# Fuzzy Associative Summaries for Databases

Gloria Bordogna
CNR-IDPA
Via Pasubio 5, 24044 Dalmine (BG)–Italy
gloria.bordogna@idpa.cnr.it

Giuseppe Psaila
Università di Bergamo, Facoltà di Ingegneria
Viale Marconi 5, 24044 Dalmine (BG)–Italy
psaila@unibg.it

***Abstract*— The paper introduces the problem of *mining Fuzzy Associative Summaries*. An example of summary is $(\textbf{\textit{Age}}, \textbf{\textit{Medium}}) \rightarrow (\textbf{\textit{Salary}}, \textbf{\textit{Low}}) \rightarrow (\textbf{\textit{Department}}, \textbf{\textit{A}}), 0.34$; it means that when the *Age* is *Medium*, this implies that the *Salary* is *Low*, and this further implies that the *Department* is *A*, with a support of $0.34$.**

**In this paper, we define the notion of fuzzy associative summary, and propose an algorithm to mine them from within a relational table. Then, we discuss our implementation on top of *Soft-SQL*, the extension of the classical SQL query language to express flexible queries based on soft operators and sets of linguistic predicates on classical relational databases [1], and present some experimental results.**

## I. Introduction

Large data sets that describe observations or events cannot be successfully exploited as they are. It is necessary to obtain a synthetic description that *summarizes* the content of the data set, for example, by letting (possibly) unexpected relationships or features emerge [2].

This is the role of *Data Mining* techniques. Popular techniques are clustering, classification, identification of functional dependencies, extraction of association rules [3], [4].

An association rule between two attribute values is a good example of summary that describes an association that frequently occurs in the data set, that is, "when $A$ is $x$ then $B$ is $y$", written $(A, x) \rightarrow (B, y)$. However, when analyzing a data set, a significant problem is posed by the too fine granularity of the continuous and numeric attribute domains, since an exponential number of possible combinations of attribute values have to be analyzed to find out meaningful associations between them [5]. To this end, fuzzy data mining aims at reducing the granularity of the attribute domains to make feasible the discovery of fuzzy association rules [6], [7], [8], [9], [10], [11]. We go a step further, by defining *Fuzzy Associative Summaries* as chains of fuzzy association rules, and by proposing an algorithm to automatically discover them from the analysis of a classical relational data set.

To clarify, consider the following example. Data about employees are stored into a database table called *Employees*. Three possible fuzzy associative summaries might be the following.

$(Age, Medium), Supp = 0.45$
$(Age, Medium) \rightarrow (Salary, Low), Supp = 0.34$
$(Age, Medium) \rightarrow (Salary, Low) \rightarrow (Department, A),$
$Supp = 0.34$

The first summary has length 1 and says that the value of attribute *Age* is *Medium*, with a support of $0.45$ (the support is the frequency with which the summary holds in the table).

The second summary has length 2 and says that the fact that the *Age* is *Medium*, implies that the *Salary* is *Low* with a support of $0.34$; notice that this is a fuzzy association rule.

The third summary has length 3 and is a further step towards the description of more complex relationships: a *Medium Age* implies a *Low Salary* that, in turn, implies that the employee works in *Department A*, with a support of $(0.34)$, giving a clear (and possibly unexpected) evidence of the strength of second summary: the *Salary* of the middle aged employees of *Department* $A$ is always *Low*. This is a chain of fuzzy association rules.

To the best of our knowledge, no previous works have proposed a similar approach.

The contribution of this paper is the definition of the concept of *Fuzzy Associative Summaries* (Section IV), and the proposal of an algorithm to mine them from within a relational table (Section V). In particular, we rely on the general relational model, dealing with both categorical and numerical attributes. We propose to transform the original table into a *Fuzzy-Valued Table*: all the numerical attributes are converted into fuzzy linguistic attributes. Each numerical value $x$, is mapped into a fuzzy value $(lv, \mu_{lv}(x))$, where $lv$ is a linguistic value, and $\mu_{lv}(x) \in [0, 1]$ is the membership degree of $x$ w.r.t. $lv$. Thus, the mining algorithm takes a table in which all attributes are categorical and have an associated membership degree.

We then describe (in Section VI) our implementation of the mining algorithm on top of *Soft-SQL* [1], in order to demonstrate the feasibility of our approach. *Soft-SQL* is an extension of the classical SQL query language to specify flexible queries based on soft operators and sets of linguistic predicates on classical relational databases, described in Section III.

## II. Related Work

In the last two decades, the area of *Data Mining* has been very popular among researchers. An impressive amount of models, techniques and solutions has been proposed; some very popular are clustering, classification, identification of functional dependencies, extraction of association rules. Since its birth [12], extraction of association rules has been object of many research works, that studied the problem from many different perspectives (efficiency [13], problem modeling, integration with relational databases [14], [15], etc.).

The community of researchers working on soft computing studied the problem as well, providing interesting generalizations (see [16] for a synthetic survey). Mainly, fuzzy association rules are introduced to deal with numerical attributes in a flexible way. Fuzzy association rules are defined as fuzzy implications that relate the presence of items in fuzzy transactions [6]. Generally, items are things we can buy and transactions are market baskets containing several items. Numerical attributes are transformed into linguistic values with a given membership degree [17] (a problem already known in crisp approaches too [5]) to effectively find fuzzy association rules. In our work, we do not focus on the problem of discretizing attributes, identifying linguistic values in an automatic way: we assume that linguistic values and their membership functions are provided by the user, as in [9]. We relax the concept of fuzzy transaction to consider all items (tuples) in a database as a single basket.

In our approach, fuzzy association rules relate the presence of items in the whole database, as in [11]. For example, every tuple that contains *large bread* also contains *cheap butter*. Then, fuzzy association rules can be regarded as a form of fuzzy associative summary of the database content.

