Top_Keyword: an Aggregation Function for Textual Document OLAP

Franck Ravat, Olivier Teste, Ronan Tournier, Gilles Zurfluh

IRIT (UMR5505), Université de Toulouse, 118 route de Narbonne
F-31062 Toulouse Cedex 9, FRANCE
{ravat,teste,tournier,zurfluh}@irit.fr

Abstract. For more than a decade, researches on OLAP and multidimensional databases have generated methodologies, tools and resource management systems for the analysis of numeric data. With the growing availability of digital documents, there is a need for incorporating text-rich documents within multidimensional databases as well as an adapted framework for their analysis. This paper presents a new aggregation function that aggregates textual data in an OLAP environment. The TOP_KEYWORD function (TOP_KW for short) represents a set of documents by their most significant terms using a weighing function from information retrieval: *tf.idf*.

Keywords. OLAP, Aggregation function, Data warehouse, Textual measure.

1 Introduction

OLAP (On-Line Analytical Processing) systems allow analysts to improve the decision making process. These systems are based on multidimensional modelling of decisional data [6]. In OLAP systems, data is analysed according to different levels of detail and aggregation functions (e.g. sum, average, minimum…) are used to provide a synthetic view. Drilling operations (i.e. drill-down and roll-up), which are frequently used by decision makers, make intensive use of aggregation functions. For example, in **Fig. 1**, a decision maker analyses the number of keywords used within publications by authors and by month. To get a more global vision the decision maker changes the analysis detail level and rolls up from the month detail level of the TIME analysis axis to the year level. As a consequence, monthly values are aggregated into yearly values, applying the selected aggregation function (COUNT).

Analysis based on numeric centric multidimensional database is a well mastered technique [15]. However, according to a recent study only 20% of the data that spreads over an information system is numeric centric [17]. The remaining 80%, namely "digital paperwork," mainly composed of textual data, stays out of reach of OLAP due to the lack of tools and adapted processing. Not taking into account these data leads inevitably to the omission of relevant information during an analysis process and may end in producing erroneous analyses [17].

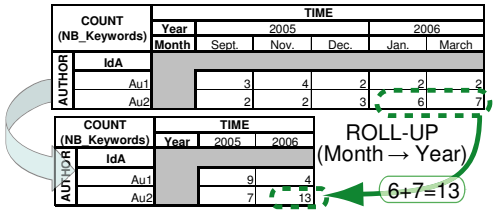

Fig. 1 An analysis with a Roll-up process.

Recently, XML[1] technology has provided a wide framework for sharing documents within corporate networks or over the Web. Textual data in XML format is now a conceivable data source for OLAP systems.

By *multidimensional document analysis* throughout this paper we mean to analyse in an OLAP framework text-rich document data sources, e.g. conference proceedings, quality control reports, e-books…

The problem may be summarised as follows: during an analysis of data extracted from text-rich XML documents, textual analysis indicators are used. However, how one may aggregate textual data when all available aggregation functions of the OLAP environment are based on arithmetic functions (sum, average…)?

1.1 Related Works

In order to analyse data extracted from XML documents, three types of propositions have been made. Firstly, were proposed the multidimensional analysis of documents within a standard OLAP framework [8, 9, 5, 17]. Nevertheless, all these propositions limit themselves to numerical indicators and do not allow content analysis of these documents. Secondly, some works have detailed adaptations of aggregation functions for XML data. An aggregation function for XML structures, XAggregation, has been proposed [18, 19]. And it has recently been followed by an adaptation of the Cube function for XML data [20]. These new functions allow the analysis of XML documents. But these operators are not text-oriented, thus, these operators do not allow the analysis of the textual content of XML documents. Thirdly, in order to answer more precisely to the problem of the analysis of the contents of text-rich XML documents, in [11], the authors describe a set of aggregation functions adapted to textual data inspired by text mining techniques. Unfortunately, the authors provide no detailed description, no formal specification and no implementation guidelines for their functions. Moreover, the authors describe their framework with an associated model based on an xFACT structure [10] but they do not specify how to handle textual indicators within the framework.

In conclusion, the integration of methods and analysis tools adapted to text-rich XML documents within the OLAP environment is still a promising issue.

