Finding Associations in Composite Data Sets: The CFARM Algorithm

M. Sulaiman Khan[1], Maybin Muyeba[2], Frans Coenen[1], David Reid[3], Hissam Tawfik[3]

[1] University of Liverpool, Department of Computer Science
[2] School of Computing, Mathematics & Digital Technology, Manchester Metropolitan University, UK
[3] Liverpool Hope University, School of Computing, Liverpool, UK

{mskhan@liv.ac.uk, m.muyeba@mmu.ac.uk, frans@csc.liv.ac.uk, reidd@hope.ac.uk, tawfikh@hope.ac.uk }

Abstract. A fuzzy association rule mining mechanism (CFARM), directed at identifying patterns in datasets comprised of composite attributes, is described. Composite attributes are defined as attributes that can take simultaneously two or more values that subscribe to a common schema. The objective is to generate fuzzy association rules using “properties” associated with these composite attributes. The exemplar application is the analysis of the nutrients contained in items found in grocery data sets. The paper commences with a review of the back ground and related work, and a formal definition of the CFARM concepts. The CFARM algorithm is then fully described and evaluated using both real and synthetic data sets.

Keywords: Association rules, fuzzy association rules, composite attributes, quantitative attributes.

1 Introduction

Data mining is an important well established research area and Association Rule Mining (ARM) is a very popular topic in the data mining community. The objective of ARM is to identify patterns, expressed as Association Rules (ARs), usually from binary-valued transaction data sets (Fayyad et al. 1996), (Ferenc, B. 2003), (Coenen et al. 2004), (Agrawal et al. 1993). Work has been done on a variety of extensions of the standard (binary-valued) approach to ARM thus allowing for its applicability to quantitative and categorical (non-binary) data (Gyenesei, A. 2001), (Dong and Tjortjis, 2003), (Srikant and Agrawal, 1996), (Au and Chan, 1999). To deal with quantitative data, values are divided into ranges such that each range represents a binary valued attribute and then labelling the identified range attributes; for example “low”, “medium”, “high” etc. There are two possible ways for assigning ranges: using crisp boundaries or fuzzy boundaries. Fuzzy ARM uses the latter to identify fuzzy ARs. Some earlier works show that more expressive ARs can be obtained using fuzzy ARM than “crisp” methods (Gyenesei, A. 2001), (Kuok et al. 1998), (Dubois et al. 2006), (Khan et al. 2006). ARM (both fuzzy and standard) algorithms typically use the support-confidence framework to identify “interesting” ARs during the rule generation process. This framework, however, has a number of disadvantages, for example, generating a vast AR set many of which are either obvious, subsumed by other rules or largely redundant. Consequently there are motivations in the data mining community for finding more expressive, succinct or significant and useful ARs. Some earlier work (Kuok et al. 1998), (Khan et al. 2006) demonstrate this using the certainty measure, which is of note in the context of the work described here.

In this paper we introduce a particular category of a fuzzy ARM application called Composite item Fuzzy ARM (CFARM). CFARM's objective is to generate fuzzy ARs from "properties" associated with composite attributes (Kim et al. 1989) i.e. attributes or items composed of sets of sub-attributes or sub-items that have a common schema. Image mining is a typical example where different areas of an image has groups of pixels such that each group can be represented by the normalized summation of the RGB values of the pixels in that group. In this case the set of composite attributes (I) is the set of groups, and the set of properties (P) shared by the groups is equivalent to the RGB summation values (i.e. $P = \{R, G, B\}$). We can then express fuzzy sets such as "light", "medium" and "dark" and find associations between such composite attribute attributes with their properties. Considering the familiar market basket scenario, we can have define I as a set of groceries and P as a set of nutritional properties that these groceries may possess, for example protein, iron, calcium and copper (i.e. P = {Pr, Fe, Ca, Cu...}). Of note is the difference in these two examples. In the shopping basket, I is constant i.e. it only represents a categorical list of common properties. In the image mining example, I is a normalized summation of properties.

Further, a stock control database can have I as a collection of stock items where P a collection of stock item properties is common to all items, including for example cost price, sale price, reorder time etc. Given that we have quantitative attributes that can be partitioned into intervals or ranges, we rename such partitions with linguistic values or in this case, introduce fuzzy sets for these attributes. We are motivated by the fact that the approach described in this paper is a new way of dealing with so-called composite attributes that may potentially have fuzzy features.

The main contributions of the paper are:

1. The concept of CFARM.
2. The potential of ARs from itemset properties.
3. A practical example of the use of CFARM.
4. Employment of certainty factor, a quality measure to produce strong rules.
5. New Fuzzy Apriori-T algorithm for better efficiency.

We also demonstrate that a more succinct set of property ARs (than that generated using a non-fuzzy method) can be produced using the proposed approach.

The paper is organised as follows. In section 2 we present the background and related work to the proposed composite fuzzy ARM approach described. Section 3 presents a sequence of terms and concepts for the work and section 4 introduces the CFARM algorithm. The motivation for the work is expanded upon in Section 5 where an example application is described. A complete analysis of the operation of the CFARM algorithm is given in Section 6, and section 7 concludes the paper with a summary of the contribution of the work and directions for future work.

2 Background and Related Work

The most familiar ARM approach is to first generate all the itemsets (attribute sets) and then derive sets of ARs (Agrawal et al. 1993). A frequent itemset is defined as one that appears most often in the given data set. To determine "frequency" of an item, there is a user supplied support threshold measure that checks item frequencies. Similarly, a confidence threshold is a conditional probability measure of the strength of ARs generated. The user must select support and confidence thresholds to influence the number and strength of ARs. To ensure that itemsets with low support but from which high confidence rules may be generated depends on careful selection of support and confidence.

Some drawbacks on using only the support and confidence framework to assess association rules have been reported (Berzal et al. 2002), (Silverstein et al. 1998), (Sanchez, D. 1999). To avoid some of these and to ensure interesting discovered rules, the certainty factor and the new concept a very strong rule was proposed in (Berzal et al. 2002), (Sanchez, D. 1999). Implementations can be found in (Gyenesei, A. 2001), (Kuok et al. 1998), (Khan et al. 2006), (Khan et al. 2008).

From database literature, the term composite item has been used previously in the context of data mining. Authors in (Wang et al. 2006), (Ye, X. and Keane, J. 1997) define a composite item as combining several items e.g. if itemset {A, B} and {A, C} are not frequent then rules {B}→{A} and {C}→{A} will not be generated, but by combining B and C to make a new *composite* item {BC} which may be frequent, rules such as {BC}→{A} may be generated. The difference with the approach in this paper is that we define a composite item to be a structured attribute as indicated in the introduction to this paper and explained further in Section 3. The definition concurs with database literature (Kim et al. 1989), (Kim et al. 1987), the earliest references to composite attributes that the authors are aware of, which also defines composite attributes (items) in this manner, i.e. as attributes that comprise two or more sub-attributes. The difference with fuzzy ARM algorithms is their non-use of composite items.

ARM typically operates using binary valued attributes. Given quantitative attributes, these can be discretised into a number of interval partitions where each partition is regarded as a binary valued attribute. A major problem in discretising quantitative attributes using interval partitions (Gyenesei, A. 2001), (Khan et al. 2006), (Kuok et al. 1998) is the "sharp boundary problem". Fuzzy ARM (Kuok et al. 1998), (Gyenesei, A. 2001), (Chen, G. and Wei, Q. 2002), (Au and Chan, 1999) is one approach to addressing this problem. Fuzzy ARM is used to discover frequent itemsets using fuzzy sets (with overlapping partitions) in order to handle the quantitative attributes. Fuzzy approaches deal with quantitative attributes by mapping numeric values to membership degrees from their partitions. The mapping is undertaken in such a way that individual item contributions to support counts remain at unity regardless of whether an item value belongs to one or more fuzzy sets (a similar approach was used in (Gyenesei, A. 2001)). The main benefit of using fuzzy ARs is that fuzzy sets can soften the effect of sharp boundaries and make the rules more understandable to the user. Detailed overviews for fuzzy association rules are given in (Gyenesei, A. 2001), (Kuok et al. 1998), (Delgado et al. 2003), (Au and Chan, 1999).

More generally, fuzzy data mining algorithms have been utilized in many application domains, example include (i) fuzzy ARs for classifier in capturing correlations between genes (Mohammad et al. 2008), (ii) parallel fuzzy c-Means clustering for large data sets (Terence et al. 2002) and (iii) Acquisition of fuzzy association rules from medical data (Delgado et al. 2001, 2002).

To the best of our knowledge there seems to be no work on composite association rule mining using fuzzy approaches.

To illustrate the work described here we consider super market basket analysis where the set of groceries (I) have a shared set of nutritional quantitative properties (P). Some examples are given in Table 1. The objective is then to identify patterns linking the properties (nutrients in the case of the example).

Items/Nutrients	**Protein**	**Fibre**	**Carbohydrate**	**Fat**	**…**
Yogurt	2.9	0.0	5.1	0.3	…
Pitta Bread	9.5	3.6	39.9	1.6	…
Wafers	7.1	4.9	58.3	24.9	…
…	…	…	…	…	…

Table 1: Example composite attributes (groceries) with their associated properties (nutrients)

Table 1 shows some items from market basket data that we can extract nutrients, presumably "edible items", all with common properties or nutrients. The context of our problem is illustrated using figure 1. The figure shows edible composite items with common properties such as Protein, Fibre, Iron etc. defined by the same five fuzzy sets {Very Low, Low, Ideal, High, Very High}.

