ONTOCUBO: Cube-based Ontology Construction and Exploration

Carlos Garcia-Alvarado
Pivotal Inc.
San Mateo, CA 95134, USA

Carlos Ordonez
University of Houston
Dept. of Computer Science
Houston, TX 77204, USA

ABSTRACT

One of the major challenges of big data analytics is the diverse information content on the Internet, which has no pre-defined structure or classification. This is in contrast to the well-designed structure of a database specified on an ER model. In the information retrieval community ontologies are a standard mechanism to understand interrelationships and structure of documents. With such motivation in mind, we present a system that enables data management and querying of documents based on ontologies, leveraging DBMS functionality. In this paper we present ONTOCUBO, a novel system based on our research for text summarization using ontologies and automatic extraction of concepts for building ontologies using Online Analytical Processing (OLAP) Cubes. ONTOCUBO is a database-centric approach that excels in its performance, due to an SQL-based single pass summarization phase through the original data set that computes values such as, keyword frequency, standard deviation, and lift. This approach is complemented with a set of User-Defined-Function-based algorithms that analyze the summarization results for concepts and their interrelationships. Finally, we show in detail our application that extracts and builds an ontology, but also allows the concept summarizations and allows domain experts to explore and modify the resulting ontology.

1. INTRODUCTION

Several artificial intelligence, information retrieval, and natural language processing systems that are used across semantic web applications have relied on a declarative representation of knowledge (ontologies) for effective querying and exploration of a corpus [5]. Despite the need of these systems to have an ontology, this labor-intensive task is normally tailored for a specific domain of knowledge and set of domain experts, making it cost-prohibitive to build a new ontology every time that the domain is extended or modified. Automatic ontology building arises as an alternative for easing this difficult task. However, automatically finding the concepts and their relationships is especially challenging in the Internet content without a known structure or classification. Concept extraction has been approached through natural language processing [6], clustering [2], singular value decomposition (SVD), and statistical techniques. While normal applications of Online Analytical Processing (OLAP) include business reports and financial analysis [1], in this work we show that OLAP can also be useful in concept extraction for building ontologies and computing summarizations. More specifically, we show that cuboids of an OLAP Cube can be used to efficiently obtain classes and their relationships. In this case, the data is represented by the collection of keywords and documents, while the level of aggregation allows us to obtain various combinations of these keywords to generate concepts. These concepts are now filtered and pruned to compute ontologies. ONTOCUBO is a novel system inside database management systems (DBMS), extended from our research shown in [3, 4], in which we integrated both works and refined the resulting ontologies. A DBMS is used due to the support for computing efficient aggregations with SQL queries and the ability to adapt, in an efficient manner, the cube operator.

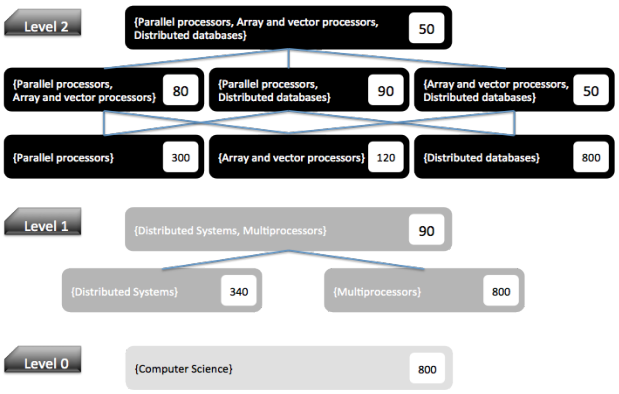

Figure 1: OLAP Cube with Concepts.