[1] XML, Extended Markup Language, from http://www.w3.org/XML/.

1.2 Objectives and Contributions

Our goal is to provide an OLAP framework adapted for the analysis of text-rich XML document contents. We require a new approach for aggregating textual data. Our contribution for multidimensional document analysis is to provide an adapted framework composed of textual analysis indicators associated to compatible aggregation functions [12]. We thus revise measure concept to take into consideration textual measures. To allow analyses based on textual measures, textual data must be aggregated by an adapted aggregation function. Inspired by MAXIMUM_k that aggregates a set of numbers into the k highest values, we define a function that aggregates a set of text fragments into the k most representative terms of those fragments.

The rest of the paper is organised as follows: the following section defines our adapted constellation multidimensional model; section 3 specifies the adapted aggregation function and finally section 4 describes the implementation.

2 Multidimensional Conceptual Model

Traditional multidimensional models (see [16, 13] for recent surveys) allow the analysis of numerical indicators according to analysis axes. However, these models are not sufficient for performing OLAP analyses on text-rich documents. Our proposition is to extend these traditional models with textual analysis indicators. The model is a constellation [6, 13] composed of dimensions and facts.

Dimensions model analysis axes and are composed of a set of parameters which are organised into one or more hierarchies. Each *hierarchy* represents an analysis perspective along the axis. The *parameters* represent different levels according to which analysis data may be observed. The subject of analysis, namely a *fact*, is a conceptual grouping of measures. Traditionally these measures are numeric, thus this concept must be extended in order to cope with our issue of textual data.

2.1 Measures in the OLAP Environment

In order to consider specificities of documents the concept of measure is extended.

Definition: A *measure M* is defined by $M = (m, type, f_{AGG})$ where:
- m is the measure name;
- *type* is the measure type (numerical additive, non additive,…);
- f_{AGG} is the list of available aggregation functions for the measure.

f_{AGG} is used by manipulation languages (see for example [13]) to ensure multidimensional query consistency. Note that F_{AGG} is the list of all available aggregation functions of the OLAP system. The measure type conditions the list of compatible aggregation functions (F^T_{AGG}), thus for a measure type T ($F^T_{AGG} \subseteq F_{AGG}$). Amongst the compatible functions, the designer selects the aggregation functions that will be available and that will constitute the list f_{AGG} ($f_{AGG} \subseteq F^T_{AGG}$).

Different Types of Measures. Two types of measures are distinguished:

Numerical measures are exclusively composed of numbers and are either additive or semi-additive [4, 6]. With additive numerical measures, all classical aggregation functions operate. Semi-additive measures represent instant values, i.e. snapshots such as stock or temperature values and the *sum* function does not operate.

Textual measures are composed of textual data that are non numeric and non additive [4, 6]. Contents of textual measures may be words, bags of words, paragraphs or even complete documents. According to these different possibilities the aggregation process may differ, thus several types of textual measures are distinguished:

- A *raw textual measure* is a measure whose content corresponds to the textual content of a document for a document fragment (e.g. the content of a scientific article in XML format stripped of all the XML tags that structure it).
- An *elaborated textual measure* is a measure whose content is taken from a raw textual measure and has undergone a certain amount of processing. A textual measure such as a *keyword* measure is an elaborated textual measure. This kind of measure is obtained after applying processes on a raw textual measure such as withdrawing stop words and keeping the most significant ones regarding the document's context.

OLAP Aggregation Functions. According to the measure type, not all aggregation functions may be used for specifying analyses. Table 1 lists the compatibility between measure types (T) and available aggregation functions (F^{T}_{AGG}). Within OLAP environments several arithmetic aggregation functions are available:

- Additive function: SUM (the sum of values that have to be aggregated);
- Semi-additive functions: AVG (the average of the values), MIN and MAX (the minimal or the maximal values).
- Generic functions: COUNT (counting the number of instances to be aggregated) and LIST (the identity function that list all the values).

Table 1. the different types of measures and the possible aggregation functions.