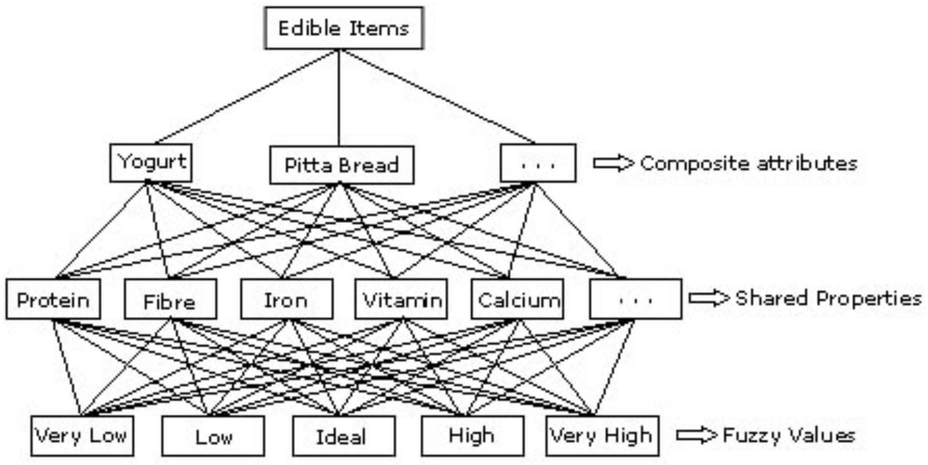

Fig 1: Edible items, Nutrients & Fuzzy Intervals

In figure 1, a composite item such as "yogurt" can have properties of nutrients (protein, iron, calcium...) measured quantitatively as such and therefore can be expressed as fuzzy sets in the usual way. The amount of nutrients in an item can be measured by the degree of membership, for example membership of "fibre" in "yogurt" is zero. We define the problem of addressing composite item fuzzy association rules in the next section.

3 Problem Definition

In this section a sequence of terms and concepts is presented to: (i) define the term composite attributes, (ii) describe the concept of fuzzy association rule mining and (iii) the fuzzy approach adopted by the authors. The normalization process for Fuzzy Transactions (*FT)* and rule interestingness measures will also be discussed later in this section. In this section, we look data, and fuzzy specific concepts, and finally quality measures for fuzzy association rules.

3.1 Data Specific Concepts

Raw Data: A Raw Dataset (the input data) D consists of a set of transactions $T = \{t_1, t_2, t_3, \cdots, t_n\}$, a set of composite items $I = \{i_1, i_2, i_3, \cdots, i_{|I|}\}$ and a set of properties $P = \{p_1, p_2, p_3, \cdots, p_m\}$. Each transaction t_i (the "i^{th}" transaction) is some subset of I, and each item $t_i[i_j]$ (the "j^{th}" item in the "i^{th}" transaction) is a subset of P. Thus each item i_j will have associated with it a set of values corresponding to the set P, i.e. $t_i[i_j] = \{v \mid v_1, v_2, v_3, \cdots, v_m\}$. The "$k^{th}$" property value for the "j^{th}" item in the "i^{th}" transaction is given by $t_i[i_j[v_k]]$. Note that a property attribute can take either a categorical or a quantitative value. An example is given in Table 2 where each composite item is represented using the notation <label,value>; thus record 1 comprises two items a and b which have (numeric) values {2,4,6} and {4,5,3} respectively. Note that in this example each distinct item has a common set of property values (in practice this is usually the case).

TID	Record
1	{<a,{1,5,4}>, <b,{3,7,2}>}
2	{<c,{6,2,4}>, <d,{2,5,1}>}
3	{<a,{1,5,4}>, <c,{6,2,4}>, <d,{2,5,1}>}
4	{<b,{3,7,2}>, <d,{2,5,1}>}

$D = \{T_1, T_2, T_3, T_4\}$
$I = \{a, b, c, d)$
$P = \{x, y, z\}$

Table 2: Example raw dataset D

In the rest of this paper the term item is used to mean an *item* in an itemset in the manner associated with traditional ARM, and the term *attribute* is used to mean a property item (sub-item).

Property Dataset: In the process described here the given raw dataset D is initially transformed into a property data set D^p. A property dataset D^p consists of property transactions $T^p = \{t_1^p, t_2^p, t_3^p, ... t_n^p\}$ and a set of property attributes P (instead of a set of composite items I). Each transaction t_i^p (the "i^{th}" transaction) is some subset of $P = \{p_1, p_2, p_3, \cdots, p_m\}$. The value for each property attribute $t_i^p[p_j]$ (the "j^{th}" property attribute in the "i^{th}" property transaction) has a numeric value obtained by aggregating the numeric values for all p_j in t_i. Thus:

$$(t_i^p[p_j]) = \frac{\sum_{j=1}^{|t_i|} t_i[i_j[v_k]]}{|t_i|} \quad (1)$$

An example is given in Table 3.

TID	X	Y	Z
1	2.0	6.0	3.0
2	4.0	3.5	2.5
3	3.0	4.0	3.0
4	2.5	6.0	1.5

Table 3: Example property data set D^p generated from raw data set given in table 2

In table 3 the values are calculated by first aggregating and then averaging the items' property values using table 2 e.g. in table 3 row 1 the value for property X is calculates by first aggregating the property values from table 2 row 1 using item "a" and "b" as 1.0+3.0=4.0 and later averaging them 4.0/2.0=2.0. Same for the property value Y as 5.0+7.0=12.0, then 12.0/2.0=6.0.

Fuzzy Dataset: With respect to the work described here, once a property data set D^p has been established this is further transformed into a fuzzy dataset D'. A fuzzy dataset D' consists of fuzzy transactions $T' = \{t'_1, t'_2, t'_3, ..., t'_n\}$ and a set of fuzzy property attributes P' each of which in turn has a number of fiuzzy sets associated with it identified by a set of linguistic *labels* $L = \{l_1, l_2, l_3, ..., l_{|L|}\}$ (for example $L = \{small, medium, l\arg e\}$). The fuzzy sets describe a sequence of overlapping user defined ranges into which all possible values for property attributes may be mapped. Each property attribute $t_i^p[p_j]$ is associated (to some degree) with several fuzzy sets. The degree of association is given by a *membership degree* value, in the range $[0,1]$, which indicates the correspondence between the value of a given $t_i^p[p_j]$ and the set of *fuzzy linguistic labels*. The "kth" label for the "jth" property attribute for the "ith" fuzzy transaction is given by $t'_i[p_j[l_k]]$. The nature of the user defined fuzzy ranges is expressed in a *properties table* (see definition 6 below). The numeric values for each property attribute $t_i^p[p_j]$ are *fuzzified* (mapped) into the appropriate membership degree values using a membership function $\mu(t_i^p[p_j], l_k)$ that applies the value of $t_i^p[p_j]$ to the definition of a specified label $l_k \in L$,thus

$$t'_i[p_j] = \{\mu(t_i^p[p_j]], l_1), \mu(t_i^p[p_j]], l_2), \mu(t_i^p[p_j]], l_3), \cdots, \mu(t_i^p[p_j]], l_{|L|})\}$$

The nature of the function is discussed in more detail in sub-section 3.2 below. The complete set of fuzzy property attributes P' is then given by $P \times L$.

An example fuzzy data set is given in Table 4 based on the property data set given in Table 3. Note that the membership values have all been normalised so that the contribution to the support count for a single attribute in a single record remains in [0,1].

TID	X			Y			Z		
	Small	**Medium**	**Large**	**Small**	**Medium**	**Large**	**Small**	**Medium**	**Large**
1	0.79	0.21	0.00	0.00	0.00	1.00	0.16	0.84	0.00
2	0.00	0.43	0.57	0.53	0.47	0.00	0.62	0.38	0.00
3	0.00	1.00	0.00	0.01	0.99	0.00	0.16	0.84	0.00
4	0.18	0.72	0.00	0.00	0.00	1.00	1.00	0.00	0.00

Table 4: Example Fuzzy data set ($L = \{\text{small}, \text{medium}, \text{large}\}$, μ unspecified).

Composite Itemset Value Table: A Composite Itemset Value (CIV) table is a table that allows us to get property values for specific items. For completeness the CIV table for the example raw dataset given in Table 2 is given in Table 5 below.

Item	Property attributes		
	X	**Y**	**Z**
A	2	4	6
B	4	5	3
C	1	2	5
D	4	1	3

Table 5. CIV Table

Property	Linguistic values		
	Low	**Medium**	**High**
X	$v_k \leq 2.6$	$1.9 < v_k \leq 4.2$	$3.7 < v_k$
Y	$v_k \leq 3.9$	$3.1 < v_k \leq 4.8$	$4.3 < v_k$
Z	$v_k \leq 3.0$	$2.3 < v_k \leq 4.1$	$3.6 < v_k$

Table 6. Property Table for raw dataset given in Table 2

Properties Table: A Properties Table is a table that maps all possible values for each property attribute $t_i^P[p_j]$ onto user defined (overlapping) ranges, each associated with a linguistic label taken from the set of available linguistic labels L. Properties tables provide a mapping of property attribute values to membership values according to the correspondence between the given values to the given ranges (linguistic labels). An example is given in Table 6 for the raw data set given in Table 2.