2. DEFINITIONS

Let C be a collection, or corpus, of n documents $C = \{d_1, d_2, \ldots, d_n\}$, where each document d_i is composed of a set of keywords $\{k_1, k_2, \ldots\}$. In addition, let x_i be a frequency vector of all the different keywords in C for each document d_i. Every concept that is part of an ontology is defined as a class, and the relation between classes is mod-

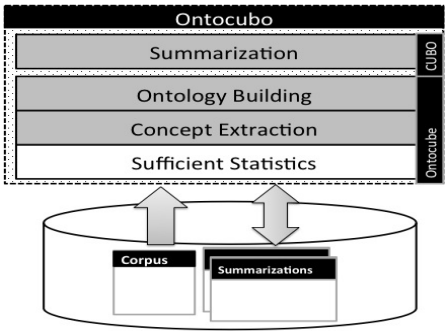

Figure 2: ONTOCUBO Architecture.

eled as a taxonomy of "is a" or "has a" relationships. An ontology $\mathcal{O}$, on the other hand, is a tree-like structure where the nodes represent dimensions (set of concepts) and the branches model the relationships between them.

Inside the DBMS, C is stored in a vertical list format. This list is stored in a distance table $D(i, k, p)$ that contains the document id, keyword, and position of a keyword in the d_i document. A summarization table stores the x_i vector of every document in a vertical format too. The summarization table is defined as table $S(i, k, f)$, where S contains the document id, the keyword, and the keyword frequency f. The size of the OLAP data cubes that generates the classes (concepts of different keyword lengths) is $2^{\hat{k}} - 1$ cuboids, where $\hat{k}$ is the number of most frequent keywords selected from $\sum_i^n x_i$. A cuboid (or depth in the lattice) l of the lattice is denoted by $\binom{\hat{k}}{l}$, such that $l \leq \hat{k}$. Finally, these classes are validated with the computation of correlation and lift using sufficient statistics (represented by n , L, Q). Let L be an additional set containing the total sum of all the different keywords in the collection ($L = \sum_{i=1}^n x_i$). In addition, let Q be a lower triangular matrix containing the squared measures between the keywords ($Q = \sum_{i=1}^n x_i x_i^t$. Moreover, lift λ is computed as $\lambda = \frac{n_{k,k'}}{Max(n_k, n_{k'})}$, where n_k and $n_{k'}$ are the number of documents containing keywords k and k', respectively. The final step of the computation is the aggregation of measurements by exploiting the ontology generated. The array of OLAP cubes is exemplified in Figure 1.

3. ONTOCUBO

ONTOCUBO is the result of three phases as shown in Figure 3: concept extraction, ontology building, and ontology summarization. The first phase extracts the most important concepts by computing the sufficient statistics from table S, and computing the frequency of the concepts of size l from table D. The second phase takes the computed concepts from the previous phase and generates the final ontology based on the measurements obtained with the sufficient statistics, the frequencies computed with OLAP, and the hypernyms (semantic relation of the type-of form) given by an existing conceptualization [8]. The third phase takes the extracted concepts and generates OLAP cubes with the computed measurements. In the following subsections, we present the basic steps for achieving concept extraction, ontology building, and the final OLAP Cube summarization of measurements.

3.1 Document Corpus Preprocessing

The concept extraction, which is a one-time preprocessing step, focuses on computing the frequency of the keywords and generates combinations between the most frequent ones. In order to find valid concepts, the following questions must be answered: (1) Do $k_1, k_2, \ldots$ appear in the same documents frequently? (2) Are $k_1, k_2, \ldots$ close to each other? (3) How often do $k_1, k_2, \ldots$ appear together within the corpus? (4) How often is a k found in a d_i but not in d_j?

These questions are answered by computing statistical measurements (in a single pass through S) such as the correlation between a pair of keywords, the distance between a pair of keywords, and the ratio between the number of documents containing a pair of keywords and the maximum of their frequency (lift). The algorithm for obtaining such classes is as follows in a single scan on D: (1) Obtain all available classes by pairing all keywords $\hat{k}$ within each document. (2) For each class, determine the frequency of occurrence and average position gap within the corpus. (3) Filter out all classes whose keywords have a position gap greater than a user-defined threshold, T. (4) Filter out all classes whose frequency, correlation, or lift does not meet user-defined thresholds for all the pairs of keywords in a class.

3.2 Ontology Building

The second phase is ontology building, which takes a set of rules based on the correlation, lift and frequencies computed in the previous step and decides the relation between the concept in the form of a "is a" or "has a" relation.