Measure Type	Applicable Functions	Example
Numeric, Additive	Additive, Semi-Additive, Generic	A quantity of articles
Numeric, Semi-Additive	Semi-Additive, Generic	Temperature values
Textual, Raw	Generic	Content of an article
Textual, Keyword	Generic	Keywords of a fragment of a document

2.2 Example

A decision maker analyses a collection of scientific articles published by authors at a certain date (see **Fig.** 2 where graphic notations are inspired by [2]). The fact *ARTICLES* (subject) has three analysis indicators (measures): a numerical measure (*Accept_Rate*, the acceptance rate of the article), a raw textual measure (*Text*, the

whole textual content of the article) and a keyword elaborated textual measure (*Keyword*). The fact *ARTICLES* is linked to 2 dimensions: *AUTHORS* and *TIME*.

Within this example, the measure *Accept_Rate* is associated to AVG, MIN, MAX and LIST aggregation functions. The two other measures are associated only to the two generic aggregation functions (COUNT and LIST)

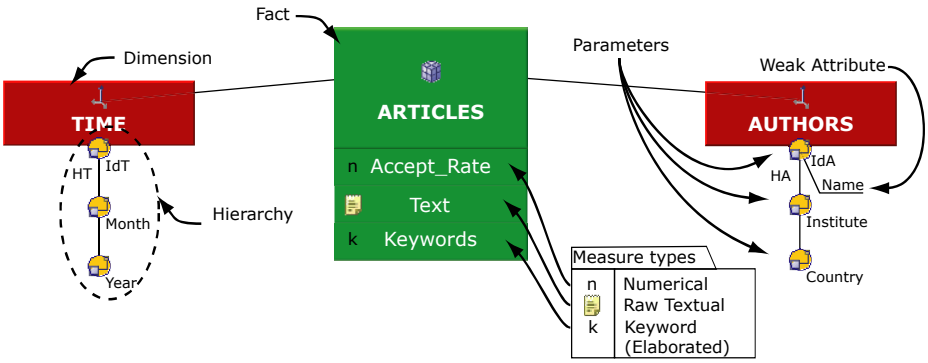

Fig. 2 Example of a constellation with textual measures for document content anlaysis.

3 Top_Kw Aggregation Function

Our objective is to provide adapted aggregation functions for textual measures (raw and elaborated). However, only generic aggregation functions operate on such measures, which is not sufficient for multidimensional analysis. Thus, we propose to extend the list of available functions dedicated to textual measures. In order to handle elaborated textual measures, more precisely keyword measures, we have previously defined an aggregation function inspired by the AVERAGE function. This function aggregates a set of keywords into a smaller set of more general keywords [12].

In the rest of this paper we focus on the aggregation of raw textual measures, e.g. the full text scientific articles in XML format.

Preliminary remark: note that raw textual measures are stripped from stop words (e.g. words like: a, an, the, these…).

3.1 Formal Specification

In order to summarise data from a raw textual measure, there is a need for aggregating its textual data. The objective of the aggregation function TOP_KW_k is to extract from a textual measure (composed of n words) a set of k most representative terms.

As in the arithmetic aggregation function MAX_k which extracts the k highest values of a set of numbers, this function extracts the k terms that have the most representativeness from a text represented through a set of n words.

$$Top\_Kw_k : \quad W^n \longrightarrow T^k \qquad \{w_1, \ldots, w_n\} \mapsto \langle t_1, \ldots, t_k \rangle \qquad \text{Eq. 1}$$

With as **input**: a text represented through a set of *n* words and as **output** an ordered subset of W^n composed of the *k* most representative terms. $T^k \subseteq W^n$, $\forall\ t_i \in T^k$, $t_i \in W^n$. Concretely, the function orders all the words that represent documents according to a weight assigned corresponding to its representativeness regarding the document and returns *k* first weighed terms.

3.2 Ordering Terms of Documents

In order to determine the *k* most representative terms, we adapted to the OLAP context well-known and well-mastered techniques from Information Retrieval (IR) [1]. IR uses functions to assign weights to terms within documents. Following this example, we shall use and adapt a function that weighs terms according to their representativness and thus allowing their ordering according to this representativness.

In information retrieval, it is necessary to know the representativness of a term compared to the whole collection of documents that contain this term. In the OLAP context, it is not necessary to know the representativness of terms according to the whole collection but rather according to the whole set of textual fragments that are to be aggregated with the function. The issue is to have to operate on a variable list that changes at each OLAP manipulation, instead of a fixed list (the collection).