3.2 Fuzzy Specific Concepts

Fuzzy Association Rules: A Fuzzy Association Rule (Kuok et al. 1998) is an implication of the form:

$$\text{if } \langle A \text{ } is \text{ } X \rangle \text{ then } \langle B \text{ } is \text{ } Y \rangle$$

where A and B are disjoint itemsets and X and Y are fuzzy sets. In our case the itemsets are made up of property attributes and the fuzzy sets are identified by linguistic labels (for example “small”, “medium”, “large”).

Fuzzy Frequent Itemset: A property attribute set X , where $A \subseteq P \times L$, is a fuzzy frequent attribute set if its *fuzzy support value* is greater than or equal to a user supplied minimum support threshold (the notion of fuzzy support values is discussed further in sub-section 3.3 below). The significance of fuzzy frequent attribute sets is that fuzzy association rules are generated from the set of discovered frequent attribute sets.

Fuzzy Normalisation: Fuzzy normalisation is the process of finding the contribution to the fuzzy support value, m', for individual property attributes $t_i^p[p_j[l_k]]$ such that a partition of unity is guaranteed. This is given by the equation (where μ is the membership function):

$$t_i'[p_j[l_k]] = \frac{\mu(t_i^p[p_j[l_k]])}{\sum_{x=1}^{|L|} \mu(t_i^p[p_j[l_x]])} \tag{2}$$

Without normalisation, the sum of the support contributions of individual fuzzy sets associated with an attribute in a single transaction may no longer be unity. This is illustrated in Tables 7 and 8 (both taken from the example application outlined in Section 5 below). In the tables, the possible values for the item “Proteins” have been organised into five fuzzy sets labelled: “Very Low” (VL), “Low” (L), “Ideal”, “High” (H) and “Very High” (VH). Table 7 shows a set of raw membership degree values, while Table 8 shows the normalised equivalents.

TID	**Proteins**					…
	VL	**L**	**Ideal**	**H**	**VH**	…
1	0.0	0.0	0.0	0.32	1.0	…
2	0.0	0.38	0.83	0.0	0.0	…
3	…	…	…	…	…	…

Table 7: Fragment of example data set without normalization

TID	**Proteins**					
	VL	**L**	**Ideal**	**H**	**VH**	…
1	0.0	0.0	0.0	0.24	0.76	…
2	0.0	0.31	0.69	0.0	0.0	…
3	…	…	…	…	…	…

Table 8: Fragment of example data set with normalization

In table 7, without normalisation, it would increase the support of protein by 0.32 in row I and 0.21 in row 2. That means, these transactions will be counted 0.32+1.0=1.32 and 0.38+0.83=1.21 times for protein. However, it is unreasonable for one transaction to contribute more than others, if the corresponding discrete sets are disjoint.

The normalisation process ensures fuzzy membership values for each property attribute are consistent and are not affected by boundary values.

Fuzzy Membership Function: Contribution or membership degree to a particular fuzzy set (described by a linguistic label), $t_i[p_j[l_k]]$ is determined by a membership function. There are many different types of membership function and the type of representation of the membership function depends on the nature of the fuzzy set. The most common membership function is an isosceles trapezoidal function, others include triangular, rectangular and semi-circular functions. An example, using the market basket analysis application introduced in Section 1, is given in Figure 2. The figure demonstrates the membership functions for the Protein nutrient. With respect to the application, the trapezoidal shape was chosen as it best captures the intuition (promoted by nutritionists) that nutrient values above or below the ideal is undesirable. Note that the ideal nutrient value equates to 1.

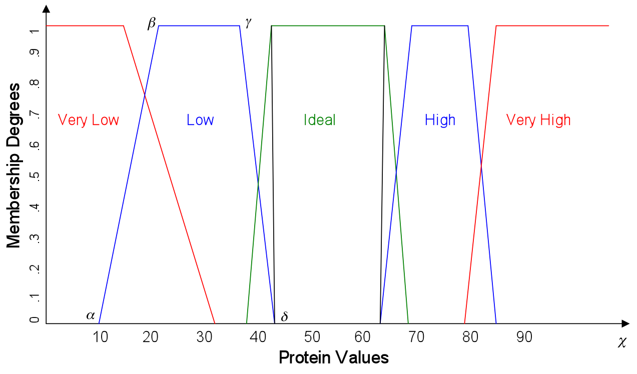

Fig 2: Fuzzy membership functions

$$\mu(x,\alpha,\beta,\gamma,\delta) = \begin{cases} 0, \delta < x < \alpha \\ \dfrac{(x-\alpha)}{(\beta-\alpha)}, \alpha \le x \le \beta \\ \dfrac{(\delta - x)}{(\delta - \gamma)}, \gamma \le x \le \delta \\ 1, \beta < x < \gamma \end{cases} \tag{3}$$

Equation 3 (Paetz, J. 2002) is a function representing all the membership degrees of an input value “x”. Other parameters α , β , γ and δ refer to the corners of the trapezium proceeding in a clockwise fashion starting with the bottom-left corner. The value x has an “ideal” value between the points β to γ along the “X” axis, with the lowest value α and the highest value δ . From the example in figure 2, an “ideal” protein intake will have values in the range [40,60]. If there are missing properties (or trace elements) in an item, as shown in figure 1 (e.g. Bread has no proteins, so called “trace” elements), the fuzzy function evaluates to zero degree membership.

3.3 Quality Measures for Fuzzy Association Rules

A very important aspect in data mining is the discovery of interesting knowledge., where interestingness relates to unexpectedness(Fayyad et al. 1996). Extensive study in databases has been carried out recently in order to find out the most interesting rules with subjective and objective measures. Subjective measures (Silberschatz and Tuzhilin, 1995) take into account the user’s goals and domain knowledge. Objective measures (Freitas, A. 1998) evaluate the interestingness of a rule in terms of rule structure and the underlying data in rule generation such as support, confidence, certainty and entropy.

However, the support-confidence framework remains the most popular approach in traditional ARM and identifies frequent itemsets and assesses the relevance of the generated ARs. The support-confidence framework, with some modifications, can also be applied to composite item fuzzy association rule mining.

Fuzzy Support: Frequent fuzzy attribute sets are identified by calculating fuzzy support (significance) values. Fuzzy Support $(Supp_{Fuzzy})$ is typically calculated as follows:

$$Supp_{Fuzzy}(A) = \frac{\text{Sum of votes satisfying A}}{\text{Number of records in } T}$$

where $A = \{a_1, a_2, a_3, ..., a_{|A|}\}$ is a set of property attribute-fuzzy set (label) pairs such that $A \subseteq P \times L$. A record t'_i “satisfies” A if $A \subseteq t'_i$. The individual vote per record, t_i, is obtaining by multiplying the membership degree associated with each attribute-fuzzy set pair $[i[l]] \in A$:

$$\text{vote for } t_i \text{ satisfying } A = \prod_{\forall [i[l]] \in A} t'_i[i[l]] \tag{4}$$

$$Supp_{Fuzzy}(A) = \frac{\sum_{i=1}^{i=n} \prod_{\forall [i[l]] \in A} t'_i[i[l]]}{n} \tag{5}$$

Note that by using the product operator (often referred to in fuzzy ARM literature as the *mul* operator) for fuzzy aggregation, the degree of contribution of all items is taken into account and thus provides for a more effective result (and also ensures that the overall contribution remains within the range $[0,1]$). Alternatives found in the literature include the *min* and *max* operators as:

$$Supp_{Fuzzy}(A) = \frac{\sum_{i=1}^{i=n} \min_{\forall [i[l]] \in A} (t'_i[i[l]])}{n} \qquad Supp_{Fuzzy}(A) = \frac{\sum_{i=1}^{i=n} \max_{\forall [i[l]] \in A} (t'_i[i[l]])}{n}$$

However these do not include the contribution of all values. Table 9 demonstrates the effect of using mul, min and max fuzzy support calculation using $T' = \{t'_1, t'_2, t'_3, t'_4\}$ and $A = \{a_1, a_2, a_3, a_4\}$. Note that the vote for t'_3 is zero because t'_3 is not a subset of A.

T'	A					vote for t_i satisfying A		
	a_1	a_2	a_3	a_4		***Max***	***Min***	***Mul***
t'_1	0.4	0.6	0.7	0.9	→	0.900	0.400	0.151
t'_2	0.9	0.8	0.5	0.6	→	0.900	0.500	0.216
t'_3	0.7	0.5	0.3	0.8	→	0.800	0.300	0.084
t'_4	0.8	0.9	0.7	0.2	→	0.900	0.200	0.101
					FS(A)	0.875	0.350	0.138

Table 9: Effect of fuzzy *mul* operator

We use an example to illustrate the computation of the fuzzy support value. Let A={X, Z} and P={Small, Large} and a part of database shown in table 4. The fuzzy support of {A, P} is calculated as follows: Support$_{\text{Fuzzy}}$ (A, P)=(0.5+0+0.5)/3=0.33

Fuzzy Confidence: Frequent attribute sets with fuzzy support above the user specified threshold are used to generate all possible rules. A fuzzy AR derived from a fuzzy frequent attribute set C is of the form:

$$A \rightarrow B$$

where A and B are disjoint subsets of the set $P \times L$ such that $A \cup B = C$. Fuzzy Confidence $(Conf_{Fuzzy})$ is calculated in the same manner that confidence is calculated in traditional ARM:

$$Conf_{Fuzzy}(A \rightarrow B) = \frac{Supp_{Fuzzy}(A \cup B)}{Supp_{Fuzzy}(A)} \tag{6}$$

Certainty Measure: The Fuzzy Confidence measure $(Conf_{Fuzzy})$ described above is often criticised because it does not take into account the effect of $Supp_{Fuzzy}(B)$. The certainty measure $(Cert)$ addresses this. The certainty measure is a statistical measure founded on the concepts of *covariance* (Cov) and *variance* (Var). Certainty is calculated using equation 7:

$$Cert(A \rightarrow B) = \frac{Cov(A,B)}{\sqrt{Var(A) \times Var(B)}} \tag{7}$$

Certainty values range between -1 and +1, positive when the dependence between A and B is positive, 0 when there is independence and negative when the dependence is negative. We are only interested in rules that have a certainty value that is greater than 0. As the certainty value increases from 0 to 1, the more related the attributes are and consequently the more interesting the rule is. It is worth noting that the certainty of an association rule reaches its maximum possible value, 1, if and only if the rule is completely accurate (Delgado et al. 2003).