Initially, there is a need to determine if there is a relationship of any type between two classes. In order to do so, each class is paired with all the other classes within the class pool. Let each set of two classes be a relationship if for every possible pair between all the keywords of the concept there is a high correlation and lift. For example, suppose we are looking for a relationship between the classes $\{k_1, k_2\}$, $\{k_3, k_4\}$. In order to find out if there is a relationship, we need to obtain the correlation and lift for the following set of keywords: $\{k_1, k_3\}$, $\{k_1, k_4\}$, $\{k_2, k_3\}$, and $\{k_2, k_4\}$. Pairing them would allow us to observe how closely related these two classes are to one another. A high correlation and a high lift from any one of these four subsets would confirm that this is a valid relationship. Otherwise, if these four subsets do not show a connection, we discard this relationship.

The parent of a class is determined with a heuristic based on the frequency of appearance of a pair of classes. We consider a class A be the parent of a class B if the following criteria is satisfied: (1) The number of documents containing class A minus the number of documents containing class B is less than a user threshold θ (e.g. 10%) of total documents. (2) The number of classes containing keywords in class A is greater than the number of classes containing keywords in class B.

If a hypernym is a available for any of the classes, the class is substituted by the corresponding hypernym. Moreover, during the pairing, all the classes substituted for the same hypernym are pruned. The rationale behind this step is to limit the number of instances that are identified as valid classes. In addition to this, if we discover that several classes that contained the same hypernyms have been assigned differently, we select the relationship with more incidences. If there is a tie, we take one of the relationships arbitrarily.

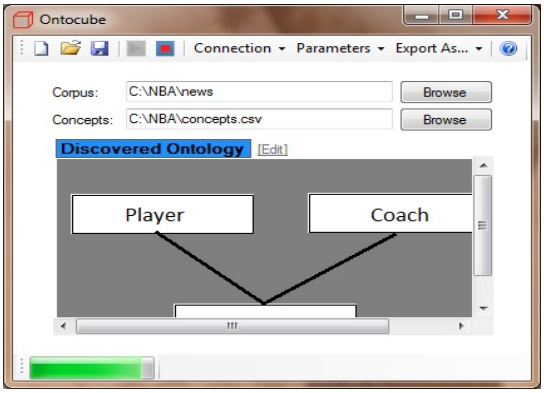

Figure 3: ONTOCUBO Building.

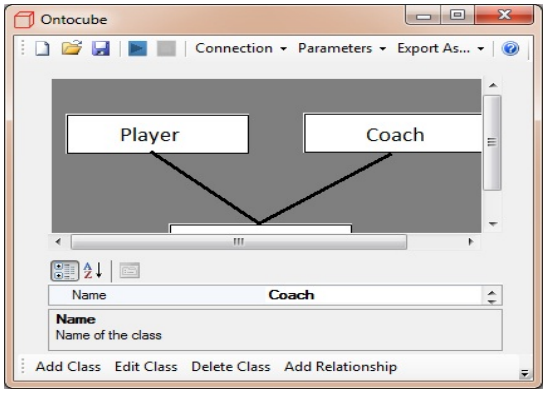

Figure 4: ONTOCUBO Exploration.

3.3 Ontology Exploration

CUBO is an array of data cubes that contain only the existing dimensions within the fact table D given a set of dimensions (concepts). However, the notion of CUBO is based on the premise that a set of documents does not contain all the dimensions in every document. Thus, only a set of "on-demand" computations of the dimensions are required for every document. Moreover, our algorithm takes advantage of this property when computing the aggregation on superior levels in the hierarchy. In other words, CUBO has a lazy policy that waits to perform the computations until it is absolutely necessary. Furthermore, the algorithm avoids redundant computations by focusing only on storing those dimensions that contain attributes to aggregate. If the entire data cube is desired, a post-processing phase can be executed to compute all the missing combinations.

The algorithm for obtaining the CUBO given an ontology $\mathcal{O}$, input corpus D, dimensions and a maximum ontology depth is shown in [4]. The result is temporarily stored in R that contains a level, combination of classes (combo) and the aggregation measurements. Furthermore, the entire computation of all the data cubes is computed through a single scan over the filtered data set. This additional scan can be avoided if the desired measurement is already contained from the original ontology generation. CUBO is composed of three major steps: load ontology (which is computer as part of ONTOCUBE), computation of on-demand combinations and the data cube aggregation. The final step of our algorithm stores the resulting data cubes into a relational table.

