Incorporating Deviation-Detection Functionality into the OLAP Paradigm

Carem C. Fabris
CEFET-PR
CPGEI
Av. Sete de Setembro, 3165.
Curitiba-PR, 80230-901. Brazil
caremf@zipmail.com.br

Alex A. Freitas
PUC-PR
PPGIA-CCET
Rua Imaculada Conceicao, 1155
Curitiba-PR, 80215-901. Brazil
alex@ppgia.pucpr.br
http://www.ppgia.pucpr.br/~alex

Abstract

Although the integration of data mining and OLAP is obviously of great interest (if only due to the widespread availability of commercial-strength OLAP tools), relatively little progress has been made in this integration. The main goal of this paper is to contribute to this integration, by incorporating some data mining functionality (more precisely, deviation-detection functionality) into the OLAP paradigm. The data mining method addressed in this paper is Attribute Focusing. This method discovers interesting attribute values, in the sense that the observed frequency of those values deviate from their expected frequency. However, standard Attribute Focusing assumes that attribute values are "flat" (i.e. non-hierarchical), which is a serious limitation for its application in the OLAP paradigm. In order to make Attribute Focusing useful for OLAP we have adapted this method to the kind of hierarchical dimension often found in data cubes.

1. Introduction

Both data mining and OLAP (OnLine Analytical Processing) can be regarded as technologies developed for supporting strategic decision-making. However, there are several important differences between these two technologies. For our purposes, it is useful to draw attention to two of these differences.

The first one involves system autonomy. It is generally agreed that data mining systems are considerably more autonomous than OLAP ones. Indeed, in OLAP the human analyst is responsible for formulating hypotheses and making major decisions about the data analysis process, such as to which part of the data he/she should drill-down/roll-up. The analyst uses an OLAP tool's functionality to help his/her navigation through the data cube and to verify (validate or refute) his/her hypotheses, but a conventional OLAP tool can hardly suggest new hypothesis or data-navigation directions. By contrast, most data mining algorithms have considerably more autonomy to formulate new hypotheses - say new candidate rules - and evaluate them on the data, returning to the user only a (hopefully) relatively small set of interesting patterns.

Another distinction between data mining and OLAP is that the former is often carried out using "flat" data, without any hierarchy of attribute values (there are exceptions – see e.g. [8], [9]). By contrast, OLAP is usually performed in data cubes with hierarchical attribute values.

Hence, an integration of data mining and OLAP can combine the best of both worlds. Incorporating a data mining method into an OLAP system would make that system somewhat more autonomous. The obvious advantage is that the output of the data mining method could be used to partially alleviate the workload of human data analysis, for instance automatically suggesting new directions for drill-down and other data cube-navigation operations. In other words, the results of a data mining algorithm can help the user to navigate the data cube in a less *ad hoc* manner.

In addition, such an integration extends the range of kinds of databases which can be mined by the data mining method, giving that method access to increasingly popular multidimensional databases and widespread commercial-strength OLAP tools. A related advantage is that the data mining method can benefit from precomputed aggregation information widely available in data warehouses and data cubes.

Although the integration of data mining and OLAP is obviously of great interest, relatively little progress has been made in this integration, which is sometimes called *OnLine Analytical Mining*. (There are of course some exceptions, and some projects integrating data mining and OLAP will be reviewed later.)

In this paper we achieve an integration between data mining and OLAP by adapting a data mining method – namely Attribute Focusing [1],[2], a deviation detection method (see section 2) - to the OLAP framework of data cubes. The main obstacle to this integration is that standard Attribute Focusing assumes that attribute values are "flat", i.e. non-hierarchical, which is not the case in data cubes. We overcome this obstacle by adapting Attribute Focusing to the kind of hierarchical dimensions often found in data cubes.

In other words, we extend the functionality of the OLAP paradigm by incorporating some data mining functionality (more precisely, deviation-detection functionality) into that paradigm. To summarize, in this paper Attribute Focusing is applied to a data cube organized in a star scheme and with hierarchical dimensions, as typically found in OLAP systems.

The rest of this paper is organized as follows. Section 2 reviews Attribute Focusing and the task of deviation detection. Section 3 proposes a way to adapt Attribute Focusing for hierarchical dimensions, which are commonly found in OLAP. Section 4 reports the results of experiments running the proposed modified Attribute Focusing. Section 5 discusses related work. Finally, section 6 concludes the paper.