This interpretation of fuzzy association rules among attributes of tables that do not describe transaction data has been investigated also in [11]. Nevertheless, in our approach, we consider chains of fuzzy association rules, whose length can be the arity of the tuples in the mined relational table.

This paper is a first attempt in obtaining such a kind of descriptions (called *fuzzy associative summaries*); in particular, our proposal exploits sets of linguistic terms to reduce the granularity of the attribute values in classical relational databases, as first proposed in [8].

## III. Soft-SQL

*Soft-SQL* is the extension of SQL for flexible querying on top of classical relational databases [1]. The main idea behind this proposal is the possibility of explicitly defining *sets of linguistic values* for each attribute, named *term sets*. A term set can be used within selection predicates to evaluate the satisfaction degree of tuples.

*Soft-SQL* is implemented on top of *PostgreSQL*, as a Java package that can be used in Java applications. The *SoftSQL package* translates *SoftSQL* statements and flexible queries into crisp SQL queries.

Here, we provide a brief introduction to the main features of *Soft-SQL* that are relevant for this paper.

*Definition of Term Sets. Soft-SQL* clearly provides the concept of *Term Set*, i.e., a named set of linguistic values, that is stored into the database; this way, a term set and its linguistic values can be exploited for formulating queries. Linguistic values are denoted by a *name* and their semantics is defined by means of a trapezoidal function, used to evaluate the membership degree of a valued expression.

To illustrate, consider the *Soft-SQL* definition of term set `Ages` reported in Figure 1, that defines three linguistic values (*Young*, *Middle*, *Old*) for the age of employees.

Let us consider the statements in details.

The `NORMALIZED WITHIN` clause specifies the range of values to normalize in $[0, 1]$; values less than the lower bound (in this case, less than `0`), are treated as the lower bound, while values greater than the upper bound (in this case, greater than `100`) are treated as the upper bound.

The `EVALUATE` clause specifies the name of the attribute (linguistic variable) for which the linguistic values are defined. In the example, the simple expression `Age` is evaluated, i.e., the age is directly evaluated against the normalization range.

Clause `VALUES` specifies the list of linguistic values. Each linguistic value is defined by a name (a string) and by the 4-tuple that reports the coordinates on the $x$-axis of the four key points of the trapezoidal membership function. Figure 2 shows the corresponding trapezoidal functions of the linguistic value in Figure 1.

*Soft Query Language. Soft-SQL* extends the classical `SELECT` statement of SQL to perform flexible querying on classical (not fuzzy) relational databases. A new fuzzy comparison predicate is introduced, so that in the query tuples are associated with a membership degree. To meet the closure property of relational query languages, the resulting tables are again traditional relational tables: to get the membership degree, a predefined attribute named `DEGREE` is available, that refers to the membership degree of a tuple.

Consider the following query, that selects employees such that their age is `'Middle'` (linguistic value in the term set `Ages`).

```
SELECT NAME, DEGREE
FROM Employees
WHERE Age IS 'Middle' IN Ages
ORDER BY DEGREE DESC
```

The `WHERE` clause expresses a soft predicate by means of the `IS...IN` operator: the value of attribute `Age` is evaluated against the linguistic value `'Middle'` defined in the term set `Ages`. In details, through the application of the `NORMALIZED WITHIN` clause, the values of attribute *Age* are normalized in the range $[0, 1]$; Then, the membership degree of each value is obtained by evaluating it against the trapezoidal function defining the semantics of *Middle*.

A tuple is selected if its membership degree is greater than 0. The generated table is obtained by projecting the selected tuples on attributes `Name` and `DEGREE` (the implicit attribute whose value is the membership degree of the tuple); this value is used to sort the selected tuples in descending order.

If we consider the instance of table `Employees` reported in Figure 3, the above query produces the following table.

| **Name** | **DEGREE** |
|---|---|
| Susan | 1 |
| Mary | 1 |
| Kate | 0.6 |

## IV. Fuzzy Associative Summaries

### A. Basic Concepts

**Definition 1**: **Fuzzy-Valued Relation.** $R_F(A_1, \ldots, A_n)$ denotes a *Fuzzy-Valued Relation* (table), and $r_F$ denotes its instance. The domain of an attribute $A_i$ is a set of pairs $(lv, \mu)$,

| | | |
|---|---|---|
| CREATE TERM SET Ages | CREATE TERM SET Salaries | CREATE TERM SET Expertises |
| NORMALIZED WITHIN (0, 100) | NORMALIZED WITHIN (0, 10000) | NORMALIZED WITHIN (0, 10) |
| EVALUATE Age | EVALUATE Salary | EVALUATE Expert |
| WITH PARAMS Age AS Integer | WITH PARAMS Salary AS Float | WITH PARAMS Expert AS Integegr |
| VALUES | VALUES | VALUES |
| ('Young', (0,0,0.2,0.3)) | ('Low',(0,0,0.08,0.12) | ('Low',(0,0,0.3,0.4)) |
| ('Middle', | ('Medium', | ('Medium', |
| (0.25,0.30,0.45,0.55)) | (0.1,0.15,0.3,0.35)) | (0.25,0.35,0.7,0.8)) |
| ('Old', (0.5,0.6,1,1)); | ('High',(0.32,0.4,1,1)); | ('High',(0.75,0.85,1,1)); |

Fig. 1. SoftSQL Definitions of Term Sets used in the Example

where $lv \in$ *TermSet*$(A_i)$ and $\mu \in [0,1]$ is the membership degree to the fuzzy valued relation. *TermSet*$(A_i)$ is the term set of linguistic values or a finite set of categorical values. For a tuple $t \in r_F$, $t[A_i].lv$ denotes the actual linguistic value of attribute $A_i$, and $t[A_i].\mu$ denotes the actual membership degree. □

In order to mine fuzzy associative summaries from classical relational tables, it is first necessary to discretize the attribute values, by associating a single linguistic value drawn from the attribute term set. In our implementation (Section VI), we select the label $lv \in$*TermSet*$(A_i)$ that has the greatest membership degree $t[A_i].\mu$.