4 The Fuzzy Apriori-T (CFARM) Algorithm

For fuzzy association rule mining standard ARM algorithms can be used or at least adopted after some modifications (Khan et al. 2006), (Gyenesei, A. 2001), (Khan et al. 2008). There is limited work addressing performance issues in fuzzy association rule mining but still there are some contributions in this area (Chen and Wei, 2002), (Khan et al. 2006). An efficient algorithm is required because a significant amount of processing is undertaken to prepare the raw data prior to the application of fuzzy association rule mining. For example, filtration where data is filtered or extracted, specifically edible items from non-edible ones, conversion of quantitative properties into fuzzy sets and normalizing membership contributions of the properties.

The proposed Composite Fuzzy Apriori-T ARM (CFARM) algorithm is developed using T-tree data structures (Coenen et al. 2004[a]) and works in a fashion similar to the Apriori algorithm (Ferenc, B. 2003).

The CFARM algorithm consists of four major steps:

Data preprocessing Steps:

1. Transformation of ordinary transactional data set (T) into a property data set (T^p).
2. Transformation of property data set (T^p) into a fuzzy data set T'.

Association Rule Mining Steps:

3. Apply Fuzzy Apriori-T association rule mining algorithm to T'using fuzzy support, confidence and certainty measures of the form described above to produce a set of frequent item sets F .
4. Process F and generate a set of fuzzy ARs R such that $\forall r \in R$the interestingness threshold (either confidence or certainty as desired by the end user) is above some user specified threshold.

The algorithms for steps 1and 2 are presented in Tables 10 and 11.

```
Input:
$T$ = Raw data set
Output:
$T^p$ = Property data set
1. $T^p = \phi$
2. $foreach\ t_i \in T\ do$
3.     $foreach\ p_k \in P\ do$
4.         $foreach\ i_j \in t_i\ do$
5.             $value \Leftarrow value + t_i[i_j[p_k]]]$
6.         $t_i^p[p_j] \Leftarrow value / | t_i |$
7.     $T^p \Leftarrow T^p \cup t_i^p$
```

Table 10: rawToPropertyDataSetConverter (T)

To illustrate steps 1 and 2 from table 10, a fragment of a raw data set (T) is given in Table 2. This raw data is then cast into a properties data set (T^P). This is done, as described above; by averaging the property values for each transaction (see section 3.1 and table 3). For example, assuming the CIV table given in table 5 and considering transaction $t_1 = \{a,b\}$, from Table 2, a has property values $\{2,4,6\}$ and b has property values $\{4,5,3\}$.

Input:
T^p = property data set

Output:
T' = Fuzzy data set

```
1. $T' = \phi$
2. $foreach\ t_i^p \in T^p\ do$
3.     $foreach\ p_j \in t_i^p\ do$
4.         $foreach\ l_k \in L\ do$
5.             $t_i'[p_j[l_k]] \Leftarrow \mu([t_j^p[p_j]], l_k)$
6.         $T' \Leftarrow T' \cup t_i'[p_j]$
```

Table 11 : propertToFuzzyDataSetConverter (T^p)

Thus $t_1^p = \{(2+4)/2, (4+5)/2, (6+3)/2\} = \{3.0, 4.5, 4.5\}$, assuming the properties table of the form presented in Table 6 where $L = \{\text{small}, \text{medium}, \text{large}\}$. The result is as shown in Table 3 which is then cast into a fuzzy data set T' as shown in Table 4.

The final part of the CFARM algorithm is given in Table 14. In the Table: C_k is the set of candidate itemsets of cardinality k , F is the set of frequent item sets, R is the set of potential rules and R' is the final set of generated fuzzy ARs.

The Fuzzy Apriori-T Algorithm

The Fuzzy Apriori-T algorithm (Apriori-Total) is founded on tree structure called the T-tree (Coenen et al. 2004[b]). This is a set enumeration tree structure in which to store frequent item set information. What distinguishes the T-tree from other set enumeration tree structures is:

1. Levels in each sub-branch of the tree are defined using arrays. This thus permits "indexing in" at all levels and consequently offers computational advantages.
2. To aid this indexing the tree is built in "reverse". Each branch is founded on the last element of the frequent sets to be stored. This allows direct indexing with attribute number rather than first applying some offset.

Thus given a data set of the form (ignoring any fuzzy membership issues):
A=0.6, B=0.5, C=0.3, D=0.2, E=0.8, F=0.3

{ 1 2 3 4 5 6 }
{0.6 0.3 0.2 0.4 0.1 0.9}
{0.5 0.2 0.8 0.5 0.6 0.4}
{0.5 0.2 0.3 0.3 0.2 0.1}

and assuming a support count of less than 0.01, we can identify the following frequent sets (support counts in parenthesis):

1 (0.53)	**1 2** (0.13)	**4 5** (0.13)
2 (0.23)	**1 4** (0.21)	**4 6** (0.19)
3 (0.43)	**1 6** (0.26)	**1 4 6** (0.11)
4 (0.40)	**2 6** (0.12)	
5 (0.30)	**3 4** (0.19)	
6 (0.47)	**3 5** (0.18)	

These can be presented in a T-tree of the form given in Figure 3 (note the reverse nature of the tree).

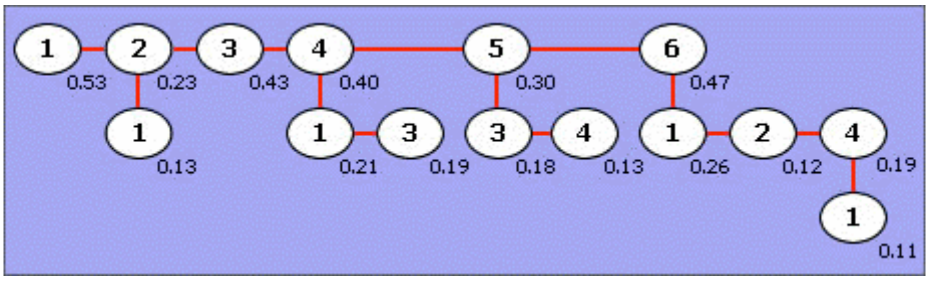
Fig 3: Conceptual example of the T-tree data structure

The internal representation of this "reverse" T-tree founded on arrays of T-tree nodes that can be conceptualised as shown in Figure 4. The storage required for each node (representing a frequent set) in the T-tree is then 12 Bytes:

1. Reference to T-tree node structure (4 Bytes)
2. Support count field in T-tree node structure (4 Bytes)
3. Reference to child array field in T-tree node structure (4 Bytes)

Thus housekeeping requirements are still 8 Bytes, however storage gains are obtained because it is not necessary to explicitly store individual attribute labels (i.e. column numbers representing instantiated elements) as these are implied by the indexing. Of course this approach must also require storage for "stubs" (4 Bytes) where nodes are missing (unsupported). Overall the storage advantages for this technique is thus, in part, dependent on the number of missing combinations contained in the data set.

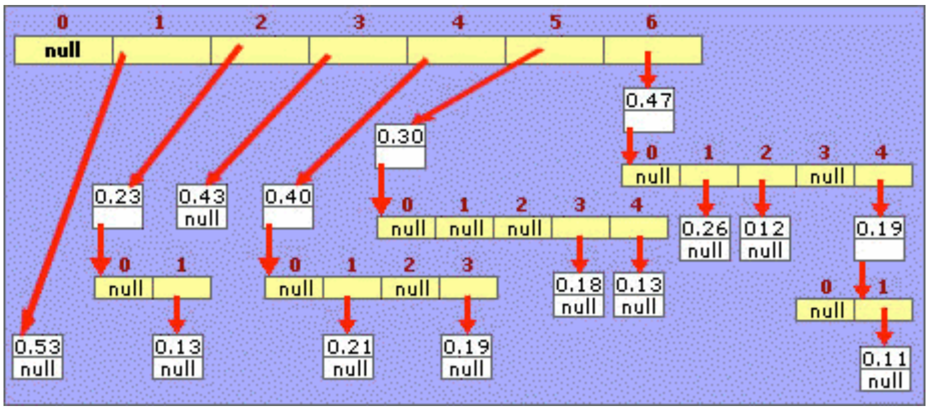

Fig 4: Internal representation of T-tree presented in Figure 3

The T-tree described above is built in an Apriori manner, as first proposed in (Ferenc, B. 2003), starting with the one item sets and continuing until there are no more candidate N-itemsets. Thus, at a high level, a standard Apriori algorithm is used (see table 12).

```
K ← 1
nextlevelFlag=true;

generate candidate K-itemsets
Loop
    count support values for candidate K-itemsets
    prune unsupported K-itemsets
    K ← 2
    generate candidate K2 itemsets from previous level
    if no K2 itemsets
        break
end Loop
```

Table 12 : Apriori Algorithm

```
Method: createTotalSupportTree
Arguments: none
Return: none
Fields: NA
------------------------------
createTtreeTopLevel()
generateLevel2()
createTtreeLevelN()
------------------------------
```

Table 13 : *The* `createTotalSupportTree` *method*

In more detail the Apriori-T algorithm commences with a method createTotalSupportTree which is presented in Table 13. The method starts by generating the top level of the T-tree (createTtreeTopLevel) and then generating the next level (generateLevel2) from the supported sets in level 1. Remember that if a 1-itemset is not supported, none of its super sets will be supported according to downward closure property in ARM. Once we have generated level 2 further levels can be generated (createTtreeLevelN).