2. Review of Attribute Focusing and Deviation Detection

Attribute Focusing is a method designed for detecting interesting attribute values, in the sense that those values differ from the expected value [1], [2]. Attribute Focusing is one of the simplest data mining methods. Its simplicity is partly due to the fact that it was designed for the task of deviation detection. In this task the goal is to detect interesting patterns in the data, where interestingness has to do with deviation from expected patterns [13]. In general, this task is considerably simpler (from an algorithmic viewpoint) than other tasks involving prediction, such as classification.

However, one should not underestimate the importance of deviation detection nor the usefulness of Attribute Focusing. For one thing, Attribute Focusing and other deviation detection algorithms address the problem of discovering *interesting* patterns. In the context of data mining, one can argue that interestingness is more important than predictive accuracy. This basic argument can be traced back to the origin of modern data mining. Although the basic idea of data mining is quite old, one can consider that the birth of modern data mining and knowledge discovery as a self-described scholarly discipline was the IJCAI-89 Workshop on Knowledge Discovery in Real Databases [18]. This workshop helped to de-emphasize the role of predictive accuracy as a measure the discovered-knowledge quality. For instance, when asked to describe *bad* directions for future research, a member of the workshop panel (J. Ross Quinlan) said that: “A more serious example of bad research is trying to show how to squeeze out the last half of one percent of performance” [18]. Although the discovery of interesting patterns can certainly be achieved in prediction-related tasks such as classification [15], [5], [17], [14], it seems that the deviation detection task lends itself naturally to the discovery of truly interesting patterns. Since this task does not focus on predictive accuracy, the data mining algorithm is free to focus on the degree of interestingness of discovered patterns. Indeed, the deviation detection task has led to the discovery of some truly interesting, useful patterns [16].

In particular, Attribute Focusing has been successfully used to discover interesting, useful knowledge [1], [2], and it has been the target of parallelization techniques aimed at increasing its computational efficiency when mining large databases [7].

2.1 The Interestingness Functions of Attribute Focusing

As mentioned above, the goal of Attribute Focusing is to detect interesting attribute values, i.e. attribute values that significantly deviate from their expected values. In order to achieve this goal, [1] has proposed two interestingness functions, I_1 and I_2.

The function I_1 is used to detect interesting values of a given attribute. The function I_1 measures how much the observed frequency of an attribute value deviates from its expected frequency assuming a uniform probability distribution. More precisely,

$$I_1(A=V_i) = |\mathrm{Pr}(A=V_i) - 1/\mathrm{Card}(A)| , \qquad (1)$$

where A is the attribute being analyzed, V_i is i-th value of the domain of A, $\mathrm{Pr}(A=V_i)$ is the probability of attribute A having value V_i, $\mathrm{Card}(A)$ is the cardinality of the domain of attribute A, and $|x|$ denotes the absolute value of x. $\mathrm{Pr}(A=V_i)$ is computed as the ratio of the number of records in which $A=V_i$ over the number of records in which A has some value (i.e. records with missing values of A are ignored). The higher the value of I_1, the more interesting the corresponding attribute value is.

The function I_2 is used to detect an interesting pair of attribute values, where each of the two values belongs to a different attribute. This function measures how much the observed joint frequency of a pair of attribute values deviates from the expected frequency assuming that the two attributes are statistically independent. More precisely,

$$I_2(A=V_i,B=V_j) = |\mathrm{Pr}(A=V_i,B=V_j) - \mathrm{Pr}(A=V_i) \times \mathrm{Pr}(B=V_j)| , \qquad (2)$$

where A, V_i and $\mathrm{Pr}(A=V_i)$ are as defined above for formula (1), B, V_j and $\mathrm{Pr}(B=V_j)$ are defined in the obviously analogous way, and $\mathrm{Pr}(A=V_i,B=V_j)$ is the probability that both A has value V_i and B has value V_j. This probability is computed as the ratio of the number of records in which both $A=V_i$ and $B=V_j$ over the number of records in which both A and B have some value (i.e. records with missing values of A or B are ignored). The higher the value of I_2, the more interesting the corresponding pair of attribute values is. It should be noted that, unlike I_1, I_2 is useful for detecting interesting attribute interactions. Attribute interactions have a crucial role in the discovery of interesting knowledge, as argued by [6].

3. Adapting Attribute Focusing for Hierarchical Dimensions

The definitions of the function I_1 and I_2 discussed in the previous section implicitly assume that all attributes are "flat", i.e. they have no hierarchy. This is a common situation when mining data extracted from a relational DBMS.

However, in the OLAP framework hierarchical attribute values are very common. The existence of hierarchical dimensions in data cubes introduces new opportunities and requirements for adapting the computation of the functions I_1 and I_2 of Attribute Focusing. We discuss below an example for the case of function I_2, since this is the more challenging case. Similar, somewhat simpler opportunities and requirements exist for the case of function I_1.

