An Implementation of Exploratory OLAP System Based on Prasad's Approach

Géza Molnár
Department of Computer Science
Corvinus University of Budapest
geza_molnar@freemail.hu

***Abstract*—Unstructured and semi-structured data stored in data warehouses (e.g. comments, descriptions) are often not processed or ignored. As a result, valuable information may be lost, although there are several ways to handle these mostly textual data. Once this type of information is extracted, the result can be further investigated; amongst others, it can be incorporated into existing OLAP analysis. Technologies, more specifically frameworks, that allow the creation of a multidimensional schema from unstructured data of data warehouse are called exploratory OLAP. This article discusses a possible implementation of exploratory OLAP system based on Prasad's model [1].**

Keywords—exploratory OLAP, text analysis, star schema

I. INTRODUCTION

. Business reports are often based on a special data structure, so-called OLAP (Online Analytical Processing) cube. OLAP is a computational method that allows users to efficiently query and analyze data from a variety of perspectives. In another approach, it is a multidimensional (MD) schema. It can be queried with MDX (Multidimensional Expressions) statements [2].

OLAP technologies work well for adequately structured data. However, it can be a challenge to work with unstructured or semi-structured data. For example, creating a report on a product campaign, it needs to combine structured information (e.g., product sales data) and unstructured information (e.g., user reviews).

Exploratory OLAP systems provide an answer to this challenge. They work on different principles, but they have in common that they all create a semantic layer in the data warehouse / BI system. This layer has several advantages; amongst others, it allows interpreting non-structured data, as well.

II. RELATED WORKS

The literature is not uniform in the conceptual framework and implementation principles of exploratory OLAP. The best-known approaches are from Abello [3], Prasad [1] and Ibragimov [4].

Abello's exploratory OLAP concept uses semantic web technologies, namely, ontologies. They are frequently used for knowledge representation [5].

In Abello's system, there is a mapping between the source data and an ontology, so-called reference ontology. The mapping makes it possible to explore Functional Dependencies (FD) and multidimensional (MD) identifiers. The purpose of the latter is to identify the facts, and so the MD scheme can be created (Fig. *1*).

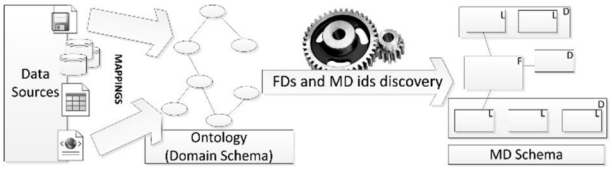

Fig. 1. Abello's model

Prasad suggested another approach. His solution is based on text mining and analysis techniques (bag-of-words approach [6]) that can be used to integrate structured and unstructured data into the data warehouse. The solution consists of text analysis (statistical and semantic analysis), identification of key terms, and creation of text tags (text tagging). Its system's main components are shown in the following figure (Fig. *2*).

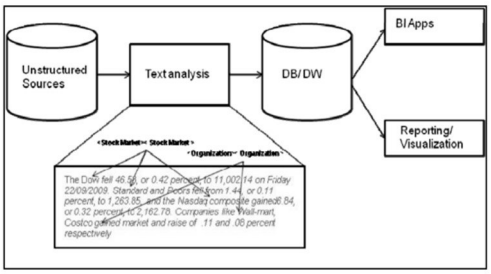

Fig. 2. Prasad's model

First, text analysis/preprocessing procedures are applied for the text extracted from unstructured sources. Then the results data will be loaded into specific tables of the database/data warehouse. The data model is star-schema (Fig. 3), so it is suitable for creating OLAP cubes and reports.

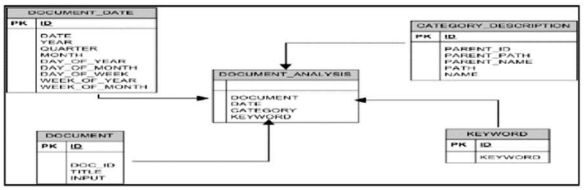

Fig. 3. Star schema

The architecture proposed by Ibragimov works with web data stored in a so-called RDF (Resource Description Framework) format. To describe things, RDF uses triplets in the form of an object-subject-statement [7]. Since it is a computer-readable and XML-based language, it has become a standard model for web data exchange. The RDF structure can be queried using special query languages such as SPARQL. Ibragimov's basic idea is to create a mapping between