4. SYSTEM OVERVIEW

ONTOCUBO is a standalone system developed entirely in C# as two modules: a thin client that uses ODBC to connect to the DBMS and call plain SQL queries, and User-defined functions (UDFs) deployed inside the DBMS. This is also shown in Figure 3, in which the tasks performed with SQL queries are contained in white boxes, while the tasks performed in user-defined functions are contained in shaded boxes.

The computation of the sufficient statistics (stored for later user in a new table) is executed with an efficient SQL query of the form:

```
SELECT S1.kid as k1, S2.kid as k2
 , sum(S1.f) as L1
 , sum(S2.f) as L2
 , sum(S1.f*S1.f) as Q1
 , sum(S2.f*S2.f) as Q2
 , sum(S1.f*S2.f) as Q12
FROM S as S1, S as S2
WHERE S1.kid<=S2.kid AND S1.i=S2.i
GROUP BY S1.kid, S2.kid
```

Notice that the keyword was replaced by a surrogate that allows an efficient computation. In addition to this, indexes were added in the surrogate keys in order to obtain an efficient join. Also notice that this query would compute the sufficient statistics for the whole collection C, and this should be limited for the most frequent keywords $\hat{k}$.

The computation of the OLAP cuboids and resulting ontologies are then obtained inside a table-valued function (which is a type of UDF that allows returning a table) by calling:

```
SELECT * FROM ONTOCUBE('D_table',2,10,2,0.10)
```

Where the first parameter represents the distance table, the second parameter has the cuboid(s) l, the third parameter contains the top-k keywords for exploring $\hat{k}$, the fourth parameter represents the maximum gap between keywords T, and the last parameter is the parent class validation criterion θ.

In Figure 4, the main screen of ONTOCUBO is shown. Our system allows the user to select a location in the file system containing the corpus to import. In addition to this, the application takes as possible input a file containing an existing conceptualization; otherwise WordNet 2.1 is used [7]. Another setting needs to be provided to the program, such as the connection details for the DBMS and the tuning parameters l (a full lattice is also possible but the system is constrained by the memory allocated to the DBMS), T, and θ during the ontology building. Once all the parameters have been set, the user can start the automatic concept extraction and ontology building. Progress of the execution is shown in the status bar of ONTOCUBO, and the current ontology is shown to the user "on-the-fly". In Figure 4, the domain expert has the option of exploring and modifying the resulting ontology directly in our application, which includes adding, removing, or modifying an existing class; as well as adding and removing an existing relationship. Our system also allows exporting the resulting ontology as RDF or OWL files. The last step of our system allows building with the resulting ontology an OLAP cube with the desired

measurements. As such, a CUBO is built through a single pass on the original data set, via a TVF, unless the required summarizations were already computed as part of the ontology computation.

Our application has been tested, as shown in [3], by using data sets of sport news. The main advantage of ONTOCUBO is that the minimum number of parameters that are required to be tuned allows our system to be used in a variety of data sets. Furthermore, our single pass sufficient statistics computations, in-main memory OLAP computation, and rules allow the building of an ontology in an efficient manner.

5. SYSTEM DEMONSTRATION

Our demonstration will show how it is possible to obtain automatically generated ontologies from medium size document collections, including digital libraries, with OLAP Cubes and show the resulting CUBO. A local computer will show how a large data set, remotely available, is explored efficiently and generates ontologies without human intervention. All the algorithms (sufficient statistics, summarizations, measurements, and generated pairs) will be available as UDFs. In addition, the user will have the opportunity to explore every partial result of the entire ontology generation. The data set and the algorithms will be loaded in a remote machine running an instance of a commercial relational database system on a computer with 32 bit CPU, 4 GB RAM and 1 TB on disk. Resulting tables will be transferred to a local computer using ODBC. The user will analyze text data sets of different sizes, such as dbpedia ($\sim$ 70 GBs), Reuters 21578 ($\sim$ 20 MBs), and a data set of sports news articles ($\sim$ 1 GB). These articles are already preprocessed and stored in the format defined in Section 2.