Suppose we use attribute focusing with function I_2 to analyze sales of a product and find interesting combinations of values of two attributes, say store, with hierarchy store $\rightarrow$ city $\rightarrow$ state, and time, with hierarchy day $\rightarrow$ month $\rightarrow$ year. Should the function I_2 for combinations of states and years (highest hierarchical level of both attributes) be computed in the same way as for the combinations of stores and days (lowest hierarchical level of both attributes)? What about combinations of stores and years (mixed hierarchical levels)?

We address this kind of question by modifying the computation of the Attribute Focusing functions I_1 and I_2 to take into account a correction factor based on the hierarchical levels of the two attribute values being analyzed.

The first step of this modification consists of defining the bias of Attribute Focusing concerning the hierarchical level of the attribute(s) being analyzed. In other words, should Attribute Focusing favor the discovery of interesting attribute values at higher or lower levels of the attribute(s) being analyzed?

We have chosen the former option, mainly for two reasons. First, this decision is in accordance with one of the basic goals of data mining, namely the discovery of comprehensible patterns. In general, the higher the hierarchical level of an attribute value, the higher its associated level of abstraction, and so the more comprehensible the attribute value tends to be for the user.

Second, selecting interesting attribute values at higher hierarchical levels has the advantage of facilitating drill-down by the user. The user can first see the result of Attribute Focusing and then use it to decide to which finer-grain region of the data cube he/she will drill down. This is important because most OLAP users perform drill-down more often than roll-up, since top-down data analyses are intuitively easier (for a human analyst) than bottom-up analyses.

In the next two subsections we discuss in detail how we modify the two interestingness functions of Attribute Focusing to favor the discovery of attribute values at higher hierarchical levels.

3.1 Modifying the Function I_1 for Hierarchical Dimensions

The correction factor for function I_1, denoted CF_1, is defined as:

$$CF_1 = ((NHL_A - L_A) / NHL_A)^{1/2}, \qquad (3)$$

where NHL_A is the total number of hierarchical levels of attribute A and L_A is the number of the hierarchical level of attribute A being currently analyzed by Attribute Focusing. L_A ranges from 0 for the root (highest) level to k -1 for the deepest level, where k is the number of levels in the hierarchy. This correction factor is multiplied by the current value of I_1 to get the new, corrected value of I_1. Hence, we have:

$$I_1(A{=}V_i) = |\Pr(A{=}V_i) - 1/\mathrm{Card}(A)| \times CF_1 \qquad (4)$$

To understand the effect of CF_1 in formula (4), let us consider an example. Assume the following attribute hierarchy with three levels: State (level 0) → County (level 1) → City (level 2). Suppose that we want to compute the value of I_1(*City=New York*). In this case we would have: $NHL_A = 3$ and $L_A = 2$, so that $CF_1 = 0.577$. Now suppose that we want to compute the value of I_1(*State=Texas*). In this case we would have: $NHL_A = 3$ and $L_A = 0$, so that $CF_1 = 1$. Hence, attribute values at lower hierarchical levels, such as *City=New York*, are penalized by having a correction factor significantly smaller than attribute values at higher hierarchical levels. In particular, attribute values at the root level of the hierarchy, such as *State=Texas*, have a correction factor of 1, which corresponds to no penalty at all in formula (4).

3.2 Modifying the Function I_2 for Hierarchical Dimensions

The correction factor for function I_2, denoted CF_2, is defined as follows:

$$CF_2 = ((NHL_{max} - (L_A + L_B)/2) / NHL_{max})^{1/2}\ , \qquad (5)$$

where $NHL_{max} = \max(NHL_A, NHL_B)$; NHL_A and NHL_B are the total number of hierarchical levels of attributes A and B, respectively; L_A and L_B are the number of the hierarchical level of attributes A and B being currently analyzed by Attribute Focusing. L_A and L_B range from 0 for the root (highest) level to k -1 for the deepest level, where k is the number of levels in the hierarchy of the corresponding attribute. This correction factor is multiplied by the current value of I_2 to get the new, corrected value of I_2. Hence, we have:

$$I_2(A{=}V_i,B{=}V_j) = |\Pr(A{=}V_i,B{=}V_j) - \Pr(A{=}V_i) \times \Pr(B{=}V_j)| \times CF_2 \qquad (6)$$

The effect of CF_2 in formula (6) is similar to the effect of CF_1 in formula (4), i.e. it penalizes attribute values at deeper hierarchical levels and so favors the discovery of interesting attribute values at higher (closer to the root) hierarchical levels.