SPARQL querying an RDF schema and MDX statements querying an OLAP schema. The proposed system consists of four modules (Fig. *4*):

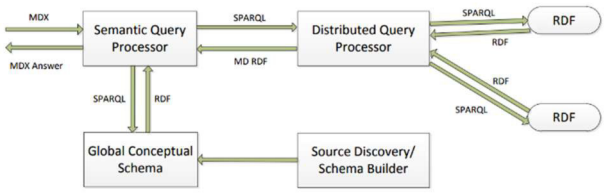

Fig. 4. Ibragimov's model

- Global Conceptual Schema - stores information about the OLAP cube
- Semantic Query Processor - Generates SPARQL queries from MDX queries
- Distributed Query Processor - queries endpoints, collects data
- Source Discovery Schema Builder - keeps in touch with users as the schema is created.

Each of the exploratory OLAP framework proposals presented above handles text data processing differently. The following table compares the three approaches.

TABLE 1 Comparison of Exploratory OLAP systems

Point of comparison	Abello	Ibragimov	Prasad
Data sources	Any	Linked Open Data	Any
Tools, technologies	Ontologies	RDF, MDX, SPARQL	Text analysis, XML
Biggest challenge	Ontologie →MD	MDX → SPARQL	Text analysis
Is there a prototype plan?	No	Yes	Yes
Output location	OLAP cube	OLAP cube	Data Warehouse
Output format	OLAP schema	OLAP schema	Star schema

The next part presents the creation of a prototype based on Prasad's idea. As can be seen from the previous table, the output of the prototype is not an OLAP cube but a star schema. It is not a problem in practice, since the OLAP cube can be built easily on the star schema.

III. Implementation of Exploratory OLAP System Based on Prasad's Model

Creating the prototype consists of four steps. The first one is the data preparation, which, on the one hand, means extracting the necessary data from the source system and, on the other hand, creating an initial list of keywords related to the topic. It is followed by a text processing, which results in the data needed to create the structure proposed by Prasad. The third step is the star schema creation, as shown in Fig. 3. The last step is the creation of the OLAP cube.

The following software was used during the implementation:

- Service Desk (ticketing) program to extract data from source systems [8]
- Python 3.7 (Jupyter, Anaconda) for text processing
- MS SQL Server 2017 Express, SQL Server Management Studio and SQL Server Import and Export Wizard for star schema creation
- Power BI Desktop for emulating the OLAP-cube.

The prototype uses a ticketing (helpdesk) system's textual data, which arise when users report various problems ("incidents").

A. Data Preparation

The data source contains data of incidents from the ticketing system covering a six months interval. Each report has two textual data, namely error summary and error description. Besides, the ticket creation date (Open_Date) and the ticket identification number (Ticket Number) are also essential. For privacy reasons, the Ticket Number has been replaced with a counter-type field.

The first step of the data preparation is the data export (Ticket Number, Open_Date, Summary, Description) from the ticketing system using the filter and transformation as mentioned above. For the sake of simplicity, during the export CSV format was applied. It is appropriate either for the text analyses, or the SQL-based database/data warehouse systems as an input format. It is different from the XML format suggested by Prasad, but this is irrelevant for the later steps. Since the ticketing system contains only consistent data, so minimal data cleaning (e.g. data deduplication) was needed. The latter can occur if, e.g. multiple users report the same problem in the same way at the same time.

The second step is to define the keywords. These are words that are typically associated with error reports and can be used to categorize the incidents. The easiest way to create a list of keywords manually based on the common error descriptions. A better way is to look for the most common words (possibly word combinations) in the text data source and then select the relevant ones (e.g. server, network, printer). It is easily accomplished with a Python-script, which

- opens the source data
- eliminates unnecessary spaces
- converts everything to lowercase
- breaks the text into words
- omits stopwords and words not found in the English dictionary
- keeps only nouns
- finally, filter for words that occur at least 100 times

The resulting list has been filtered for words that can be associated with the ticketing system. It is not necessary to maintain a list of keywords for each data export. It is enough to update it a few times a year.

In case of a keyword list, also the CSV file format is preferred. Consequently, after the data preparation, two CSV files have been created with the following structure: INCIDENTS.CSV (ID, OPEN_DATE, SUMMARY, DESCRIPTION) and KEYWORDS.CSV (ID, KEYWORD). Currently, the first CSV file contains about 3000, and the second one has 50 records. Later, the amount of data will be increased.