3.3 Aggregation and Displaying within a Multidimensional Table

Context. The restitution of an OLAP analysis is done through multidimensional tables (see Fig. 1). Values are placed in cells c_{ij} that are at the intersection of i^{th} line and the j^{th} column. Each cell contains the aggregated values of the analysed measures.

Note that for each cell of a multidimensional table, the aggregation function is applied. Each cell represents a certain number of documents or fragments of documents. To each cell c_{ij} corresponds:

- A set of documents (or fragments of documents) D_{ij} composed of d_{ij} documents;
- A total number of terms n_{ij} that are in each of the d_{ij} documents.

Weight Calculus. Within each cell, weights are assigned to each one of the n_{ij} terms. In order to "rank" these terms we use a weighing function. We chose the *tf.idf* function that corresponds to the product of the representativness of a term within the document (*tf*: term frequency) with the inverse of its representativness within all available documents of the collection (*idf*: inverse document frequency). We have adapted this

function to our context, i.e. the *idf* is calculated only for the documents of the cell (not the whole collection). For each cell c_{ij} and each term t corresponds:

- A number of occurrences $n_{ij}(t)$ of the term t in the document of c_{ij}, i.e. D_{ij};
- A number of documents $d_{ij}(t)$ that contain the term t amongst the documents of c_{ij} ($d_{ij}(t) \leq d_{ij}$).

This gives us, for each term of the cell c_{ij} an adapted *tf.idf* function:

$$tf_{ij}(t) = \frac{n_{ij}(t)}{n_{ij}} \text{ and } idf_{ij}(t) = \log \frac{d_{ij} + 1}{d_{ij}(t)} \quad \text{Eq. 2}$$

$$\text{Thus the weight of the } t \text{ term is: } w_{ij}(t) = tf_{ij}(t) \times idf_{ij}(t) \quad \text{Eq. 3}$$

In the previous equations (see Eq. 2), $tf(t)$ is the number of times when the term t is in a fragment of text normalised by the total number of terms of that fragment of text. This reduces the bias introduced by very long fragments of text compared to small ones. The quantity $idf(t)$ is the inverse of the number of documents that contain the term t compared to the number of documents contained in the cell c_{ij}. The 1 added is to avoid obtaining a null *idf* if all the documents contain the term t. The use of a logarithm function is useful to smooth the curve thus reducing the consequences of large values compared to small ones. We invite the reader to consult [14] for a recent detailed analysis of the *tf.idf* function.

Restitution. The application of the *tf.idf* function (see Eq. 3) to a set of documents whose words have been extracted, allows obtaining a list of weighed terms.

Words are extracted from each document of the set D_{ij} and weights are assigned to each by applying the adapted *tf.idf* function. If a weighed term appears in several documents of D_{ij}, the weights are calculated separately and then added in order to increase the repsentativness of the term. Weighed terms are then ordered according to their weights: $L_{ij} = \langle t_1, \ldots, t_n \rangle \mid w_{ij}(t_1) > w_{ij}(t_2) > \ldots > w_{ij}(t_n)$.

The aggregation function TOP_KW$_n$ finally extracts the first n terms of the list L_{ij} with the highest weights and displays them into the corresponding c_{ij} cell of the multidimensional table.

3.4 Example

In the following example, the function TOP_KW is set to return the 2 most representative terms ($k = 2$). The analysis consists in the major keywords of scientific articles analysed according to their author and the year of publication (see **Fig. 3**). For this analysis, the aggregation function is applied to four groups of documents: one for each cell of the multidimensional table that correspond to each of the following braces: (*Au1*, *2005*), (*Au1*, *2006*), (*Au2*, *2005*) and (*Au2*, *2006*). Note that only 4 terms are represented.