4. Computational Results.

Henceforth, we will refer to Attribute Focusing with the modified interestingness functions shown in formulas (4) and (6) as HAF (Hierarchical Attribute Focusing), for short. We have applied our HAF method to a real-world data cube containing insurance data. The original data was stored in a star-scheme format in disk, containing a fact table with 51332 records and 9 dimension tables. For the purposes of our experiment the data was loaded into main-memory arrays, to make the computation of Attribute Focusing more efficient.

Note the goal of our experiments is different from most OLAP experiments. Most of the OLAP research is efficiency-oriented, i.e. the main goal of an experiment is usually to measure the processing time required to run an OLAP function. By contrast, since our research extends the OLAP paradigm with a data mining technique, it is effectiveness-oriented, i.e. the main goal of our experiment is to evaluate the quality of discovered knowledge. The issues of computational efficiency and scalability are somewhat orthogonal to our proposal of our new HAF method. Readers interested in computational efficiency issues are referred to [7] and [11].

For our experiments, we have manually selected five dimensions, which seem to be the dimensions more promising for the discovery of interesting patterns. The selected dimensions were Claimant, Covered_Item, Event_Time, Insured_Party, and Policy. Figure 1 shows, for each of these dimensions, which were the attributes used in our experiments. Attribute-value hierarchies are indicated by nesting the names of the attribute levels. For instance, the dimension Claimant has an attribute Gender with just one hierarchical level, attribute Age_Group with just one hierarchical level, and an attribute with three hierarchical levels: State → Count → City. The number of the hierarchical level shown between brackets ranges from 0 for the root (highest) level to k -1 for the deepest level, where k is the number of levels in the hierarchy.

We have applied our HFA method using function I_1 to each of the five dimensions shown in Figure 1. For each dimension, the algorithm was applied to each hierarchical level of each attribute. In each case the value of the function I_1 given by formula (4) was computed. Then our system sorts the discovered interesting attributes values in decreasing order of I_1 value and shows the user the attribute values with the n highest I_1 values (in order), where n is a user-specified parameter. The value of this parameter can be set according to the user's available time. In other words, a very busy user will probably want to analyze, say, just the most interesting 10 attribute values, while a less busy user might want to analyze, say, the most interesting 50 attribute values. In the experiments reported here we set n = 20.

```
CLAIMANT
├─Gender          (level 0)
├─State           (level 0)
│ └──County             (level 1)
│        └── City              (level 2)
└─Age_Group   (level 0)

COVERED_ITEM
├─ Covered_Item_Type    (level 0)
└─ Covered_Item_Description    (level 0)

EVENT_TIME
├─Week_Number    (level 0)
├─Year                 (level 0)
│  └── Quarter                (level 1)
│         └── Month_Name        (level 2)
├─ Week_Day_Name   (level 0)
└─ Fiscal_Year          (level 0)
    └── Fiscal_Quarter   (level 1)

INSURED_PARTY
├─Gender          (level 0)
├─State           (level 0)
│ └──County             (level 1)
│        └── City              (level 2)
└─Age_Group      (level 0)

POLICY
└─Risk_Grade (level 0)
```

Figure 1: Description of the dimensions of the data cube used in our experiments

Table 1 shows the results. Each row of this table corresponds to an interesting attribute value discovered by our HAF, as identified by the first three columns of the table. The fourth column indicates the number of levels in the hierarchy of the corresponding attribute, and the fifth column indicates the number of the hierarchical level which the discovered attribute value belongs to. The notation for the number of the hierarchical level is the same as used in Figure 1. Actually, the information contained in the fourth and fifth columns could be obtained from Figure 1, but we decided to repeat it in this table for convenience of the reader. The sixth column of Table 1 shows the value of the function I_1 for the corresponding attribute value.

Four out of the five dimensions shown in Figure 1 had at least some attribute value among the 20 interesting values output by HAF. The only exception was the Policy dimension. Out of the 4 dimensions with discovered interesting values, three contain hierarchical attributes: Claimant, Insured_Party and Event_Time (as shown in Figure 1). Out of the 20 interesting values shown in Table 1, 12 belong to hierarchical attributes. Out of these 12 values, 8 are values at the root of the corresponding hierarchy. In addition, 8 values belonging to non-hierarchical attributes were also deemed interesting by hierarchical Attribute Focusing. Therefore, in total, out of the 20 attribute values shown in Table 1, 16 are root-level values. (Note that the value of a non-hierarchical attribute can also be considered a “root-level” value, in the sense that it is the root of its own one-node hierarchy.) This is evidence that the use of the correction factor in formula (4) is performing the desired role of favoring of discovery of attribute values closer to the root of an attribute hierarchy.