In order to generate fuzzy associative summaries, we need o transform each classical relation with categorical and numerical attributes into a fuzzy valued relation. For each attribute in the source relation, the user must define a (term) set of *linguistic values*. Thus, the *Fuzzy-Valued Relations* is obtained from a traditional relation by means of *Soft-SQL* queries in which linguistic values are evaluated (as illustrated in Section VI).

**Definition 2**: **Fuzzy Support** Consider an attribute $A$ and a set of linguistic values $V_A = \{lv_1, \ldots, lv_n\}$. Consider now a tuple $t$, where for attribute $A$ there is a given linguistic value, i.e., $t[A].lv \in V_A$, with a given membership degree $t[A].\mu$. Now, let $S$ be a set of fuzzy-valued tuples and let $S(A, lv_i)$ be the subset of $S$ of tuples such that $t[A].lv = lv_i$. The *fuzzy support* of the linguistic value $lv_i$ for attribute $A$ w.r.t. $S$ is defined as [7]:

$$FSup(S, A, lv_i) = \frac{\sum_{t \in S(A, lv_i)} t[A].\mu}{\sum_{t \in S} t[A].\mu}$$

We can generalize the concept to a set of attribute-value pairs. Consider a set $V = \{p_i = (A_i, v_i)\}$ such that there are no two distinct pairs $p_i, p_j \in V$ concerning the same attribute (i.e., $A_i \neq A_j$ for each $i \neq j$). The *Generalized Fuzzy Support* is defined as follows:

$$GFSup(S, V) = \frac{\sum_{t \in \overline{S}} min_{A_i \in V}(t[A].\mu)}{\sum_{t \in S} min_{A_i \in V}(t[A].\mu)}$$

where $\overline{S} \subseteq S$ is the set of tuples such that $\forall p_i = (A_i, v_i) \in V$, it is $t[A_i].lv = v_i$. □

If $S = r_F$, the fuzzy support is called *Full Support*, since it refers to all the tuples in the relation.

**Definition 3**: **Minimum Attribute Relevance Threshold (MART)** Consider a set $S$ of tuples, an attribute $A$, the set $V_A = \{lv_1, \ldots, lv_k\}$ of its linguistic values and, for each $lv_i$, its *full support* *FSup*$(r_F, A, , lv_i)$. We denote with *avg*$(A)$ and *var*$(A)$, resp., the average and the variance of full support values among all linguistic values for attribute $A$, defined as follows (where $r_F(A, lv)$ is the subset of tuples in $r_F$ such that $t[A].lv = lv$).

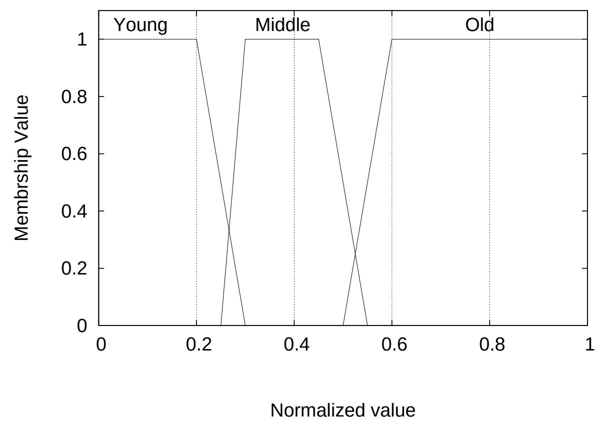


Fig. 2. Trapezoidal functions for term set Ages.

$$avg(A) = \frac{\sum_{lv \in V_A} FSup(r_F, A, , lv) \times |r_F(A, lv)|}{|r_F|}$$

$$var(A) = \frac{\sum_{lv \in V_A} (FSup(r_F, A, , lv) - avg(A))^2 \times r_F(A, lv)}{|r_F|}$$

The *Minimum Attribute Relevance Threshold* for a linguistic value of attribute $A$, *MART*$(A)$, is defined as:

$MART(A) = avg(A) - (\sqrt{var(A)}/2)$ □

The *Minimum Attribute Relevance Threshold* is an adaptive minimum relevance threshold, that is used to select linguistic values that might generate potentially relevant summaries.

### *B. Main Definitions*

**Definition 4**: **Fuzzy Associative Summaries.** Consider a Fuzzy-Valued Relation (table) $R_F(A_1, \ldots, A_n)$ and its instance $r_F$.
A *Fuzzy Associative Summary ASum* is a tuple.

*ASum*$=$ $(AVSeq : \langle p_1, \ldots p_k \rangle, Supp, Strength, Conf)$

where *AVSeq* is a non empty sequence of attribute-value pairs $p_i : (A_i, v_i)$, with $A_i \in R_F$ and $v_i \in$ *TermSet*$(A_i)$.
*Supp*$\in (0,1]$, *Strength*$\in (0,1]$ and *Conf*$\in (0,1]$ are, respectively, the *support*, the *strength* and the *confidence* of the summary w.r.t. relation instance $r_F$, as formalized in Definition 5. □

In the following, we use the notation $n$-*Summary* to denote a summary such that its length $n = |AVSeq|$.

**Definition 5**: **Relevance of Fuzzy Associative Summaries** Consider a Fuzzy Associative Summary *ASum* and a fuzzy-valued relation $r_F$. *Support*, *strength* and *confidence* are measures of relevance of the summary. In particular, the *support* is the frequency with which the summary holds for tuples in $r_F$; the *strength* is the conditional probability that the summary holds, having found the first attribute-value pair; the confidence is the probability that the summary holds, having found the first $n-1$ pairs. The detailed definitions are the following.