The method to generate the top level of a T-tree is as presented in Table 14. Note that the method includes a call to a general T-tree utility method pruneLevelN described later.

```
  Method: createTtreeTopLevel
  Arguments: none
  Return: none
  Fields: D: number of attributes
          startTtreeRef: start of T-tree
          dataArray 2D: array holding input sets
  --------------------------------------
Dimension and initialise top level of T-tree(length=D)

  Loop from i= 0 to i = number of records in dataArray
      Loop   j=0   to   j=number   of   attributes   in
dataArray[i]
          startTtreeRef[i][j]++
      End loop
  End Loop

  pruneLevelN(startTtreeRef,1)
  --------------------------------------
```

Table 14 : *The* `createTtreeTopLevel` *method*

The generateLevel2 method loops through the top level of the T-tree creating new T-tree arrays where appropriate (i.e. where the immediate parent nodes is supported). The method is outlined in Table 15. Note that the method includes a call to a general T-tree utility method generateNextLevel (also described later).

```
Method: createTtreeLevelN
Arguments: none
Return: none
Fields: startTtreeRef: start of T-tree
        nextlevelFlag: set true if next level exists
--------------------------------------
K <-- 2
while (nextlevelFlag)
    addSupportToTtreeLevelN(K)
    pruneLevelN(startTtreeRef,K)
    nextlevelFlag ← false
    generateLevelN(startTtreeRef,K,{})
    K ← K+1
End loop
```

Table 15 : *The* `createTtreeLevelN` *method*

Once we have a top level T-tree and a set of candidate second levels (arrays) we can proceed with generating the rest of the T-tree using an iterative process --- the createTtreeLevelN method presented in Table 15. The createTtreeLevelN method calls a number of other methods addSupportToTtreeLevelN, pruneLevelN (also called by the createTtreeTopLevel method) and generateLevelN which are presented in Tables 16, 17 and 18 respectively.

```
  Method: addSupportToTtreeLevelN
  Arguments: K the current level
  Return: none
  Fields: startTtreeRef: start of T-tree
          dataArray 2D: array holding input sets
  -------------------------------------
  Loop i = 0 to i = number of records in dataArray
      length ← number of attributes in dataArray[i]

addSupportToTtreeFindLevel(startTtreeRef,K,length,
                  dataArray[i]
  End loop
  -------------------------------------

  Method: addSupportToTtreeFindLevel
  Arguments:
      linkref: refers to current array in T-tree
      K: level marker
      Length: array length at current branch in t-tree
          record input data record under consideration
  Return: none
  Fields: None
  -------------------------------------
  if (K=1)
      Loop from i = 0 to i = length
          if (linkref[record[i]] != null)
                increment
linkref[record[i]].weightedsupport
          End if
      End Loop
  else
      Loop from i = K-1 to i = length
          if (linkref[record[i]] != null &&
                     linkref[record[i]].childRef      !=
null)

addSupportToTtreeFindLevel(linkref[record[i]].childRe,
                          K-1,i,record)
          End if
      End loop
  end if else
  -------------------------------------
```

Table 16 : `addSupportToTtreeLevelN`, `addSupportToTtreeFindLevel` methods

```
  Method: pruneLevelN
  Arguments: linkref reference to current array in T-
tree
          K level marker
  Return: true if entire array pruned
  Fields: minSupport the minimum support threshold
  --------------------------------------
  if (K=1)
      allUnsupported <-- true
      Loop from i = 1 to i = length of array
          if (linkref[i] != null)
              if (linkref[i].support < minSupport)
                          linkref[i] <-- null
              else allUnsupported <-- false
              End if else
          End if
      return allUnsupported
      End Loop
  else
      Loop from i = K to i = length of array
          if (linkref[i] != null)
              if (pruneLevelN(linkref[i].childRef,K-1)
                  linkref[i].childRef <-- null
              End if
           End if
      End loop
  End if else
  --------------------------------------
```

Table 17 : *The* `pruneLevelN`

```
Method: generateLevelN
  Arguments: linkref reference to current array in T-
tree
          K level marker
          I the item set represented by the parent node
Return: None, Fields: None
  --------------------------------------
  if (K=1)
      Loop from i = 2 to i = length of array
          if (linkref[i] != null)
              generateNextLevel(linkref,i,I union i)
          End if
      End loop
  else
    Loop from i = K to i = length of array
     if (linkref[i]!=null && linkref[i].childRef != null)
        generateLevelN(linkref[i].childRef,K-1,I union i
  --------------------------------------
  Method: generateNextLevel
  Arguments: linkref reference to current array in T-tree
          I: index to parent node in vurrent array
          I: the item set represented by the parent node
  Return: None
  Fields: nextLevelExists flafg set to true or false
  --------------------------------------
  linkref[i].childRef<--empty t-tree array nodes length i

  Loop from j = 1 to j = i
       if (linkRef[j] != null)
           newI ← I union j
           if (all newI true subsets all supported in the
                         T-tree sofar)
               linkRef[i].childRef[j] <-- new T-tree Node
               nextLevelExists <-- true
           else linkRef[i].childRef[j] <-- null
           End if else
       End if
  End loop
  ------------------------------------------
```

Table 18 : *The* `generateLevelN` *method* and its related `generateNextLevel` method

5 An Example Application

To evaluate our approach we used a real market basket analysis data set (FIMI), comprising 1600 composite edible items out of 16,469 total distinct products; the objective is to determine consumers' consumption patterns for different nutrients according to the government promoted Recommended Daily Allowance (RDA). The properties for each item comprised the 27 nutrients contained in the government sponsored RDA table (the complete list of nutrients is given in table 19 (a)).

#	Nutrient	#	Nutrient	#	Nutrient	#	Nutrient
1.	Biotin	8.	Folacin	15.	Protein	22.	VitaminB6
2.	Calcium	9.	Iodine	16.	Riboflavin	23.	Vitamin C
3.	Carbohydrate	10.	Iron	17.	Selenium	24.	Vitamin D
4.	Cholesterol	11.	Magnesium	18.	Sodium	25.	Vitamin E
5.	Copper	12.	Manganese	19.	Thiamin	26.	Vitamin K
6.	Fats	13.	Niacin	20.	Vitamin A	27.	Zinc
7.	Fiber	14.	Phosphorus	21.	Vitamin B12		

Table 19 (a): Nutrients listed in RDA table.

This RDA table is thus the CIV table used in the evaluation with actual nutrient values for individual items. The property data set will therefore comprise $1600 \times 27 = 43200$ attributes. The linguistic label set L was defined as follows L – {Very Low (VL), Low (L), Ideal (I), High (H), Very High (VH)}. Thus the set of fuzzy attributes $A = P \times L$ has $27 \times 5 = 135$ attributes. A fragment of this data set is given in Table 19 (b).

Nutrients / Fuzzy Ranges	**Very Low**				**Low**				**Ideal**				**High**				**Very High**		
	Min	**Core**		**Max**	**Min**	**Core**		**Max**	**Min**	**Core**		**Max**	**Min**	**Core**		**Max**	**Min**	**Core**	
Fiber	0	1	10	15	10	15	20	25	20	25	30	35	30	33	38	39	35	40	…
Iron	0	.6	8	12	8	12	16	18	16	18	19	20	19	20	22	23	22	23	…
Protein	0	1	15	30	10	20	35	40	35	40	60	65	60	65	75	80	75	70	…
Vit.A	0	15	150	200	150	200	300	400	300	350	440	500	440	490	550	600	550	600	…
Zinc	0	.8	8	10	8	10	15	20	15	20	30	40	30	40	46	50	46	50	…
	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…

Table 19 (b): Fragment of market basket composite item data set[1].

For the example application we used Trapezoidal membership function shown in figure 2. Core (β - γ) is the region where an attribute has a full membership degree i.e. 1. Min (α - β) is the region before core where the value approaches core and Max (γ - δ) is the region after core region where an attribute membership value start decreasing until zero.

A representative fragment of a raw data set (T), comprising edible items, is given in Table 20(a). This raw data is then cast into a properties data set (T^P) using the given CIV/RDA table to give the properties data set in Table 20(b).