Table 1: Interesting attribute values detected using function I_1.

Dimension	Attribute	Attr. Value	Total No. of Levels	Level of Att. Value	I_1
Claimant	State	PA	3	0	0.42297
Claimant	State	DE	3	0	0.42297
Insured_Party	State	PA	3	0	0.42215
Insured_Party	State	DE	3	0	0.42215
Event_Time	Weekday_Flag	y	1	0	0.21511
Event_Time	Weekday_Flag	n	1	0	0.21511
Claimant	Claimant_type	Policy_Holder	1	0	0.18655
Claimant	Claimant_type	Third_party_indiv	1	0	0.18655
Claimant	City	Philadelphia	3	2	0.13809
Insured_Party	City	Philadelph	3	2	0.13595
Event_Time	Fiscal_Year	1997	2	0	0.09896
Event_Time	Fiscal_Year	2000	2	0	0.08740
Covered_Item	CovItem_Type	Automobil_Sedan	1	0	0.08419
Covered_Item	CovItem_Desc	Four_Door_Car	1	0	0.08419
Event_Time	Fiscal_Year	1996	2	0	0.08241
Event_Time	Fiscal_Year	1998	2	0	0.07893
Claimant	County	New_Castle	3	1	0.07319
Insured_Party	County	New_Castle	3	1	0.07252
Covered_Item	CovItem_Type	Auto_SportCoupe	1	0	0.06883
Covered_Item	CovItem_Desc	Two_Door_Car	1	0	0.06883

It is interesting to note that the four attribute values with largest I_1 value involve the same root level of State in two dimensions: Claimant and Insured_Party. This pattern draws attention to the fact that insurance claims are unevenly distributed across the states of PA and DE, with respect to both the Claimant and the Insured Party. This seems an important pattern, which can be useful for further analyses of the data. For instance, it suggests that a drill-down on these states is a promising direction for navigation in the data cube.

Table 2: Interesting attribute values detected using function I_2.

Dimens.-1	Attr.-1	AttrValue-1	Dimens.-2	Attrib.-2	AttrValue-2	I_2
Claimant	Gender	F	Insur_party	Gender	F	0.2467
Claimant	Gender	F	Insur_party	Gender	M	0.2467
Claimant	Gender	M	Insur_party	Gender	F	0.2467
Claimant	Gender	M	Insur_party	Gender	M	0.2467
Claimant	County	Philadelphia	Insur_party	County	Philadelphia	0.1495
Claimant	County	Montgomery	Insur_party	County	Montgomery	0.1412
Claimant	City	Philadelphia	Insur_party	County	Philadelphia	0.1295
Claimant	County	Philadelphia	Insur_party	City	Philadelph	0.1295
Claimant	County	Bucks	Insur_party	County	Bucks	0.1192
Claimant	City	Philadelphia	Insur_party	City	Philadelph	0.1057
Claimant	County	Delaware	Insur_party	County	Delaware	0.1005
Claimant	County	Chester	Insur_party	County	Chester	0.0921
Claimant	State	PA	Insur_party	State	PA	0.0710
Claimant	State	PA	Insur_party	State	DE	0.0710
Claimant	State	DE	Insur_party	State	DE	0.0710
Claimant	State	PA	Insur_party	County	New_Castle	0.0648
Claimant	State	DE	Insur_party	County	New_Castle	0.0648
Claimant	County	New_Castle	Insur_party	State	DE	0.0648
Claimant	City	Wilmington	Insur_party	State	DE	0.0580
Claimant	State	PA	Insur_party	City	Wilmington	0.0580
Claimant	State	DE	Insur_party	City	Wilmington	0.0580
Claimant	County	New_Castle	Insur_party	County	New_Castle	0.0580
Claimant	City	Wilmington	Insur_party	County	New_Castle	0.0502
Claimant	County	New_Castle	Insur_party	City	Wilmington	0.0502
Claimant	County	Philadelphia	Insur_party	County	Montgomery	0.0437
Claimant	City	Wilmington	Insur_party	City	Wilmington	0.0410
Claimant	City	Philadelphia	Insur_party	County	Montgomery	0.0379
Claimant	County	Philadelphia	Insur_party	County	Bucks	0.0354
Claimant	City	Philadelphia	Insur_party	County	Bucks	0.0306
Insur_party	State	PA	Event_time	Fiscal_Year	1999	0.0213
Insur_Party	State	DE	Event_time	Fiscal_Year	1999	0.0213
Claimant	State	PA	Event_time	Fiscal_Year	1999	0.0211
Claimant	State	DE	Event_time	Fiscal_Year	1999	0.0211
Claimant	County	New_Castle	Event_time	Fiscal_Year	1999	0.0211
Claimant	County	Bucks	Insur_party	County	Delaware	0.0203
Insur_party	County	New_Castle	Event_time	Fiscal_Year	1999	0.0195
Claimant	City	Southeastern	Insur_party	County	Chester	0.0187
Claimant	County	Chester	Insur_party	City	Southeaste	0.0187
Claimant	City	Norristown	Insur_party	City	Norristown	0.0187
Claimant	County	Montgomery	Insur_party	Gender	F	0.0185