- For a 1-*Summary* $s$, $s.Supp$ is the full fuzzy support of the attribute-value pair in the summary; formally, given $s.AVSeq = \langle (A_1, v_1) \rangle$, it is $s.Supp = FSup(r_F, A_1, v_1)$.
  By definition, $s.Strength = 1$, and $s.Conf = s.Supp$.
- Let us denote, with $s$, a $n$-*Summary* (with $n > 1$) and, with $s'$, the $(n-1)$-*Summary* such that $s'.AVSeq \subset s.AVSeq$; in particular, $(a_n, v_n) = s.AVSeq - s'.AVSeq$ is the $n$-th pair in $s$ that is not present in $s'$.
  Let us denote, with $S'$, the set of tuples in $r_F$ for which $s'$ holds and with $S \subseteq S'$ the set of tuples in $r_F$ for which $s$ holds.
  The support of the $n$-*Summary* $s$ is defined based on *GFSup* (Definition 2):
  $$s.Supp = GFSup(r_F, ASeq) = = \frac{\sum_{t \in S} min_{A \in ASeq}(t[A].\mu)}{\sum_{t \in r_F} min_{A \in ASeq}(t[A].\mu)}$$
  The strength of the $n$-*Summary* $s$ is defined based on *FSup* (see Definition 2):
  $$s.Strength = Tnorm(s'.Strength, FSup(S', A_n, v_n)) = = s'.Strength \times FSup(S', A_n, v_n)$$
  The confidence of the $n$-*Summary* $s$ is
  $$s.Conf = FSup(S', A_n, v_n))$$

□

Notice that the definitions of support and confidence are the usual fuzzy extensions of the crisp definitions [7].

### *C. Problem Statement*

We are now ready to state the problem.

"Compute the set $Sum(r_F)$ of *Fuzzy Associative Summaries* involving the attribute-value pairs $(A_i, v_i)$ whose fuzzy support is greater than or equal to the *Minimum Attribute Relevance Threshold* $MART(A_i)$".

### *D. Discussion*

We defined three different measures to assess the validity of summaries, because they highlight different properties of summaries. The *support* measures the relevance of the summary over the entire table: it is the fuzzy frequency with which the summary holds for tuples in the table.
The *confidence* is the conditional probability that a $n$-*Summary* holds, provided that the $(n-1)$-*Summary* from which it derives holds. Following this idea, the confidence of a 1-*Summary* coincides with the support, since we assume that a 1-*Summary* holds, provided that the empty summary holds for the entire table.

| **Name** | **Age** | **Salary** | **Expertise** | **Department** |
|---|---|---|---|---|
| John | 25 | 1000 | 5 | A |
| Kate | 49 | 1150 | 6 | B |
| Susan | 30 | 1100 | 6 | A |
| Mary | 35 | 1200 | 8 | A |
| Marc | 55 | 3800 | 8 | B |

Fig. 3. Table `Employees`.

While support and confidence are the typical validity measures that characterize fuzzy association rules, we defined a new measure called *Strength*. Formally, it is the conditional probability that a summary holds, provided that the first attribute-value pair holds. For a 1-*Summary*, the strength is set to 1, since the summary holds exactly for the same tuples for which the first (and unique) pair holds.

In other words, this is the strength of the associative chain w.r.t. the first attribute-value pair. A summary step-by-step narrows the focus with which it analyzes/describes tuples in the table (we can call it *telescope view* or *telescope analysis*): if the strength is close to 1, this means that the entire summary holds for almost all the tuples for which the first attribute-value pair holds; this is a case of strong associative relationships among the data.

## V. Mining Fuzzy Associative Summaries

The process of mining Fuzzy Associative Summaries can be performed by an algorithm designed for this purpose. Figure 4 reports the algorithm we devised to demonstrate our approach. It is a recursive algorithm: function *GenSummaries* is the recursive component; procedure *MineAssociativeSummaries* is the main part of the algorithm.

In the following, we present the algorithm in details.

- *Preliminary Task.* Line *8.* in procedure *MineAssociativeSummaries* gets the source fuzzy-valued relation from the database.
  Then, Line *9.* computes the set *RelPairs* of relevant attribute-value pairs $(A, v)$ (such that fuzzy support of value $v$ is greater than or equal to the *Minimum Attribute Relevance Threshold* for attribute $A$).
  Line *10.* initializes the set *Sums*, the set of generated associative summaries, to the empty set.
- *Generation of 1-Summaries* The **for each** loop at Line *11.* generates a 1-*Summary* $\overline{s}$ for each relevant attribute-value pair in *RelPairs*.
  Line *12.* actually generates the 1-*Summary* $\overline{s}$, using the fuzzy support of the pair w.r.t. the source fuzzy valued table $r_F$ as support and confidence value for $\overline{s}$ (the strength is 1 for 1-*Summaries*).
  Then, for each generated 1-*Summary* $\overline{s}$, Line *13.* generates $r_{\overline{s}}$, the set of tuples in $r_F$ such that attribute $A$ has value $v$ (i.e., the pair chosen to generate the summary $\overline{s}$).
  Line *14.* calls function *GenSummaries* to obtain the set of

**function** *GenSummaries*(*RelPairs*, *Sums*, $s$, $r_s$, $r_F$)
**begin**
1. *Previous* := *Attrs*($s$);
2. **for each** $p(A, v) \in$ (*RelPairs* $-$ *Previous*) **do**
   **begin**
3. $\overline{s}$ := ($s$.*AVSeq* $\bullet \langle p \rangle$, *GFSup*($r_F, \overline{s}$.*ASeq*),
   $s$.*Strength* $\times$ *FSup*($r_s, A, v$), *FSup*($r_s, A, v$));
4. **if** $\overline{s}$.*Conf* $> 0$ **then**
5. $r_{\overline{s}}$ := $\{t | t \in r_s \wedge t[A] = v\}$;
6. *Sums* := *GenSummaries*(*Sums* $\cup \{\overline{s}\}$, $\overline{s}$, $r_{\overline{s}}$, $r_F$);
   **end if**
   **end**
7. **return** *Sums*;
**end**