[1] *Values could be in grams, milligrams, micrograms, International unit or any unit)*

.Here Min is the minimum value i.e. α *, Core is the core region* β, δ *and Max is the maximum value* γ *in the fuzzy membership graph.of figure 2.*

TID	Items
1.	30, 31, 32
2.	33, 34, 35
3.	36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46
4.	38, 39, 47, 48
5.	38, 39, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58
6.	32, 41, 59, 60, 61, 62
7.	3, 39, 48
8.	63, 64, 65, 66, 67, 68
9.	32, 69
10.	48, 70, 71, 72

(a) Fragment of retail data set (T)

(b) Property data set (T^P)

TID	Bio	Cal	Car	Chl	…
1	1	762	255	68	…
2	0	2	0	201	…
3	19	246	1240	1295	…
4	…	…	…	…	…

(c) Classical ARM data set

TID	Bio	Cal	Car	Chl	…
1	2	7	11	16	…
2	1	6	11	18	…
3	5	6	15	16	…
4	…	…	…	...	…

Table 20**:** Example data fragment from example application

At this point, two solutions may exist for the next mining step. One is to code fuzzy sets {very low, low, ideal, high, very high} as, for example, {1, 2, 3, 4, 5}, for the first nutrient (Biotin), {6, 7, 8, 9, 10} for the second nutrient (Calcium) and so on (Muyeba et al. 2006). The encoded data (Table 20(c)) can be mined by any binary association rule algorithm to find association rules. This approach only gives us, for instance, the total support of various fuzzy sets per nutrient and not the degree of (fuzzy) support. This directly affects the number and quality of rules (see section 6 for further evidence). To overcome this, the fuzzy approach advocated in this paper has been adopted, where we convert RDA property data set (Table 20(b)) to linguistic values (Table 21) for each nutrient and corresponding degrees of membership for the fuzzy sets they represent. Each transaction then will have fuzzy values {very low, low, ideal, high, very high} for each nutrient present in every item of that transaction. Table 21 shows only two nutrients (i.e. a total of 10 fuzzy sets).

TID	Calcium (Cal)					Carbohydrate (Car)					
	VL	L	Ideal	H	VH	VL	L	Ideal	H	VH	…
1	0.0	1.0	0.0	0.0	0.0	0.83	0.17	0.0	0.2	0.0	…
2	0.4	0.0	0.0	0.0	0.0	0.0	0.0	0.95	0.05	0.0	…
3	0.0	0.4	0.46	0.0	0.0	0.0	0.0	0.0	0.0	1.0	…
4	…	…	…	…	…	…	…	…	…	…	…

Table 21**:** Linguistic transaction file

Note that table 21 shows a normalised transaction file according to the calculation examples in previous sections and using equation (2).

We have generated frequent sets using both discrete quantitative and fuzzy approaches. Discrete approach generated 76 frequent sets and fuzzy approach with un-normalised data and normalised data produced 43 and 39 frequent sets respectively. Rules generated using certainty factor with threshold 0.5 are shown in table 22.

Discrete Method with Crisp Intervals	**Support**	**Fuzzy Approach with Normalised Data**	**Support**	**Certainty**
5→25	0.500	5→25	0.500	0.500
25→35	0.600	25→35	0.600	0.612
25→35→62	0.400	25→35→62	0.400	0.667
25→35→110	0.400	25→35→110	0.400	0.667
25→40	0.600	25→40	0.600	0.612
25→40→95→115	0.400	25→40→95→115	0.496	0.816
25→40→115	**0.500**	**25→40→115**	**0.496**	**0.816**
25→115	**0.500**	**25→115**	**0.496**	**0.500**
35→62	0.400	35→62	0.400	0.667
35→110	0.400	35→110	0.400	0.667
40→95→115	**0.500**	**40→95→115**	**0.496**	**0.816**
40→115	**0.500**	**40→115**	**0.496**	**0.816**
85→95→135	**0.500**	**85→95→135**	**0.417**	**0.713**
85→135	**0.500**	**85→135**	**0.417**	**0.713**

Table 22**:** Rules generated using discrete and fuzzy methods

By comparing the itemsets supports with discrete and fuzzy approaches, it can be noted that for some rules (highlighted) using discrete method the support is slightly higher as compared to the same rules generated by fuzzy approach. This is due to the fact that in some cases discrete intervals allow more contribution of items near interval boundaries (sharp boundary problem), in result, higher the support and consequently more frequent sets. But in case of overlapped fuzzy intervals each item contribute its actual contribution in one or more intervals with overall maximum contribution=1 using normalisation. Table 22 gives a formal proof of that

6 Experimental Evaluation

In order to show the quality, performance and effectiveness of our approach, two sets of experiments were undertaken:

1. Comparison of CFARM, with and without normalisation, against standard (discrete) ARM to illustrate: (i) the difference of number of frequent sets generated and (ii) the number of rules generated (using both the confidence and the certainty interestingness measures).
2. Comparison of execution times (performance) using Fuzzy Apriori ARM (Khan et al. 2006), Fuzzy Apriori-T (CFARM) with and without normalisation and Apriori-TFP (Coenen et al. 2004[b]) with discrete dataset (boolean attributes).

All the experiments were performed on Mobile Intel(R) Pentium(R) 4 CPU 3.06 GHz machine with 1 GB ram and installed Microsoft Windows XP Professional with service pack 2 as operating system.

6.1 Datasets

Both real and synthetic datasets are used in experiments. For real data we used Retail dataset, it is a real market basket data (FIMI) and T10I4D100K synthetic data is obtained from the IBM dataset generator (IBM).

Dataset	# of Transactions	Distinct Items	Avg. Trans. Size	Max. Trans. Size
Retail	88,162	16,469	10.3	76
T10I4D100K	100,000	1000	10.1	30

Table 23: Frequent itemsets comparison

Table 23 characterises the two datasets in terms of the number of transactions, the number of distinct items, the average transaction size, and the maximum transaction size. It is worth mentioning that both datasets contains sparse data, since most association rules discovery algorithms were designed for these types of problems.

For the purpose of the experiments we mapped the item numbers onto products in a real RDA table.

6.2 Quality Measures

For experiment one both the real retail and synthetic datasets described above were used. Figures 5 and 6 show the results and demonstrates the difference between the numbers of frequent itemsets generated using

I. Standard ARM using discrete intervals,
II. CFARM with fuzzy partitions without normalization (CFARM1), and
III. CFARM with fuzzy partitions with normalization (CFARM2).

For the standard ARM the Apriori-TFP algorithm was used (Coenen et al. 2004[b]) and for CFARM-1 and CFARM-2 proposed Fuzzy Apriori-T was used with a range of support thresholds [0.15-0.6] for both algorithms.

As expected the number of frequent itemsets increases as the minimum support decreases. From the results, it is clear that standard ARM produces more frequent itemsets (and consequently rules) than fizzy ARM. This is because the frequent itemsets generated more accurately reflect the true patterns in the data set than the numerous artificial patterns resulting from the use of crisp boundaries in standard ARM. At low support threshold level the approach with normalization (CFARM2) starts to produce less frequent itemsets than the approach without normalization (CFARM1). This is because the average contribution to support counts per transaction is greater without using normalization than with normalization.

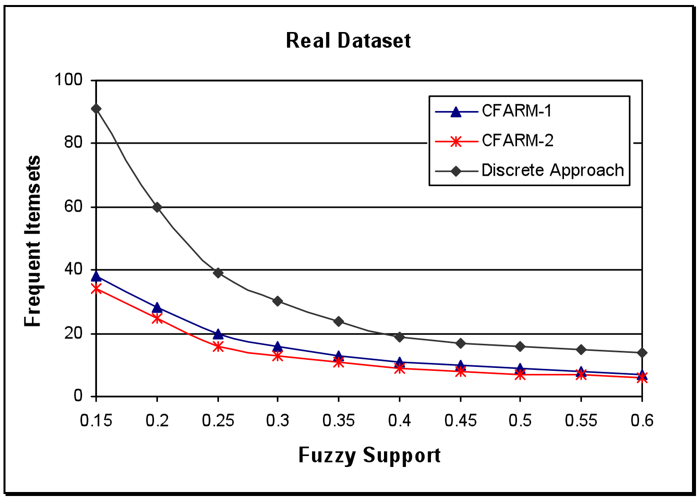

Fig 5: Number of frequent Itemsets

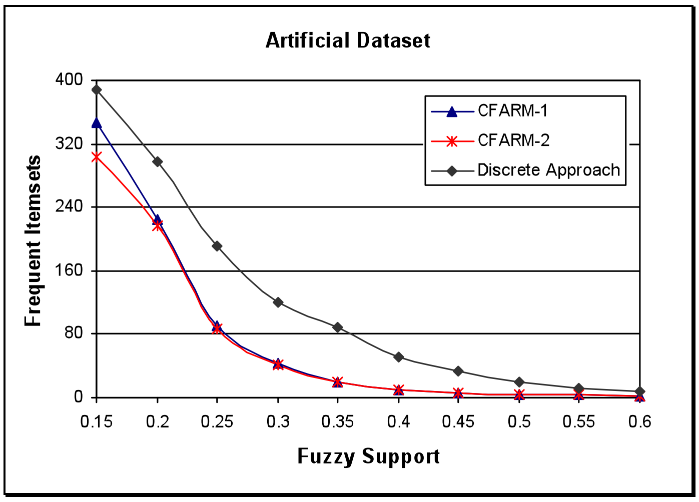

Fig 6: Number of frequent Itemsets

Figures 7 and Figure 8 show the comparison of number of interesting rules produced using specified fuzzy confidence and certainty thresholds [0.1-0.9] respectively with real and synthetic data. In both cases, the number of interesting rules is less as using CFARM2; this is a direct consequence of the fact that CFARM2 generates fewer frequent itemsets. Note that fewer, but arguably better, rules are generated using the certainty measure than the confidence measure (Figures 7 & 8) because the more related the attributes are and consequently the more interesting the rule is (see sec. 3.3).