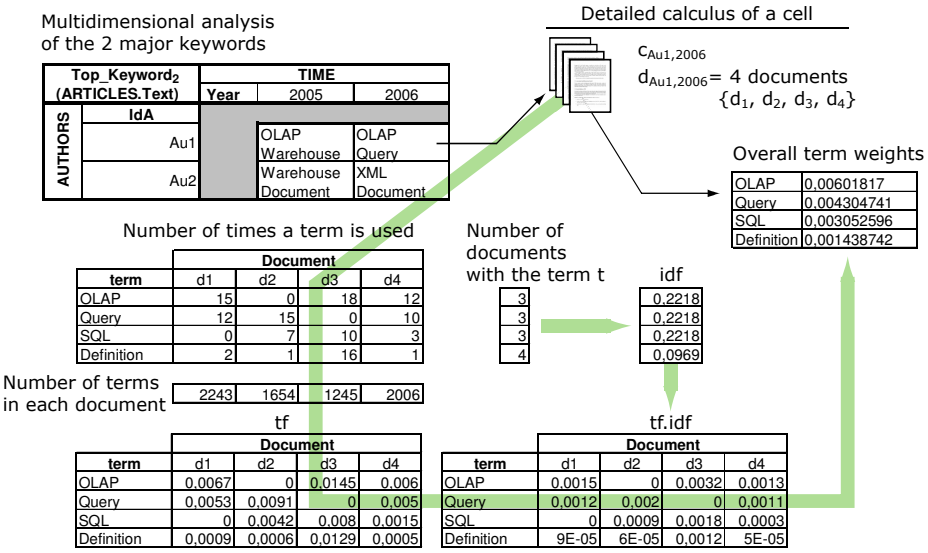

Fig. 3 Example of a detailed analysis using the function TOP_KW$_2$.

4 Implementation

The aggregation function has been implemented within an existing prototype [13]. This prototype is based on a relational environment (ROLAP). The multidimensional database is in an Oracle 10*g*2 RDBMS (with XMLDB to handle the document contents) and the high-end interface is a Java client.

4.1 Pre-Processing of document contents

The more a term is used in a document, the higher the weight of that term given by *tf.idf* function is. The problem is that very common words (named *stop words* in information retrieval) will have a very high weight although they are not the most representative terms. On a raw measure, the solution consists in removing from the aggregation process all stop words (e.g. a, an, the, as, at…).

4.2 Processing for the aggregation

The TOP_KW$_k$ aggregation function rests on the following algorithm. This algorithm takes as input the set of documents D_{ij} of a cell c_{ij}. `Weighed_Term_List` is a list of terms extracted from the D_{ij} documents and associated to a weight that correspond to the representativity of each term regarding the whole D_{ij} document set.

```
For Each c_ij cell Do
   Create a new empty Weighed_Term_List;
   For each document d of D_ij Do
      For each word of d Do
         If word is a Stop Word Then drop the word;
         Else Do
            Process weight of word;
            If word not in Weighed_Term_List Then
               Insert word and Insert weight
            Else
               Add new weight to already existing weight;
            End_If
         End_If
      End_For
   End_For
   Order Terms in Weighed_Term_List by decreasing weights
   Extract the first k terms to the cell c_ij
End_For
```

Note that some of the instructions of this algorithm may be removed if they are handled as pre-processing steps. Due to lack of space only some of the numerous optimisations will be briefly presented in the following subsection.

4.3 Performance issues

Aggregation functions may be optimised more or less easily according to their type (distributive, algebraic and holistic). Holistic aggregation functions are difficult to optimise [3, 7] as the aggregation process may not rely on intermediate result for processing the final result. Unfortunately, the modified *tf.idf* function used in the aggregation function renders it holistic. Nevertheless, a simple solution consists in using numerous materialised views for execution for speeding up runtime.

Another solution would consist in using a precalculated specific view that would provide intermediate results in order to speed up the computation of the term weights. For example, it would be possible to precalculate the *tf* part of the weighing formula for each term and document as this only requires parsing the document. However, the computation of the *idf* part would have to be done at runtime.

5 Conclusion

Within this paper, we have defined a new aggregation function for a text-rich document OLAP framework. It is possible to have a synthetic vision of document sets by extracting the *k* most representative keywords of each set. The aggregation function TOP_KEYWORD (TOP_KW$_k$ for short) rests on an adapted *tf.idf* weighing function. This weighing function allows ordering words of document sets or sets of fragments of documents and the aggregation function selects the first *k* words.