**procedure** *MineAssociativeSummaries*
**begin**
8. Get $r_F$, the fuzzy valued relation
9. *RelPairs* := $\{(A, v) \mid A \in R \wedge v \in$ *Dom*($A$) $\wedge$
   *FSup*($r_F, A, v$) $\geq$ *MART*($A$)$\}$;
10. *Sums* := $\emptyset$;
11. **for each** $p(A, v) \in$ *RelPairs* **do**
    **begin**
12. $\overline{s}$ := ($\langle p \rangle$, *FSup*($r_F, A, v$), 1, *FSup*($r_F, A, v$));
13. $r_{\overline{s}}$ := $\{t | t \in r_F \wedge t[A] = v\}$;
14. *Sums* := *GenSummaries*(*Sums* $\cup \{\overline{s}\}$, $\overline{s}$, $r_{\overline{s}}$, $r_F$);
    **end**
15. *Summaries* := *Sums*;
**end**

Fig. 4. Algorithm for Mining Associative Summaries.

all summaries that extend the 1-*Summary* $\overline{s}$; notice that $r_{\overline{s}}$ and $r_F$ are passed.

- *The Recursive Process.* For each 1-*Summary* generated in the preliminary task, the recursive process is performed by the recursive function *GenSummaries*.
  1) Function *GenSummaries* receives the set of relevant attribute-value pairs *RelPairs*, the set *Sums* of so far generated summaries, a $i$-*Summary* $s$, the set of fuzzy-valued tuples $r_s$ for which the $i$-*Summary* $s$ holds and the full fuzzy-valued table $r_F$.
  2) Line *1.* extracts (function *Attrs*) the set of attributes already involved in the $i$-*Summary* $s$; the resulting set is assigned to variable *Previous*.
  3) The **for each** loop at Line *2.* considers all the relevant attribute-value pairs $p(A, v)$ for attributes not yet involved in the $i$-*Summary* $s$. For each pair $p(A, v)$, it tries to generate a new $(i+1)$-*Summary* $\overline{s}$ by adding pair $p$ to $s$.
     In particular, the new $(i+1)$-*Summary* $\overline{s}$ is actually generated at Line *3.*. The sequence of attribute-value pairs is directly derived from $s$.*AVSeq* by appending pair $p$. The support $\overline{s}$.*Supp* is the generalized full fuzzy support of the sequence *ASeq*. The strength value $\overline{s}$.*Strength* is obtained by multiplying the strength value $s$.*Strength* of the $i$-*Summary* $s$ by the fuzzy support of pair $p$ evaluated on the set of tuples for which the $i$-*Summary* $s$ holds. The confidence value $\overline{s}$.*Conf* is the fuzzy support of pair $p$ evaluated on the set of tuples for which the $i$-*Summary* $s$ holds.
     The conditional instruction at Line *4.* checks if the confidence of the newly generated $(i + 1)$--*Summary* is greater than 0 (when the confidence is 0, support and strength are 0 as well): only in this case the summary can be actually kept and used for generating other summaries. So, Line *5.* and Line *6.* are executed only if the relevance of $\overline{s}$ is greater than 0.
     Line *5.* selects the set $r_{\overline{s}}$ of tuples for which the new $(i + 1)$-*Summary* $\overline{s}$ holds: it simply selects tuples in $r_s$ (the set of tuples for which the $i$-*Summary* $s$ holds) for which the pair $p$ holds.
     Line *6.* recursively calls function *GenSummaries*, in order to generate (if possible) all $j$-*Summaries*, with $j > (i + 1)$, derived from the $(i + 1)$-*Summary* $\overline{s}$.
  4) Line *7.* returns the set of summaries so far generated to the upper level of the recursion.

**Example 1:** To illustrate the mining process, consider the sample table `Employees` depicted in Figure 3, that describes employees of a company. If we exclude attribute *Name*, that plays the role of primary key, attributes *Age*, *Salary* and *Expertise* are numerical, while attribute *Department* is categorical.

- First of all, we have to transform table *Employees* into a *Fuzzy-Valued Table*. We make use of the three term sets defined in Figure 1, one for each numerical attribute *Age*, *Salary* and *Expertise*. For an attribute, each linguistic value of the associated term set is evaluated against the actual numerical values of the attribute. The result is the table depicted in Figure 6: notice that in place of each attribute, we reported a triple with membership degrees of the original attribute value w.r.t. the linguistic values reported below the attribute name in the table schema.
- Second, for each tuple, for each attribute the linguistic value with greatest membership degree is taken. We obtain the table depicted in Figure 7, where the value of each attribute is a pair (*Linguistic Value*, *Membership Degree*). This table is the set $r_F$ of tuples used in the successive iterative process.
  The values of the *Fuzzy Support* for each attribute-value pair are reported in Figure 5.