As another example of the utility of discovered knowledge, two of the discovered attribute values were: “Claimant’s County = New Castle” and “Insured_Party’s Count = New Castle”. Hence, the pattern “Count = New Castle” was again considered interesting in two dimensions, Claimant and Insured_Party. This reinforces the suspicion that there is something really special about New Castle. Actually, a drill-down from States to County in the dimensions Claimant and Insured_Party could reveal that the unexpected observed frequency of New Castle is one of the main reasons for the unexpected observed frequency of the above-mentioned pattern involving the states PA and DE. Hence, HAF can not only suggest promising directions for drill-down, but also provide the user with an educated guess about the possible results of a future drill-down.

More interesting conclusions can be certainly drawn from the results in Table 1 by an expert in the meaning of the data and the business of the company.

We have applied our HAF method using function I_2 to each possible combination of two dimensions that can be formed out of the five dimensions shown in Figure 1. For each dimension pair, the algorithm was applied to each possible combination of two attributes, one from each dimension. For each attribute pair, the algorithm was applied to each possible combination of hierarchical levels of the two attributes, in the cases where the attributes are hierarchical. In each case the value of the function I_2 given by formula (6) was computed. Then our system sorts the discovered interesting pairs of attributes values in decreasing order of I_2 value and shows the user the pairs of attribute values with the n highest I_2 values (in order), where n is a user-specified parameter, as explained above. In the experiments reported here we set $n = 40$.

Table 2 shows the results. Each row of this table corresponds to a pair of interesting attribute values discovered by HAF. The first attribute value of the pair is identified by the first three columns of the table, while the second is identified by the next three columns of the table. The last column of Table 2 shows the value of the function I_2 for the corresponding pair of attribute values.

As can be seen in Table 2, the vast majority of the pairs of interesting attribute values discovered by HAF consisted of one attribute from the dimension Claimant and one attribute from the dimension Insured_Party. Again there is some evidence that the use of the correction factor in formula (6) is performing the desired role of favoring the discovery of attribute values closer to the root of an attribute hierarchy; although in this case the evidence is not as strong as in the case of the results of Table 1. More precisely, in 20 out of the 40 interesting pairs of values shown in Table 2, at least one of the two values is at the root of the corresponding hierarchy. (Unlike Table 1, the actual hierarchical level of each attribute value is not shown in Table 2 due to lack of space.)

It is interesting to note that the four pairs of attribute values with largest I_2 value involve all the possible combinations of Gender values for the dimensions Claimant and Insured_Party. This indicates that the joint Gender distribution across these two dimensions is significantly different from the expected joint distribution assuming statistical independence. One of the possible reasons for these unbalanced Gender distributions can be the fact that the Claimant and the Insured_Party can often be husband and wife subjects, which causes the Gender combinations “M-F” and “F-M” to co-occur much more often than the Gender combinations “M-M” and “F-F”. Since the function I_2 of HAF discovers both surprisingly-rare and surprisingly-common combinations of attribute values (like the standard Attribute Focusing method), all the above four combinations are deemed interesting by HAF. It is interesting to note that this occurs despite the fact that neither the Claimant’s Gender alone nor the Insured_Party’s Gender alone turned out to be very interesting, since these attributes do not occur in Table 1. By drawing the attention of the user to this kind of pattern, HAF is effectively helping in data analysis. Again, more interesting conclusions can be certainly drawn from the results in Table 2 by an expert in the meaning of the data and the business of the company.

5. Related Work

In most of the projects proposing some sort of integration between data mining and OLAP the main goal is either to use precomputed aggregations of data cubes to make data mining more efficient or to improve the computational efficiency of OLAP, without extending the functionality of this paradigm. An example of the former approach is the work of [10], which constructs data cubes specifically for improving the efficiency of a data mining algorithm. This is very different from our approach, which uses data mining to extend the functionality of OLAP.