B. *Text Analysis*

The text analysis has been performed with another Python-script. It implements the TextRank algorithm [9]. Its basic idea comes from the PageRank algorithm [10], which was initially designed to rank websites. According to this, a website's ranking position depends mostly on how many other "important" websites link to it. The TextRank algorithm was implemented by using the description of Xu Liang [11].

The PageRank algorithm assigns a directed graph to web pages. The graph's nodes are web pages, and the edges describe the links between them. Weights can also be given to each node according to the following formula:

$$S(V_i) = (1-d) + d * \sum_{j \in In(v_i)} \frac{1}{|Out(V_j)|} S(V_j) \qquad (1)$$

where

- S(Vi) – the value of each weight,
- d – dumping factor
- In(Vi) – set of connected nodes (incoming edges)
- Out(Vi) – set of connected nodes (starting edges)

The graph mathematically can be represented by an n * n matrix, where n is the number of nodes, and the jth element of the ith row equal to 1 there is an edge from the ith node to the jth node. Otherwise, the given element is 0.

The initial values of PageRank are obtained by normalizing the matrix's columns and multiplied by the vector containing the weights of each node. It is followed by an iteration in which the PageRank values are recursively modified at each step. It results in a series of random variables (irreducible, aperiodic Markov chain). Due to the damping factor d, it has a boundary distribution.

The PageRank algorithm is one of the essential elements of the Google search engine. PageRank values can also be considered as the probability of a page being clicked, and the graph contains pages that can be affected during a browse.

The TextRank algorithm works with sentences instead of web pages. Here, the jump from one website to another corresponds to the similarity of two sentences. The similarity data are stored in a matrix. The main steps of the algorithm are illustrated in the following figure:

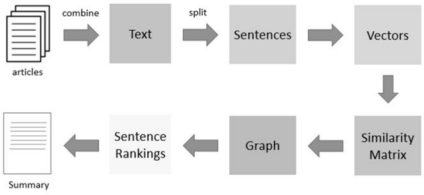

Fig. 5. TextRank algorithm

The TextRank algorithm wraps the sentences into units (words) and assigns to them grammatically appropriate categories as a label. The method is called POS tagging (Part of speech tagging), which can result in tags assigned to words, such as NOUN (noun), PROPN (proper name), and VERB (verb) [12].

It is enough to consider only nouns and proper names when searching for keywords. The keywords will be the ones with the highest weight.

The script that implements the algorithm begins by importing the necessary packages, then sets the values of initial data (number of iteration steps, attenuation factor, convergence threshold). The next step is the data processing, during which the information extracted from the text data will be brought into the following structure:

DOCUMENT (DOC_ID, DATE, TITLE, INPUT, KEYWORD, CAT)

where

- DOC_ID - the document identifier
- DATE - the date when the ticket was opened
- TITLE - the title of the document (the original summary field are included here)
- INPUT - the line that contains the keyword detected by the algorithm
- KEYWORD - the keyword discovered by the algorithm
- CAT - the keyword category (label), which can be NN: noun, NNP: proper name, VBN: verb, XX: other

The data processing algorithm's pseudocode is the following:

```
document = empty list
keyword_list = open(keywords.csv)
text = open(text.csv)
loop for incidents
    loop for sentences
        analyze
        extract keywords
        compare keywords with keyword_list
        if found common words then
            loop for common words
                create a text tag for the word
                insert a new record into the document.csv
            end of loop
        end of selection
    end of loop
end of loop
OUT: document.csv
```

It can be seen that the algorithm takes into account only the sentences containing keywords.

C. *Star Schema*

For the star schema creation, the first step is to load the results of the text analysis to the database/data warehouse. The easiest way to do this is to use the SQL Server Import and Export Wizard, which allows loading a CSV file contents into a database.

The second step is to implement the structure proposed by Prasad. It can be easily solved by creating SQL views. The relationships between the tables can be defined either here or at the Power BI data model creation. Interestingly, the fact table (DOCUMENT_ANALYSIS) does not contain measures, only date and dimension identifiers. It is important to note that both the data processing and the data loading can be automated and scheduled.

Automation will be implemented later as follows:

- The data processing script can be run using the Windows built-in task scheduler (Windows Task Scheduler)

- CSV files can be loaded into the database with the Microsoft's best-known ETL tool (SQL Server Integration Services)
- Data loading can be scheduled by the SQL Server Job Agent.