  Based on the Fuzzy Support of attribute-value pairs, we can compute the *Minimum Attribute Relevance Threshold* values *MART*($A$). They are reported in Figure 5.b.
  Based on these results, the following attribute-value pairs will not be considered in the iterative process, because their support is always lower than the corresponding *MART*: (*Age*, *Young*), (*Age*, *Old*), (*Salary*, *Medium*), (*Expertise*, *High*) (notice that (*Expertise*, *Low*) does not

| *Attribute* | *Ling. Value* | *Full Support* |
|---|---|---|
| *Age* | *Young* | 0.5/3.6 = 0.139 |
| *Age* | *Middle* | 2.6/3.6 = 0.722 |
| *Age* | *Old* | 0.5/3.6 = 0.139 |
| *Salary* | *Low* | 0.75/2.4 = 0.341 |
| *Salary* | *Medium* | 0.9/2.4 = 0.318 |
| *Salary* | *High* | 0.75/2.4 = 0.341 |
| *Expertise* | *Medium* | 3.0/4.0 = 0.75 |
| *Expertise* | *High* | 1.0/4.0 = 0.25 |
| *Department* | *A* | 3.0/5.0 = 0.6 |
| *Department* | *B* | 2.0/5.0 = 0.4 |

a)

| *Attribute* | *MART* |
|---|---|
| *Age* | 0.4368 - (0.1076 / 2) = 0.346 |
| *Salary* | 0.36 - (0.032 / 2) = 0.326 |
| *Expertise* | 0.68 - (0.0576 / 2) = 0.428 |
| *Department* | 0.52 - (0.0096 / 2) = 0.471 |

b)

Fig. 5. a) Full Supports of Attribute-Value Pairs. b) MARTS for Attribute-Value Pairs.

appear in any tuple), (*Department*, *B*).

- *1-Summaries.* Based on set $r_F$ and the results of the previous step, 1-*Summaries* can be generated. The first one might be
  $Summary_1$ =
  $(\langle(Expertise, Medium)\rangle, 0.75, 1, 0.75)$
  Notice that support and confidence coincide and are exactly the value of the fuzzy support of the attribute-value pair.
  The set of tuples for which the linguistic value of attribute *Expertise* is *Medium* is computed; we denote it as $S_1$ (tuples in set $S_1$ are marked in Figure 7 with $S_1$).
- *2-Summaries.* Based on the previous summary, and considering set $S_1$, we can generate a new set of 2-*Summaries*.
  Considering the pair (*Salary*, *Low*), we obtain the 2-*Summary*
  $Summary_2$ =
  $(\langle(Expertise, Medium), (Salary, Low)\rangle, 0.384, 0.714, 0.714)$
  Notice that the confidence is $FSup(S_1, Salary, low) = 0.75/1.05 = 0.714$, while the new computed strength is $1 \times 0.714 = 0.714$. The support is $0.75/1.95 = 0.384$
  The set of tuples in $S_1$ that have value *Low* for attribute *Salary* is generated; we denote it as $S_2$ (tuples in set $S_2$ are marked in Figure 7 with $S_2$).
- *3-Summaries.* Based on the previous summary and on set $S_2$, it is possible to generate a new set of 3-*Summaries*. Among the attribute-value pairs for the two remaining attribute, we might chose (*Age*,*Middle*), obtaining the following 3-*Summary*.
  $Summary_3$ =
  $(\langle(Expertise, Medium), (Salary, Low), (Age, Middle)\rangle, 0.1282, 0.47838, 0.67)$
  Notice that the confidence is $FSup(S_2, Age, Middle) = 1.0/1.5 = 0.67$, while the new computed strength is $0.714 \times 0.67 = 0.47838$. The support value of the summary is $0.1282$.
  The set $S_3$ of tuples in $S_2$ for which the linguistic value of attribute *Age* is *Middle* is generated (the tuple in set $S_3$ is marked in Figure 7 as $S_3$).
- *4-Summary.* Finally, only one attribute can be considered for extending $Summary_3$, i.e., attribute *Department* with value $A$. The resulting 4-*Summary* is the following.
  $Summary_4$ =
  $(\langle(Expertise, Medium), (Salary, Low), (Age, Middle), (Department, A)\rangle, 0.1282, 0.47838, 1.0)$
  Notice that the support is the same as the support of $Summary_3$: this is due to the fact that the set of tuples for which the new summary holds is the same, and the membership degrees of attribute *Department* is always 1. The fuzzy support of the attribute-value pair (*Department*, *A*) is $1.0/1.0 = 1.0$, thus the confidence is 1 and the Strength is the same as for $Summary_3$.
  This piece of information is interesting: the fact that the *Expertise* is *Medium*, the *Salary* is *Low* and the *Age* is *Middle* implies that the *Department* is *A*.

□

The mining process generates a large number of summaries, that, for the sake of space, cannot be reported in the paper. We could limit the number of generated summaries by imposing a minimum threshold on the validity measures.

## VI. Implementation on Top of Soft-SQL

In this section, we present how Fuzzy Associative Summaries can be mined on top of *Soft-SQL*, the prototypal engine supporting flexible querying on relational databases we implemented in our past work [1].

The mining algorithm (Figure 4) has been implemented within a Java application, that accesses to the database through the *Soft-SQL Package*. In particular, the application starts from a classical relational table; then, by means of a mapping between numerical attributes and term sets, it derives the corresponding *Fuzzy-Valued Relation*. From this point, the algorithm is implemented in such a way it heavily exploits SQL queries.

The *Soft-SQL* query language, being an extension of classical SQL query language, forces to address the problem from a relational point of view: the result is that *Soft-SQL* can be successfully exploited if we adopt a *vertical* approach to the problem.

*Fuzzy-Valued Table.* The first thing to do is to obtain the fuzzy-valued table $r_F$ from the source table $r$. This requires the evaluation of linguistic values for each attribute.