An example of the latter approach is the work of [19], which uses a clustering technique to estimate the probability density of multidimensional data. This leads to the ability to answer OLAP queries in a significantly more efficient and flexible way, by storing only a compact representation of the database produced by the clustering algorithm. However, the basic functionality of OLAP is essentially the same as in the traditional OLAP paradigm. To be fair and more precise, their system also somewhat improves the functionality of OLAP, by allowing OLAP querying over continuous dimensions (whereas traditional OLAP uses only categorical dimensions) and allowing any dimension to be used as a measure (whereas traditional OLAP requires fixed measure dimensions). However, the kinds of queries allowed by their system are still definitely OLAP-style, aggregate queries, in which the user specifies a region in the multidimensional space of the data cube and an aggregation function (count, sum, average, etc.) to be applied to the data in the specified region. In contrast, our work incorporates data mining functionality into an OLAP system, so that this latter can automatically run a data mining method – Hierarchical Attribute Focusing, in our case.

Now we turn to two systems that, like our work, have the goal of incorporating some data mining functionality into OLAP. The DBMiner system [9], [10], [3] includes attribute-oriented algorithms that can exploit attribute hierarchies. In this project the attribute hierarchies are usually either specified by a domain expert or automatically generated by a clustering algorithm. Although the system was originally developed to mine relational databases, rather than OLAP, multidimensional ones, it can be used to mine multidimensional data. Another system integrating data mining and OLAP is proposed by [20]. This system discovers Roll Up Dependencies, which are a kind of generalization of functional dependencies for hierarchical attributes. Yet another related work is described by [12]. In this project, if the data cube already contains relevant precomputed aggregations, they are used to render association rule mining more efficient. However, if those aggregations are not available, association rule mining proceeds in a manner tightly integrated with data cube computation. These systems use data mining methods very different from the Hierarchical Attribute Focusing method developed in our research, and they have no need for the attribute hierarchy-related correction factors proposed by our research. Actually, those systems address data mining tasks different from the task of deviation detection addressed by our system.

To the best of our knowledge, [7] is the only previous work which has extended Attribute Focusing for multi-dimensional databases (data cubes). The main contribution of their work was to introduce a parallel algorithm to compute the interestingness function I_2 given by formula (2). Hence, their research addressed the problem of making Attribute Focusing more computationally efficient, which is important for large data cubes. However, it did not adapt Attribute Focusing to one of the major characteristics of data cubes, namely the fact that dimensions contain *hierarchical* attributes. This adaptation was precisely the goal of our work.

6 Conclusions

We have adapted a data mining method for deviation detection, namely Attribute Focusing, for the OLAP framework. From the viewpoint of a data mining practitioner, our work has two main contributions. First, it extends the range of kinds of databases which can be mined by Attribute Focusing, giving that method access to increasingly popular multidimensional data warehouses and widespread commercial-strength OLAP tools. Second, it allows Attribute Focusing to benefit from precomputed information aggregation widely available in multidimensional data warehouses, which tends to render Attribute Focusing more efficient.

From the viewpoint of an OLAP user, our work has two main contributions. First, it increases the autonomy of OLAP systems, i.e. the results of Hierarchical Attribute Focusing are interesting patterns that can be directly used as a basis for intelligent decision making. Second, the output of Hierarchical Attribute Focusing can be used to help the user to navigate through the data cube. An OLAP user can rely on the interesting attribute values discovered by Attribute Focusing to decide which drill-down/roll-up operations are promising, rather than navigating the data cube in an *ad hoc* manner.

A full integration between data mining and OLAP is still a very open problem. There is a very wide range of data mining algorithms that in principle could also be incorporated into an OLAP system, provided that the necessary adaptations are performed. As future research we intend to continue to adapt other data mining methods for an OLAP framework. For instance, we have designed an algorithm for discovering surprising patterns by detecting occurrences of Simpson's paradox [4], which is currently being adapted to the OLAP framework.

Acknowledgments

We are very grateful to Kurt E. Allebach, who has given us a copy of the real-world data set used in the experiments reported in this paper.

References

[1] Bhandari, I. (1993) Attribute Focusing: machine-assisted knowledge discovery applied to software production process control. *Proc. AAAI-93 Ws. on Knowledge Discovery in Databases*, 61-69. AAAI.