Several future works may be undertaken. Firstly, in information retrieval, there exists the notion of “relevance feedback” that consists in adding terms to a query to increase the reliability of the terms used in a query. In a similar way, relevance feedback could be used to add terms to those returned by the aggregation function. The final result would then correspond to a more precise set of terms. Secondly, there

exists several variants and even complete alternatives of the *tf.idf* function [14]. We think that a comparative study of these different weighing functions should be done in order to optimise the implementation of our aggregation function. Finally, in order to enrich our environment we are considering to define and implement other aggregation functions for textual data such as, for example, those stated in [11].

References

1. Baeza-Yates, R., Ribeiro-Neto B.: *Modern Information Retrieval*. Addison Wesley, 1999.
2. Golfarelli, M., Maio, D., Rizzi, S.: The Dimensional Fact Model: A Conceptual Model for Data Warehouses. invited paper, *IJCIS*, 7(2-3), pp. 215–247, 1998.
3. Gray, J., Bosworth, A., Layman, A., Pirahesh, H.: Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Total. *ICDE*, pp. 152–159, 1996.
4. Horner, J., Song, I-Y., Chen, P.P.: An analysis of additivity in OLAP systems. *DOLAP*, pp. 83–91, 2004.
5. Keith, S., Kaser, O., Lemire, D.: Analyzing Large Collections of Electronic Text Using OLAP. *APICS, Conf. in Mathematics, Statistics and Computer Science*, pp. 17–26 2005.
6. Kimball R.: The data warehouse toolkit, Ed. John Wiley and Sons, 1996, 2nd ed. 2003.
7. Lenz, H.J., Thalheim, B.: OLAP Databases and Aggregation Functions. *SSDBM 2001*, pp. 91–100, 2001.
8. McCabe C., Lee, J., Chowdhury, A., Grossman, D.A., Frieder O.: On the design and evaluation of a multi-dimensional approach to information retrieval. *SIGIR*, pp. 363–365, 2000.
9. Mothe J., Chrisment, C., Dousset, B., Alau, J.: DocCube: Multi-dimensional visualisation and exploration of large document sets. *JASIST*, 54(7), pp. 650–659, 2003.
10. Nassis, V., Rajugan, R., Dillon, T.S., Wenny Rahayu, J.: Conceptual Design of XML Document Warehouses. *DaWaK*, LNCS 3181, pp.1–14, 2004.
11. Park, B.K., Han, H., Song, I.Y.: XML-OLAP: A Multidimensional Analysis Framework for XML Warehouses. DaWaK, LNCS 3589, pp.32–42, 2005.
12. Ravat, F., Teste, O., Tournier, R. : OLAP Aggregation Function for Textual Data Warehouse. *ICEIS 2007*, Vol. DISI, pp. 151–156, 2007.
13. Ravat, F., Teste, O., Tournier, R., Zurfluh, G.: Algebraic and graphic languages for OLAP manipulations. ijDWM, 4(1), IDEA Group Publishing, pp. 17–46, 2007.
14. Robertson, S.: Understainding Inverse Document Frequency: On theoretical arguments for IDF. *Journal of Documentation*, 60(5), Emerald Publishing Group, p. 503–520, 2004.
15. Sullivan, D.: Document Warehousing and Text Mining. Wiley John & Sons, 2001.
16. Torlone, R.: Conceptual Multidimensional Models. Chapter 3 in *Multidimensional Databases: Problems and Solutions*, M. Rafanelli (ed.), Idea Group Inc., pp.69–90, 2003.
17. Tseng, F.S.C., Chou, A.Y.H.: The concept of document warehousing for multi-dimensional modeling of textual-based business intelligence. J. DSS, 42(2), pp. 727–744, 2006.
18. Wang, H., Li, J., He, Z., Gao, H.: Xaggregation: Flexible Aggregation of XML Data. *WAIM*, LNCS 2762, pp. 104–115, 2003.
19. Wang, H., Li, J., He, Z., Gao, H.: OLAP for XML Data. *CIT*, pp. 233–237, 2005.
20. Wiwatwattana, N., Jagadish, H.V., Lakshmanan, L.V.S., Srivastava, D.: X^3: A Cube Operator for XML OLAP. *ICDE*, pp. 916–925, 2007.