Let us denote with $K(k_1, \ldots, k_h) \subset R$ the set of attributes in the primary key of table $r$; with $M = (m_1, \ldots, m_q) = R - K$ we denote the set of attributes to mine.

| **Name** | **Age** (Young, Middle, Old) | **Salary** (Low, Medium, High) | **Expertise** (Low, Medium, High) | **Department** |
|---|---|---|---|---|
| John | (0.5,0.0,0.0) | (0.5,0.0,0.0) | (0.0,1.0,0.0) | A |
| Kate | (0.0,0.6,0.0) | (0.125,0.3,0.0) | (0.0,1.0,0.0) | B |
| Susan | (0.0,1.0,0.0) | (0.25,0.2,0.0) | (0.0,1.0,0.0) | A |
| Mary | (0.0,1.0,0.0) | (0.0,0.4,0.0) | (0.0,0.0,0.5) | A |
| Marc | (0.0,0.0,0.5) | (0.0,0.0,0.75) | (0.0,0.0,0.5) | B |

Fig. 6. Step 1 Applied to Table `Employees`.

| **Name** | **Age** (value, degree) | **Salary** (value, degree) | **Expertise** (value, degree) | **Department** | |
|---|---|---|---|---|---|
| John | (Young,0.5) | (Low,0.5) | (Medium,1.0) | A | $S_1$, $S_2$ |
| Kate | (Middle,0.6) | (Medium,0.3) | (Medium,1.0) | B | $S_1$ |
| Susan | (Middle,1.0) | (Low,0.25) | (Medium,1.0) | A | $S_1$, $S_2$, $S_3$, $S_4$ |
| Mary | (Middle,1.0) | (Medium,0.4) | (High,0.5) | A | |
| Marc | (Old,0.5) | (High,0.75) | (High,0.5) | B | |

Fig. 7. The Fuzzy-Valued Table $r_F$ obtained from table `Employees`. On the right, if a tuple is marked with $S_i$, means that it belongs to the subset $S_i$ of the running example.

With $Q_m$ we denote the *Soft-SQL* query that, for attribute $m \in M$, generates, for each (possibly linguistic) value $v$ of $m$, the membership degree of $v$ for each tuple in $r$. Depending on the domain of $m$, $Q_m$ is defined in two different ways.

1) If *Dom*$(m)$ is categorical, $Q_m$ is as follows:

```
SELECT k1, ..., kh, 'm' AS ANAME,
   m AS AVALUE, 1.0 AS ADEG
FROM r;
```

2) If *Dom*$(m)$ is numerical, there is a term set *TS* associated with $m$, in which $l$ linguistic values $lv_1, \dots, lv_l$ are defined. For each linguistic value $lv_i$, we define a *Soft-SQL* query $Q^i_m$ based on the `IS... IN` predicate.

```
SELECT k1, ..., kh, 'm' AS ANAME,
   'lvi' AS AVALUE, DEGREE AS ADEG
FROM r
WHERE m IS 'lvi' IN TS;
```

To obtain query $Q_m$, queries $Q^i_m$ are concatenated by means of the `UNION ALL` SQL operator.

$Q_m = Q^1_m$ `UNION ALL ... UNION ALL` $Q^l_m$;

All $Q_m$ queries are concatenated, again based o the `UNION ALL` SQL operator, and the result is inserted into table `FVALUES`.

```
INSERT INTO FVALUES
  (Qm1
   UNION ALL
   ...
   UNION ALL
   Qmq);
```

Table `FVALUES` gives the membership degree of each (possibly linguistic) attribute value for each tuple $t$ in $r$, where each tuple $t$ is identified by the value of the primary key.

To complete the computation of the fuzzy-valued table, it is necessary to select, for each tuple $t$ in $r$, the (linguistic) value with the highest membership degree. This is done by the following query, that, based on the previously computed `FVALUES` table, generates the new table `FVT` (for Fuzzy-Valued Table).

```
INSERT INTO FVT
SELECT FVALUES.k1, ..., FVALUES.kh,
    FVALUES.ANAME, FVALUES.AVALUE,
     FVALUES.ADEG
FROM FVALUES INNER JOIN
   (SELECT k1, ..., kh, ANAME,
      MAX(ADEG) AS MDEG
     FROM FVALUES
     GROUP BY k1, ..., kh, ANAME) AS MFV
    ON FVALUES.k1=MFV.k1 AND ...
       AND FVALUES.kh=MFV.kh
       AND FVALUES.ANAME=MFV.ANAME
       AND FVALUES.ADEG=MFV.MDEG
```

*Fuzzy Support and MART.* Table `FVT` is the starting point for computing the fuzzy support of each attribute-value pair. Thus, by means of a query, table `FSUPPORTS` is computed: its schema is (`ANAME`, `AVALUE`, `FSUP`), so that each tuple describes, for an attribute-value pair (`ANAME, AVALUE`), the full fuzzy support `FSUP`.

Table `FSUPPORTS` can be used to evaluate the *Minimum Attribute Relevance Threshold* (*MART*) of each attribute, and insert the results into table `MARTS`. A few queries are necessary to perform this task: the result is inserted into table `MARTS` (whose schema is similar to the table depicted in Figure 5.b), This table is used to delete, from table `FSUPPORTS`, the tuples with value of attribute `FSUP` less than the corresponding *MART*.

*Computation of Sets $r_{\overline{s}}$.* Consider procedure *MineAssociativeSummaries* in the mining algorithm (Figure 4). After the new 1-*Summary* $\overline{s}$ is generated at Line *12.*, at Line *13.* the set $r_{\overline{s}}$ containing the tuples for which $\overline{s}$ holds is generated, by selecting these tuples from table $r_F$ (the fuzzy-valued table).

In our implementation, $r_F$ is represented in a vertical way by table FVT. So, we decided to represent $r_{\overline{s}}$ in table $\texttt{FVT}_{\overline{s}}$, by means of the primary key of $r_F$. Recall that in the 1-*Summary*, *AVSeq*= $\langle (A, v) \rangle$.

```
INSERT INTO FVT_s̄
(SELECT FVT.k_1 ..., FVT.k_h
 FROM FVT
 WHERE ANAME=A AND AVALUE=v)
```

In the recursive function *GenSummaries* of the mining algorithm, set $r_s$ is received as input parameter. At Line *5.*, the set $r_{\overline{s}}$ is computed, based on $r_s$ and using the chosen pair $(A, v)$.