5.1 Demonstration Goals

Our demonstration aims to spur the discussion of automatic ontology generators, the research challenges (e.g. OLAP), and the possibility of using the SQL language and User-Defined Functions for ontology extraction and OLAP Cube summarizations. The main contribution of our work is that we bring together the exploration power of OLAP to extract ontologies in a new manner. In this paper, we proposed an extension to ONTOCUBE and CUBO, two novel proposals that exploits OLAP techniques for building ontologies and compute summarizations inside a relational database management system. The core of the proposal is a system that relies on obtaining in an efficient manner the sufficient statistics with a one-pass SQL query for computing the correlation and lift from pairs of keywords from a corpus. Moreover, an efficient in-main-memory user-defined function OLAP algorithm is exploited to efficiently generate cuboids (or the entire lattice is desired) of sets of keywords and are verified if they are valid classes. Later, an ontology is then built and refined by using hypernyms support, which showed to be important for avoiding returning ontologies with a large number of instances. We found that OLAP Cubes are a suitable candidate for this type of application. This is due to the fact that the dimensionality limitation that accompanies OLAP when generating the lattice is never met, because we rarely need to generate the entire lattice (we mostly have concepts with less than 4 keywords which are represented by cuboids with depth less than 3).

Thus in this demonstration we will emphasize the following points: (1) It is possible analyze efficiently large text data sets within a relational database management system. (2) OLAP Cubes can be used to gather the required summarizations. (3) Understand how multiple heuristics can be applied for generating an ontology in an automated fashion. (4) Understand the in-memory processing for OLAP Cubes using User-Defined Functions.

5.2 Demonstration Scenarios

The user will have the opportunity to navigate the original data sets through a graphical user interface. After selecting the desired text data set, the user will have the opportunity of choosing if the ontology is going to generated in an automated fashion from the entire data set or from a sample of keywords. In addition, the user has the capability of assigning a value to several parameters, such as: the number of keywords to pair $\hat{h}$, position gap between keywords T, keyword frequency, correlation value, lift and the parent class threshold. The user will also have the capability of adding new hypernym for any of the classes.

Once the desired parameters have been set, the user will observe how the ontology is generated online and explore the early results in the GUI. It is also possible to explore the intermediate and final results by querying the resulting tables stored in the server.

With the final result, the user will have the opportunity of resolving conflicts or exporting the results as RDF or OWL files. Additionally, the user will be capable of querying the resulting ontology and performing additional analysis of the results stored in tables. Finally, the user will be able to compute the CUBO with the desired measurements. For example the frequency of the concepts in a document, frequency average of the concepts in document, or average correlation among the involved concepts, among many others.

6. REFERENCES

[1] Z. Chen, C. Ordonez, and C. Garcia-Alvarado. Fast and dynamic OLAP exploration using UDFs. In *SIGMOD*, pages 1087–1090, 2009.

[2] N. Choi, I.Y. Song, and H. Han. A survey on ontology mapping. *ACM Sigmod Record*, 35(3):34–41, 2006.

[3] C. Garcia-Alvarado, Z. Chen, and C. Ordonez. ONTOCUBE: efficient ontology extraction using OLAP cubes. In *Proc. of ACM CIKM*, pages 2429–2432, 2011.

[4] C. Garcia-Alvarado and C. Ordonez. Query Processing on Cubes Mapped from Ontologies to Dimension Hierarchies. In *Proc. ACM DOLAP Workshop*, pages 57–64, 2012.

[5] A.Y. Halevy and J. Madhavan. Corpus-based knowledge representation. In *Proc. of Joint Conference on Artificial Intelligence*, volume 18, pages 1567–1572, 2003.

[6] A. Maedche and R. Volz. The ontology extraction and maintenance framework text-to-onto. In *Proc. of ICDM Workshop on Integrating Data Mining and Knowledge Management*, 2001.

[7] G.A. Miller. Wordnet: A lexical database for english. *Communications of the ACM*, 13(11):39–41, 1995.

[8] S. Scott and S. Matwin. Text classification using WordNet hypernyms. In *Proc. of the Conference Use of WordNet in Natural Language Processing Systems*, pages 38–44, 1998.