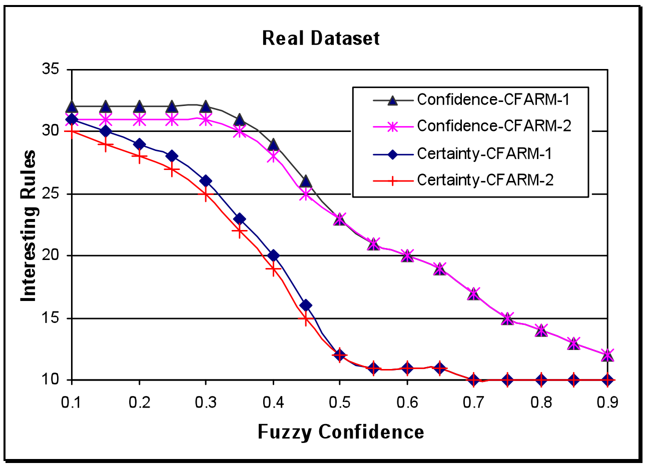

Fig 7: Number of Interesting Rules using Confidence

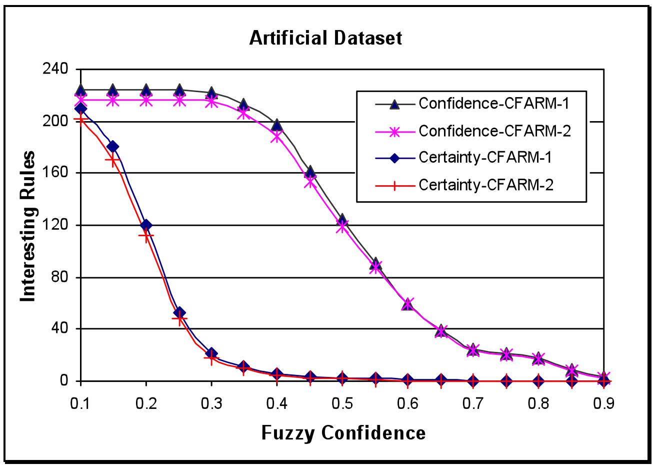

Fig 8: Number of Interesting Rules using Certainty

The experiments show that using the proposed fuzzy normalization process less fuzzy ARs are generated. In addition, the novelty of the approach is its ability to analyse datasets comprised of composite items where each item has a number of property values such as the nutritional property values used in the application described here.

Some example fuzzy ARs produced by our approach are as follows:

IF *Protein* intake is *Ideal* THEN *Carbohydrate* intake is *low*.
IF *Protein* intake is *Low* THEN *Vitamin A* intake is *High*.
IF *Protein* intake is *High* AND *Vitamin A* intake is *Low* THEN *Fat* intake is *High*.
Note that, for the above rules we have replaced quantitative numeric data with real linguistic values for better understanding.

The rules above would first need defuzzifying (Roy and Pedrycz, 2001) each fuzzy value and then either aggregating them into real values or presenting them as a tabled list of nutritional values, whichever is appropriate to be interpreted by a domain expert. These rules would therefore be useful in analysing customer buying patterns concerning their nutrition.

6.3 Performance Measures

Experiment two investigated the effect on execution time caused by varying the size of data (number of records) and the number of attributes with and without normalization. For this experiment we have compared the execution time of proposed Fuzzy Apriori-T CFARM algorithm with Fuzzy Apriori ARM (Khan et al. 2006) using fuzzified datasets and Apriori-TFP (Coenen et al. 2004^b) with discrete data (boolean attributes). A support threshold = 0.4, confidence = 0.5 and certainty value = 0.5 were used for all algorithms.

Figures 9 and 10 show the effect on execution time by increasing the number of records. To obtain different sizes both the datasets were partitioned into 10 equal partitions labelled 10K, 20K… 100K.. All 27 nutrients (properties) were used.

Note that the running time for candidate generation is independent of the number of records but run time for threshold count (support, confidence and certainty) is directly proportional to the number of records. Thus we would expect the CFARM algorithm to have near-linear scale up.

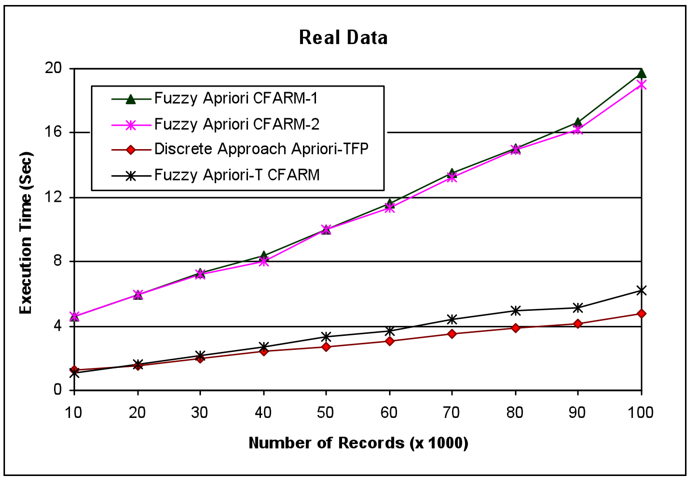

Fig 9: Performance Measures: Number of Records

From the figures it can be seen that discrete method using Apriori-TFP algorithm and CFARM approach using Fuzzy Apriori-T algorithm have almost similar timings, the slight difference is due to the extra computational cost while generating fuzzy frequent sets. But a big difference of execution time can be noted between Fuzzy Apriori ARM and Fuzzy Apriori-T algorithms. This is due to the efficient frequent sets generation with Fuzzy Apriori-T algorithm using T-tree data structures. While the execution time increasing with the number of records. However the experiments also show that CFARM scales linearly with the number of records. Note that Fuzzy Apriori-T has the similar timings for dataset with and without normalisation, so only results with normalised data are shown in the figures.

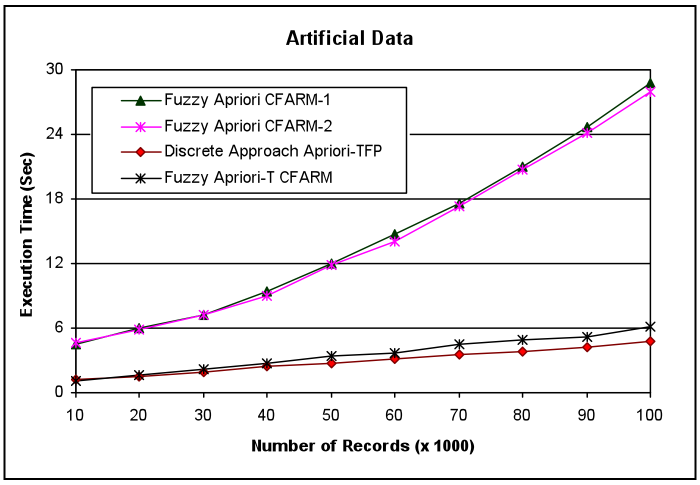

Fig 10: Performance Measures: Number of Records

Figures 11 and 12 show the effect on execution time by varying the numbers of attributes. Recall that each attribute has 5 fuzzy sets, therefore for (say) 27 attributes, we have 135 columns.

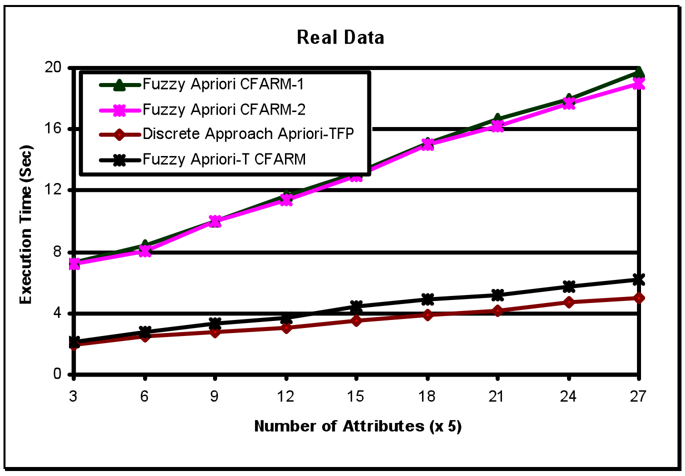

Fig 11: Performance Measures: Number of Attributes

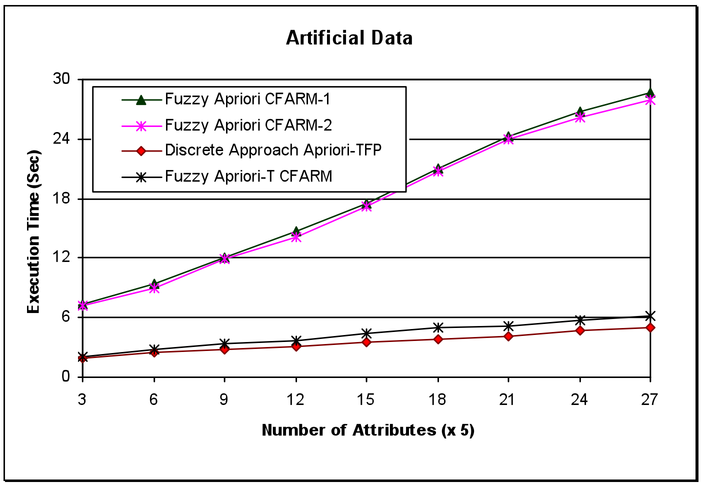

Fig 12: Performance Measures: Number of Attributes

In this experiment, we give the experimental results based on the performance of the algorithm by varying the number of attributes. As expected, similar results are produced i.e. as the number of attributes increases the execution time increases as well.