In the previous query, we saw that $r_s$ is represented by table $\texttt{FVT}_{\overline{s}}$, by means of the primary key values for which summary $s$ holds. In the following query, we show how to compute and represent $r_{\overline{s}}$ at Line *5.* of the algorithm.

```
INSERT INTO FVT_s̄
(SELECT FVT.k_1 ..., FVT.k_h
 FROM FVT INNER JOIN FVT_s
 ON FVT.k_1=FVT_s.k_1 AND ... AND FVT.k_h=FVT_s.k_h
 WHERE ANAME=A AND AVALUE=v)
```

*Computation of Fuzzy Supports.* At Line *3.* of the mining algorithm (Figure 4), the support, strength and confidence of the newly generated summary $\overline{s}$ are computed by evaluating *GFSup*$(r_F$, *ASeq*$)$ (generalized full support) and *FSup*$(r_s, A, v)$, (the fuzzy support of tuples in $r_s$ for which $t[A] = v$).

The vertical representation of tuples in $r_F$ we adopted in our implementation allows to easily compute these values, by means of simple aggregation-based SQL queries.

## VII. Concluding Remarks

We conclude the paper with some remarks about our proposals and our plans for future work on this topic.

First of all, we point out that, for the example shown in the paper, 65 fuzzy associative summaries are generated. Their number is not negligible, and therefore the user cannot easily comprehend them. A simple approach could be to define a minimum threshold for the validity measures of generated summaries to filter out only the meaningful ones. However, all summaries, together, give a good description of associative relations within the database. In particular, one interesting thing to do is to analyze if the strength of summaries decreases or remains stable when a summary is derived from another one: this could let interesting phenomena emerge, possibly unexpected to the user. When the strength remains stable in the chain, it means that there is a strong association between the fuzzy attribute values. Therefore, we want to investigate this topic, in order to define both a visualization technique, and an automatic analysis technique, to highlight significant behaviours that emerge by comparing summaries.

Finally, we observe that the proposed algorithm for mining Fuzzy Associative Summaries is not the optimal one in terms of efficiency. In fact, some preliminary experiments performed on the *Forest CoverType* dataset [18] has shown that the algorithm scales well w.r.t. the number of tuples in the mined table, while after 5 mined attributes performance dramatically gets worse, due to the exponential growth of possible combinations of attribute-value pairs. Therefore, we plan to develop an efficient algorithm that performs a limited and fixed number of scans over the mined table.

## References


[1] G. Bordogna and G. Psaila, "Customizable flexible querying for classical relational databases," *J. Galindo (ed.) Handbook of Research on Fuzzy Information Processing in Databases*, 2008.

[2] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "From data mining to knowledge discovery in databases," *AI Magazine*, vol. 17, pp. 37–54, 1996.

[3] R. Agrawal, T. Imielinski, and A. Swami, "Mining association rules between sets of items in large databases," in *Proceedings of SIGMOD-93, the ACM SIGMOD Int. Conference on Management of Data*. Washington, D.C.: British Columbia, 1993, pp. 207–216.

[4] S. Brin, R. Motwani, J. Ullman, and S. Tsur, "Dynamic itemset counting and implication rules for market basket data," *SIGMOD Record*, vol. 26 (2), pp. 255–264, 1997.

[5] R.Srikant and R.Agrawal, "Mining quantitative association rules in large relational tables," in *Proceedings of the ACM-SIGMOD International Conference on the Management of Data*, San Jose, California, May 1996.

[6] M. Delgado, N. Marn, D. Snchez, and M. Vila, "Fuzzy association rules: General model and applications," *IEEE Trascations on Fuzzy Systems*, vol. 11 (2), pp. 214–225, April 2003.

[7] C. Kuok, A.-C. Fu, and M.H.Wong, "Mining fuzzy association rules in databases," *SIGMOD Record*, vol. 27 (1), pp. 41–46, 1998.

[8] H. Lee and H. Kwang, "An extension of association rules using fuzzy sets," in *Proceedings of IFSA-97 World Congress, Prague, Czech Republic*, 1997.

[9] W. Au and K. Chan, "Mining fuzzy association rules," in *Proceedings of 6th Int. Conf. Information Knowledge Management, Las Vegas, NV, USA*, 1997, pp. 209–215.

[10] C. Chen, Q. Wei, and E. Kerre, "Fuzzy data mining: Discovery of fuzzy generalized association rules," in *in G. Bordogna, G. Pasi (eds.), Recent Issues on Fuzzy Databases*. Physica Verlag, 2000, pp. 45–66.

[11] R. Y. D. Rasmussen, "Summarysql - a fuzzy tool for data mining," *Intelligent Data Analysis*, vol. 1 (1-4), pp. 49–58, 1997.

[12] R. Agrawal, T. Imielinski, and A. Swami, "Database mining: A performance perspective," *IEEE Transactions on Knowledge and Data Engineering*, vol. 5, no. 6, pp. 914–925, December 1993.

[13] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in *Proceedings of the 20th VLDB Conference*, Santiago, Chile, September 1994.

[14] M. A. W. Houtsma and A. Swami, "Set-oriented mining for association rules in relational databases," in *11th International Conference on Data Engineering*, Taipei, Taiwan, March 6-10 1995.

[15] R. Meo, G. Psaila, and S. Ceri, "A new SQL-like operator for mining association rules," in *Proceedings of the 22st VLDB Conference*, Bombay, India, September 1996.

[16] T. Hong and Y. Lee, "An overview of mining fuzzy association rules," *H. Bustince, F. Herrera and J. Montero (eds.), Fuzzy Sets and Their Extensions: Representation, Aggregation and Models*, vol. Studies in Fuzziness and Soft Computing, Volume 220, pp. 397–410, 2008.

[17] T. Hong, C. Kuo, and S. Chi, "Mining association rules from quantitative dat," *Intelligent Data Analysis*, vol. 2(5), 1999.

[18] "Forest covertype datase," http://kdd.ics.uci.edu/databases/covertype/covertype.data.html