[2] Bhandari, I. and Biyani, S. (1994) On the role of statistical significance in exploratory data analysis. *Proc. AAAI-94 Workshop on Knowledge Discovery in Databases*, 61-72. AAAI Press.

[3] Carter, C.L. and Hamilton, H.J. (1998) Efficient attribute-oriented generalization for knowledge discovery from large databases. *IEEE Trans. Knowledge and Data Engineering 10(2)*, 193-208.

[4] Fabris, C.C. and Freitas, A.A. (1999) Discovering surprising patterns by detecting occurrences of Simpson's paradox. In: *Research and Development in Intelligent Systems XVI*, 148-160. Springer.

[5] Freitas, A.A. (1999) On rule interestingness measures. *Knowledge-Based Systems 12(5-6)*, 309-315.

[6] Freitas, A.A. (2001) Understanding the Crucial Role of Attribute Interaction in Data Mining. *To appear in Artificial Intelligence Review journal*.

[7] Goil, S. and Choudhary, A. High-performance OLAP and data mining on parallel computers. *Data Mining and Knowledge Discovery 1(4)*, 391-417. 1997.

[8] Han, J. and Fu, Y. (1996) Explorations of the power of attribute-oriented induction in data mining. In: Fayyad, U.M. et al. (Eds.) *Advances in Knowledge Discovery and Data Mining*, 399-421.

AAAI/MIT.
[9] Han, J.; Fu, Y.; Wang, W.; Chiang, J.; Gong, W.; Koperski, K; Li, D.; Lu, Y.; Rajan, A.; Stefanovic, N.; Xia, B. and Zaiane, O.R. (1996) DBMiner: a system for mining knowledge in large relational databases. *Proc. 2nd Int. Conf. Knowledge Discovery & Data Mining*, 250-255. AAAI Press.
[10] Han, J.; Gong, W. and Yin, Y. (1998) Mining segment-wise periodic patterns in time-related databases. *Proc. 4th Int. Conf. Knowledge Discovery & Data Mining*, 214-218. AAAI Press.
[11] Hung, E.; Cheung, D.W.; Kao, B. and Liang, Y. (2000) An optimization problem in data cube system design. *Knowledge Discovery and Data Mining (Proc. 4th Pacific-Asia Conf. – PAKDD 2000)*, *Lecture Notes in Artificial Intelligence 1805*, 74-85. Springer-Verlag.
[12] Kamber, M.; Han, J. and Chiang, J.Y. (1997) Metarule-guided mining of multi-dimensional association rules using data cubes. *Proc. 3rd Int. Conf. Knowledge Discovery & Data Mining*, 207-210. AAAI.
[13] Klosgen, W. (1996) Explora: a multipattern and multistrategy discovery assistant. In: Fayyad, U.M. et al. (Eds.) *Advances in Knowledge Discovery and Data Mining*, 249-271. AAAI/MIT.
[14] Liu, B.; Hsu, W. and Chen, S. (1997) Using general impressions to analyze discovered classification rules. *Proc. 3rd Int. Conf. Knowledge Discovery & Data Mining*, 31-36. AAAI Press.
[15] Liu, B.; Hsu, W. and Ma, Y. (1999) Pruning and summarizing the discovered associations. *Proc. 5th ACM SIGKDD Int. Conf. Knowledge Discovery & Data Mining*, 125-134. ACM.
[16] Matheus, C.J.; Piatetsky-Shapiro, G. and McNeil, D. (1996) Selecting and reporting what is interesting: the KEFIR application to health data. In: Fayyad, U.M. et al. (Eds.) *Advances in Knowledge Discovery and Data Mining*, 495-516. AAAI/MIT Press.
[17] Padmanabhan, B. and Tuzhilina, A. (1998) A belief-driven method for discovering unexpected patterns. *Proc. 4th Int. Conf. Knowledge Discovery & Data Mining (KDD-98)*, 94-100. AAAI Press.
[18] Piatetsky-Shapiro, G. (1991) Knowledge discovery in real databases: A report on the IJCAI-89 Workshop. *AI Magazine* 11(5), 68-70, Jan. 1991.
[19] Shanmugasundaram, J.; Fayyad, U. and Bradley, P.S. (1999) Compressed data cubes for OLAP aggregate query approximation on continuous dimensions. *Proc. 5th ACM SIGKDD Int. Conf. Knowledge Discovery & Data Mining*, 223-232. ACM.
[20] Wijsen, J.; Raymond, T.N.; Calders, T. (1999) Discovering Roll-Up Dependencies. *Proc. 5th ACM SIGKDD Int. Conf. Knowledge Discovery & Data Mining*, 213-222. ACM, 1999.