D. OLAP Cube

The OLAP cube is usually created with a server-side tool, such as SQL Server Analysis Services. However, the task can also be solved more straightforwardly - from the client side - because the data model generated using the Power BI Desktop program is compatible with it. It means that the model can be tested in Power BI and then later opened in Analysis Services without any change.

With the help of the SQL views, a star schema-based data model can be created. Power BI detects and automatically creates connections between tables, if necessary. The model ("OLAP cube") can then be tested by switching to a visual view and making, e.g. the following dashboard:

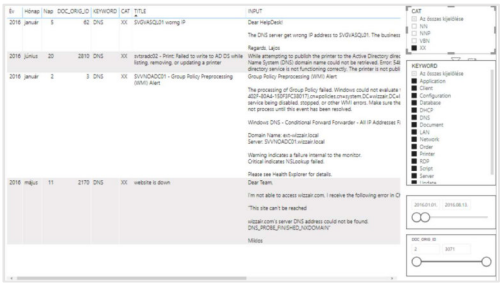

Fig. 6. Power BI Dashboard

The dashboard shows the text of each ticket description, its keywords and categories, and the date the ticket was created in a tabular layout. The list can be filtered by category, keyword, date and document ID. The results show that the process still needs to be optimized with running it many more times using different settings to achieve the best possible outcome.

IV. CONCLUSION

Textual data in databases and data warehouses can hide much valuable information. The role of exploratory OLAP systems is to uncover this hidden information. This article describes how to create a prototype of such a system. The basic idea comes from Prasad, but the implementation is slightly different. The applied algorithm creates the necessary star scheme, but its efficiency could still be improved. On the one hand, during the execution of the algorithm, it would be more effective to look at the semantic similarity of keywords instead of matching them. On the other hand, in addition to keywords, multi-word key expressions could be considered, as well.

Future work can include text mining-based taxonomy development integration [13].

ACKNOWLEDGEMENT

Funding Project No. NKFIH-869-10/2019 and its continuation in 2020 have been implemented with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the Tématerületi Kiválósági Program funding scheme.

REFERENCES

[1] K. S. N. Prasad, "Text Analytics to Data Warehousing," International Journal on Computer Science and Engineering,, 2010.

[2] E. F. C. S. B. a. S. C. T. Codd, "Providing OLAP to User-Analysts: An IT Mandate," *Computer Science,* pp. 3-5, 1993.

[3] A. Abelló, "Using Semantic Web Technologies for Exploratory OLAP: A Survey," IEEE, 2015.

[4] K. H. T. B. P. E. Z. Dilshod Ibragimov, "Towards Exploratory OLAP Over Linked Open Data – A Case Study," International Workshop on Business Intelligence for the Real-Time Enterprise, 2015.

[5] T. Gruber, "A Translation Approach to Portable Ontology Specification," Knowledge acquisition, Vol. 5 No. 2, 1993, pp. 199-200.

[6] B. K. a. M. T. Matthew Gentzkow, "Text as Data," *Journal of Economic Literature,* vol. 57(3, pp. 537-539, 2019.

[7] H. L. a. R. S. K. Selc ¸uk Candan, "Resource Description Framework: Metadata and Its Applications," *ACM SIGKDD Explorations Newsletter,* vol. 3, pp. 8-12, 2001.

[8] "CA Service Desk Manager," [Online]. Available: https://www.ca.com/us/products/ca-service-desk-manager.html.

[9] P. T. Rada Mihalcea, "TextRank: Bringing Order into Text," in *Association for Computational Linguistics*, Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, 2004.

[10] P. P. Anuj Joshi, "Google Page Rank Algorithm and It's Updates," *International Journal of Management, Technology And Engineerin,* pp. 1108-1110, 2018.

[11] X. Liang, február 2018. [Online]. Available: https://towardsdatascience.com/textrank-for-keyword-extraction-by-python-c0bae21bcec0.

[12] F. M. H. U. Khan, "Comparison of different POS Tagging Techniques (n-gram, HMM and Brill's tagger) for Bangla," in *Springer, Dordrecht*, Advances and Innovations in Systems, Computing Sciences and Software Engineering, 2007.

[13] A. &. G. S. Ko, "A Research Review and Taxonomy Development for Decision Support and Business Analytics Using Semantic Text Mining.," *International Journal of Information Technology & Decision Making (IJITDM),* no. 19(01), pp. 97-126, 2020.

[14] P. Joshi, 2018. [Online]. Available: https://www.analyticsvidhya.com/blog/2018/11/introduction-text-summarization-textrank-python/..