Further, the experiments show that proposed Fuzzy Apriori-T CFARM has better execution time, almost similar to Apriori-TFP, presently one of the efficient algorithm (Coenen et al. 2004[a, b]) and the proposed Fuzzy Apriori-T scales linearly with the number of records and attributes.

7 Conclusion

In this paper, we have presented a novel approach for extracting hidden information from composite items. We defined composite items and developed a fuzzy framework for mining such items and measuring their interestingness using fuzzy measures. We showed that within such items, common properties can be defined as quantitative (sub) itemsets, transformed into fuzzy sets, mined and extracted rules measured using fuzzy support, confidence and certainty. Using the proposed CFARM algorithm a more succinct set of fuzzy association rules can be derived using fuzzy measures and certainty. The CFARM algorithm thus represents a new way of mining items efficiently with properties than standard quantitative ARM which does not use such properties, as described in the literature. We also showed the application of our method with market basket data. As noted there is significant potential to apply CFARM to other applications than the daily recommended allowance analysis, used to illustrate the operation of CFARM in this paper, with composite items or attributes even with varying fuzzy sets between attributes. We feel that the approach presented here is novel in its use of composite items for fuzzy association rule mining. Further work will consider composite fuzzy association rule mining framework with weighted items and how the fuzzy approach can incorporate weights in generated composite fuzzy sets.

References

1. Agrawal, R., Imielinski, T. & Swami, A. (1993) Mining association rules between sets of items in large databases. In: In Proceedings of ACM SIGMOD International Conference on Management of Data. pp. 207-216.
2. Au, W.H. & Chan, K.C.C. (1999) FARM:A Data Mining System for Discovering Fuzzy Association Rules, in: Proc. 8th IEEE int'l Conf. on Fuzzy Systems, (Seoul, Korea, 1999), 1217 - 1222.
3. Berzal. F., I. Blanco, D. S´anchez, and M.A. Vila, (2002) Measuring the accuracy and interest of association rules: A new framework, Intelligent Data Analysis 6 (3), pp. 221–235..
4. Chen. G, and Wei. Q, (2002) Fuzzy Association Rules and the Extended Mining Algorithms, Information Sciences, 147(1-4) pp. 201 - 228.
5. Coenen, F., Goulbourne, G. & Leng, P. (2004) Tree structures for mining association rules. Data Mining and Knowledge Discovery, 8 (1), pp.25–51 (a).
6. Coenen, F.P., Leng, P., and Ahmed, S. (2004) Data Structures for Association Rule Mining: T-trees and P-trees, IEEE Transactions on Data and Knowledge Engineering, Vol. 16, No 6, pp774-778 (b)..
7. Delgado, M., Marin, M., Martin-Bautista MJ, Sanchez D, Vila MA: Mining Fuzzy Association Rules: An Overview, proc. of the BISC International Workshop on Soft Computing for Internet and Bioinformatics 2003.
8. Delgado, M., Sanchez, D. & Vila, M.A. (2002) Acquisition of fuzzy association rules from medical data. Fuzzy Logic in Medicine, pp.286.
9. Delgado, M., Sánchez, D., Martín-Bautista, M.J. & Vila, M.A. (2001) Mining association rules with improved semantics in medical databases. Artificial Intelligence in Medicine, 21 (1-3), pp.241–245.
10. Delgado, M., Marin, N., Sanchez, D. & Vila, M.A. (2003) Fuzzy association rules: general model and applications. IEEE Transactions on Fuzzy Systems, 11 (2), pp.214–225.
11. Dong. L. and Tjortjis. C., (2003) Experiences of Using a Quantitative Approach for Mining Association Rules, in: Lecture Notes in Computer Science Series, Vol. 2690 (Springer, Berlin, 2003) 693 - 700.
12. Dubois, D., Hüllermeier, E. and Prade, H. (2006) A Systematic Approach to the Assessment of Fuzzy Association Rules, Data Mining and Knowledge Discovery Journal, 13(2), 167 – 192.
13. Fayyad, U.M., Piatetsky-Shapiro, G. and Smyth, P., (1996) From Data Mining to Knowledge Discovery: An Overview. In Advances in Knowledge Discovery & Data Mining, AAAI/MIT. pp. 1-34.
14. Ferenc Bodon, (2003) A fast APRIORI implementation, IEEE ICDM Workshop on Frequent Itemset Mining Implementations (FIMI'03), Melbourne, Florida, USA.
15. FIMI, Frequent Itemset Mining Implementation Repository, http://fimi.cs.helsinki.fi/.
16. Freitas, A.A., (1998) On Objective Measures of Rule Surprisingness. In: Proceedings of Second European Symposium on Principle of Data Mining and Knowledge Discovery (PKDD-98), Lecturer Notes in Artificial Intelligence, 1510, pp. 1-9.
17. Gyenesei, A.: (2001) A Fuzzy Approach for Mining Quantitative Association Rules, Acta Cybernetical, Vol. 15, No. 2, pp.305-320.
18. IBM Synthetic Data Generator, http://www.almaden.ibm.com/software/quest/resources/index.html
19. Khan, M.S., Muyeba, M., Tjortjis, C. & Coenen, F. (2006) An effective Fuzzy Healthy Association Rule Mining Algorithm (FHARM). Proc. of UKCI 2007, the

7th Annual Workshop on Computational Intelligence (FUZZ-IEEE 07' co-located), 4 (5), pp.14.

20. Khan. M., Muyeba, M and Coenen, F. (2008) Mining Fuzzy Association Rules from Composite Items, in IFIP International Conference on Artificial Intelligence (IFIP-AI 2008), Milano, Italy, Volume 276; Artificial Intelligence and Practice II; Max Bramer; (Boston: Springer), pp. 67–76.
21. Kim. W., Banerjee.J., Chou. H., Garza. J. and Woelk. D., (1987) Composite object support in an object-oriented database system, in: Proc. OOPSLA'87 (Orlando, Florida, United States) 118 - 125.
22. Kim. W., Bertino. E. and Garza. J., (1989) Composite objects revisited, ACM SIGMOD Record 18(2) 337 – 347.
23. Kuok, C.M., Fu, A. & Wong, M.H. (1998) Mining fuzzy association rules in databases. ACM Sigmod Record, 27 (1), pp.41–46.
24. Mohammad Khabbaz, Keivan Kianmehr, Mohammed Al-Shalalfa, Reda Alhajj, (2008) Effectiveness of Fuzzy Classifier Rules in Capturing Correlations between Genes. Intl J of Data Warehousing and Mining IJDWM 4(4): 62-83.
25. Muyeba. M, M. Sulaiman Khan, Z. Malik, and C. Tjortjis, (2006) Towards Healthy Association Rule Mining (HARM), A Fuzzy Quantitative Approach, in: Proc. IDEAL'06, Lecture Notes in Computer Science, Vol. 4224, (Springer, Verlag, 2006), 1014 - 1022.
26. Paetz. J., (2002) A Note on Core Regions of Membership Functions, in: Proc. (EUNITE 2002), (Albufeira, Portugal, 2002) 167 - 173.
27. Roychowdhury. S. and Pedrycz, W. (2001) A survey of defuzzification strategies, International journal of intelligent systems., vol. 16, no 6, pp.679-695.
28. S´anchez. D., (1999) Acquisition of Relationships between Attributes in Relational Databases), Ph.D. thesis, Department of Computer Science and Artificial Intelligence, University of Granada, December 1999.
29. Silberschatz, A. and Tuzhilin, A., (1995) On subjective measures of interestingness in knowledge discovery. In: Fayyad, U., Uthurusamy, R. (Eds.), Proceedings of the 1st ACM SIGKDD international conference on knowledge discovery and data mining KDD-1995, AAAI/MIT Press, Cambridge. pp. 275-281.
30. Silverstein. C., Brin. S., and Motwani. R., (1998) Beyond market baskets: Generalizing association rules to dependence rules, Data Mining and Knowledge Discovery, vol. 2, pp. 39–68.
31. Srikant, R. & Agrawal, R. (1996) Mining quantitative association rules in large relational tables. ACM SIGMOD Record, 25 (2), pp.1–12.
32. Terence Kwok, Kate A. Smith, Sebastián Lozano, David Taniar, (2002) Parallel Fuzzy c-Means Clustering for Large Data Sets. Proceedings of the 8th International Conference Euro-Par, LNCS Vol 2400 Springer, Euro-Par 2002: 365-374.
33. Wang, K., Liu, J.N. & Ma, W. (2006) Mining the Most Reliable Association Rules with Composite Items. In: Sixth IEEE International Conference on Data Mining Workshops, 2006. ICDM Workshops 2006. pp.749–754.
34. Ye, X. & Keane, J.A. (1997) Mining composite items in association rules. In: Proceedings of the 1997 IEEE International Conference on Systems, Man, and Cybernetics (SMC 1997), Hyatt Orlando, Orlando, Florida, USA. pp.1367–1372.