(0) < \infty \)) [10]. It can be shown that all strict t-norms behave the same [3], i.e. the matching degrees may differ, but the obtained final ranking lists are the same. Therefore, \( TP \) is the canonical choice if one opts for using a strict t-norm. If a nilpotent t-norm is chosen, the particular choice does have influence on the result. From a practical perspective, however, \( TL \) is a pragmatic and justifiable choice. Choosing a nilpotent t-norm has the advantage that, for a given condition, the tolerance radius \( C \) has a clear and unambiguous interpretation. However, any information outside this radius is lost, which is not the case for strict t-norms.

4 A Practical Case Study

The concepts introduced in this paper have been evaluated with a prototype implemented by the second and third author. The goal was to develop a flexible query answering interface to a relational database containing cars for sale.

The most important table in the database is the list of available cars. This table has 53 columns and a total of approx. 65000 rows/records. Technical data, features, age, mileage, and the zip code where it is available can be stored for

\[ 2 \text{ in case } T \text{ is the Lukasiewicz t-norm } TL(x, y) = \max(x + y - 1, 0) \text{ (i.e. } f(x) = 1 - x), \]

the ordinary weighted arithmetic mean is obtained. In case that \( T \) is the product t-norm \( TP(x, y) = x \cdot y \) (i.e. \( f(x) = -\ln x \)), the weighted geometric mean is obtained.
Table 1. Intermediate query result before flexible interpretation; the rightmost column provides the distance from Linz (zip 4020).

<table>
<thead>
<tr>
<th>#</th>
<th>Location</th>
<th>HP</th>
<th>Year</th>
<th>Mileage (km)</th>
<th>Price (EUR)</th>
<th>Distance (km)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4364 St. Thomas</td>
<td>90</td>
<td>1994</td>
<td>164000</td>
<td>3750</td>
<td>35</td>
</tr>
<tr>
<td>2</td>
<td>4232 Müenzbach</td>
<td>116</td>
<td>2000</td>
<td>120000</td>
<td>13950</td>
<td>31</td>
</tr>
<tr>
<td>3</td>
<td>4871 Zipf</td>
<td>101</td>
<td>2000</td>
<td>17500</td>
<td>18500</td>
<td>64</td>
</tr>
<tr>
<td>4</td>
<td>4651 Stadl-Paura</td>
<td>90</td>
<td>1991</td>
<td>187900</td>
<td>2800</td>
<td>39</td>
</tr>
<tr>
<td>5</td>
<td>4064 Offerling</td>
<td>107</td>
<td>1991</td>
<td>109000</td>
<td>2900</td>
<td>13</td>
</tr>
<tr>
<td>6</td>
<td>5350 Strobl</td>
<td>101</td>
<td>1997</td>
<td>137000</td>
<td>8750</td>
<td>88</td>
</tr>
<tr>
<td>7</td>
<td>5222 Munderfing</td>
<td>90</td>
<td>1996</td>
<td>156000</td>
<td>5900</td>
<td>86</td>
</tr>
<tr>
<td>8</td>
<td>4905 Thomasroith</td>
<td>90</td>
<td>1994</td>
<td>214500</td>
<td>4590</td>
<td>54</td>
</tr>
<tr>
<td>9</td>
<td>4840 Vöcklabruck</td>
<td>110</td>
<td>1998</td>
<td>n.a.</td>
<td>5700</td>
<td>56</td>
</tr>
<tr>
<td>10</td>
<td>4656 Kirchham</td>
<td>116</td>
<td>1991</td>
<td>200000</td>
<td>1600</td>
<td>46</td>
</tr>
<tr>
<td>11</td>
<td>4141 Pfarrkirchen</td>
<td>90</td>
<td>1995</td>
<td>189000</td>
<td>3950</td>
<td>42</td>
</tr>
</tbody>
</table>

The prototype in its current version mainly complies with the principles presented in Section 3, but does not make use of NCRs. For the zip code, a complete distance table is available anyway, so there is no particular need for an NCR. All other categorical attributes are treated in a crisp way without any flexible interpretation. It is possible to choose between two t-norms, $T_L$ and $T_P$. The table contains relatively many missing values. If the respective entry is not available, a condition is considered to be true (i.e. to a degree of 1).

As one example, we consider the following query:

```
SELECT FROM CarTable
WHERE Model IS 'Volkswagen Passat'
AND Layout IS 'Wagon'
AND Location IS 'Linz' TOLERATE UP TO 20
AND HorsePower IS WITHIN (100,110) TOLERATE UP TO 10
AND YearBuilt IS AT LEAST 1998 TOLERATE UP TO 2
AND Mileage IS AT MOST 80000 TOLERATE UP TO 15000
AND Price IS AT MOST 10000 TOLERATE UP TO 1000
INTO ResultSet
```

The first two conditions are referring to categorical attributes that are not interpreted in a flexible way. Hence, we only need to consider records fulfilling those two conditions. Table 1 shows a list of 11 cars to be considered. Then Table 2 shows the results obtained for $T_L$ and $T_P$. In these tables, the columns labeled $t_1,\ldots,t_5$ contain the degrees to which records fulfill the five conditions that are interpreted as described in Section 3. The final matching degree is shown in the last columns labeled $t$. The following rankings are obtained for the query (we
Table 2. Result sets for $T_L$ (left) and $T_P$ (right; numbers rounded to three digits)

<table>
<thead>
<tr>
<th></th>
<th>$t_1$</th>
<th>$t_2$</th>
<th>$t_3$</th>
<th>$t_4$</th>
<th>$t_5$</th>
<th>$t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>1.00</td>
<td>0.20</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.00</td>
<td>0.40</td>
<td>1.00</td>
<td>0.00</td>
<td>0.28</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.00</td>
<td>1.00</td>
<td>1.00</td>
<td>0.00</td>
<td>0.60</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>1.00</td>
<td>0.20</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0.35</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.47</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0.00</td>
<td>1.00</td>
<td>0.50</td>
<td>0.00</td>
<td>0.50</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>1.00</td>
<td>0.20</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>1.00</td>
<td>0.20</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>0.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>0.80</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0.00</td>
<td>0.40</td>
<td>0.00</td>
<td>0.00</td>
<td>0.28</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>1.00</td>
<td>0.20</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$t_1$</th>
<th>$t_2$</th>
<th>$t_3$</th>
<th>$t_4$</th>
<th>$t_5$</th>
<th>$t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.174</td>
<td>0.368</td>
<td>0.135</td>
<td>0.004</td>
<td>1.000</td>
<td>0.126</td>
</tr>
<tr>
<td>2</td>
<td>0.212</td>
<td>0.549</td>
<td>1.000</td>
<td>0.069</td>
<td>0.019</td>
<td>0.173</td>
</tr>
<tr>
<td>3</td>
<td>0.041</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>0.000</td>
<td>0.096</td>
</tr>
<tr>
<td>4</td>
<td>0.142</td>
<td>0.368</td>
<td>0.030</td>
<td>0.001</td>
<td>1.000</td>
<td>0.065</td>
</tr>
<tr>
<td>5</td>
<td>0.522</td>
<td>1.000</td>
<td>0.030</td>
<td>0.145</td>
<td>1.000</td>
<td>0.296</td>
</tr>
<tr>
<td>6</td>
<td>0.012</td>
<td>1.000</td>
<td>0.607</td>
<td>0.022</td>
<td>1.000</td>
<td>0.176</td>
</tr>
<tr>
<td>7</td>
<td>0.014</td>
<td>0.368</td>
<td>0.368</td>
<td>0.006</td>
<td>1.000</td>
<td>0.103</td>
</tr>
<tr>
<td>8</td>
<td>0.067</td>
<td>0.368</td>
<td>0.135</td>
<td>0.000</td>
<td>1.000</td>
<td>0.053</td>
</tr>
<tr>
<td>9</td>
<td>0.061</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>0.571</td>
</tr>
<tr>
<td>10</td>
<td>0.100</td>
<td>0.549</td>
<td>0.030</td>
<td>0.000</td>
<td>1.000</td>
<td>0.056</td>
</tr>
<tr>
<td>11</td>
<td>0.122</td>
<td>0.368</td>
<td>0.223</td>
<td>0.001</td>
<td>1.000</td>
<td>0.093</td>
</tr>
</tbody>
</table>

denote the degree of matching for the $j$-th record/car with $t^j$:

$t^9 > t^3 > t^5 > t^2 > t^{10} > t^4 = t^7 = t^8 = t^{11}$ for $T = T_L$

$t^9 > t^5 > t^6 > t^2 > t^1 > t^7 > t^3 > t^{11} > t^4 > t^{10} > t^8$ for $T = T_P$

Obviously, the rankings do not coincide for the two basic t-norms. For a more detailed explanation and assessment on the use of different t-norms in flexible querying, see [3].

Extensive experiments were carried out with the prototype. The goal was to evaluate the general concept of OVQS and its possible advantages over classical querying. The following points are worth mentioning:

1. The language of OVQS is easy to use and easy to interpret for humans. Even non-skilled persons were easily able to interpret the queries and the result lists.
2. OVQS is computationally efficient, mainly because of its pragmatic approach, i.e. the use of Euclidean distances.
3. At least for numeric attributes, the degrees to which records fulfill queries depend on the query values in a continuous way. Therefore, the approach is robust with respect to noisy data and the choice of a particular query value.

Some issues require further attention. In particular, the negligence of categorical attribute is severe. For some attributes, an NCR would be straightforward. For categorical attributes with a small number of possible instances, distance/similarity tables seem feasible. For the model, none of the two ways is feasible, as the database currently contains around 1000 models from 90 manufacturers. An idea in this direction would be to derive the similarities from the data describing the individual cars by PCA, clustering, or machine learning [7].

21
Acknowledgements

Ulrich Bodenhofer acknowledges support of the Austrian Government, the State of Upper Austria, and the Johannes Kepler University Linz in the framework of the Kplus Competence Center Program. This work was partly carried out as a student project within the Computer Studies Curriculum (B.Sc. Hons.) at University of Derby in Austria, Branch Office Leonding.

References