Abstract- The current schema Matching techniques do not stand up to data and schema scaling up. We present existing matching approaches at large scale called holistic and Pair-wise. On the basis of the state of the art techniques we suggest the elaboration of a hybrid approach that combines these known techniques to deal with a large scale Matching.

I. INTRODUCTION

The last decade has brought a large amount of data collected in business and scientific area. The related databases and information sources are available through the web covering different dynamic domains: Web, Deep Web\(^1\), biology, digital libraries, etc. These data filling up and these sources are heterogeneous, frequently changing, distributed, and their number increases rapidly. These facts arise one of the greatest challenges in the data integration field.

Hence, Matching techniques attempt to develop automatic procedures, which search the correspondences between these data. This operation takes data as input and returns the semantic similarity values of their elements/attributes. In large-scale data integration scenarios [13], matching needs to be as automatic as possible and scalable to a large quantity of data. However, current matching algorithms have been performed with simple data holding a small number of components, whereas in practice, real world data are voluminous. The size of data can impact match accuracy because it determines the search space for match candidates. In consequence, the quality of matching (execution time, relevance, etc) will be decreased. Then, it is necessary to conceive a matching system with efficient matching algorithms and optimized techniques adapted to large scale in order to solve these inconveniences. Therefore, our main motivation is to optimize and improve matching large scale algorithms in terms of efficiency and the quality of matching solutions produce.

This paper is organized as follows. In section 2, we describe a review of state of the art in matching at large scale. In section 3, we describe our vision for large scale matching. We present the architecture of our system focusing on the pre-matching step using a data mining algorithm for frequent trees detection. Finally, we conclude and discuss future work.

\(^1\) The deep Web is qualitatively different from the surface Web. Deep Web sources store their content in searchable databases that only produce results dynamically in response to a direct request [9].

II. REVIEW OF EXISTING MATCHING APPROACHES

In the literature, we can distinguish between two matching approaches: Pair-Wise matching and Holistic matching. We discuss in this section the research works related to these approaches, and their different optimization techniques and underline the importance and the usefulness of these techniques in improving the quality of the matching at large scale.

A. Pair-wise Matching

Matching has been approached mainly by finding pair-wise attribute correspondences, to construct an integrated schema for two sources. Several pair-wise matching approaches over schemas have been developed [5, 3, 11, 5, 17] by research community. We are more interested in the approaches that integrate the clustering and fragmentation techniques [5, and 17]. In fact, these techniques aim at reducing the dimension of the matching problem. The main purpose is to optimize and improve the quality of the matching process. In [5], the authors have developed a fragment-based match approach, i.e., a divide and conquer strategy which decomposes a large matching problem into smaller sub-problems by matching at the level of schema fragments. This approach is done a priori before the matcher’s execution. For instance, COMA++ [5] implements this approach. The fragment-based approach represents an effective solution to treat large schemas. However, only a few static fragment types are supported and matching large fragments lead to a long execution time.

The authors in [17] propose a clustered schema matching technique, which is a technique for improving the efficiency of schema matching by means of clustering. The clustering is introduced a posteriori after the generation of matching elements. Clustering is then used to quickly identify regions in the schema repository which are likely to include good matchings for the smaller schema. The clustered schema matching is achieved by an adaptation of the clustering algorithm K-means [20]. However, there is no measure of a cluster’s quality that can be used to decide which clusters have better chances to produce good matchings.

B. Holistic Matching

Recently, holistic schema matching has received much attention due to its efficiency in exploring the contextual information and scalability. Holistic matching matches multiple schemas at the same time to find attribute correspondences among all the schemas at once. Several current approaches to holistic schema matching [4, 6, 7, 8, 12,15 , and 18] rely on a large amount of data to discover
semantic correspondences between attributes. Holistic approach has been introduced in [7, 8]. The authors in [8] propose statistical approaches, MGS (for hypothesis modeling, generation, and selection) and a DCM [7] (Dual Correlation Mining) framework. The MGS framework is an approach for global evaluation, building upon the hypothesis of the existence of a hidden schema model that probabilistically generates the schemas we observed. This evaluation estimates all possible “models,” where a model expresses all attributes matchings. Nevertheless, this approach does not take into consideration complex relationships of attributes. However, these approaches suffer from noisy data. The works suggested in [4, 6] outperform [7, 8] by adding sampling and voting techniques, which are inspired by bagging predictors. The approaches presented in [15] propose a novel clustering-based approach to schema matching. Schemas are clustered based on their contextual similarity. The K-means algorithm has been used in clustering task and a resampling method [14] has been proposed to extract stable attributes from a collection of data. These approaches focused only on 1:1 matchings.

III. OUR VISION FOR LARGE SCALE MATCHING

We illustrate in this section our vision for a large scale matching system. First, we argue for the combination of the holistic and pair-wise approaches. Actually, pair-wise matching is usually achieved between only two voluminous data sources. In contrast to this approach, holistic matching is performed between a set of query interfaces (few components) from the deep web. The combination of holistic and pair-wise matchers analyzes schemas/elements under different aspects, resulting in a more stable and accurate similarity for heterogeneous schemas. Therefore, their combination can effectively improve the quality of matching. Second, we note the importance of optimization techniques, especially clustering and fragmentation methods. The main purpose is to reduce the size of a large scale matching and improve the Quality of the Matching. Moreover, we notice that these techniques have been integrated exclusively either before matching operation (e.g splitting a priori) or after matching operation (e.g grouping a posteriori). Our approach includes these techniques in a priori and posteriori steps. In fact, splitting a priori represents an efficient alternative to deal with very large data and to reduce the size of large matching problems into small sub-problems. Moreover, grouping a posteriori allows us to select and preserve the highly ranked correspondences result. This step improves the efficiency of schema matching. Third, we consider that it is important to integrate a quality evaluation in every step of the matching process. Quality evaluation is essential to guarantee the reliability of data in order to avoid noisy data. It ensures the consistency of using algorithms and techniques. Moreover, it is necessary to evaluate the matching results and to estimate if the matching system satisfies the quality criteria. Precisely, this quality evaluation allows us to test the performance, accuracy, scalability, adaptability and extensibility of the matching system at a large scale. Quality metrics relating to large scale are discussed in [10]. Finally, we assess that it is essential to employ some auxiliary semantic information to identify finer matching and to deal with the lack of background knowledge in matching tasks. It is also the way to obtain semantic mappings between different input data.

Following these ideas, we describe here our proposed architecture for a large scale matching system. (Figure.1) outlines a general procedure for matching at large scale. The proposed architecture is deployed in three phases: Pre-Matching, Matching and Post-Matching.

**Pre-Matching**: This phase represents a pretreatment of voluminous schemas. Firstly, the holistic approach module is at the heart of this phase. It addresses the issue that the schemas from the same domain may share similar and common sub schemas. The assumption is that similar schemas have more common sub schemas. We adapt data mining techniques like a tree mining algorithm to find frequent sub schemas in a set of schemas. In fact, Data mining is the principle of sorting through large amounts of data and picking out relevant information. Next, the resulted sub schemas are pruned from the original schemas and grouped together in a separate partition. We assign a label to each sub schema. The same sub schemas share the same labels. Hence, this operation considerably reduces the sizes of schemas. Subsequently, we check the resulted schemas. If the size of schemas remains voluminous, we apply splitting schemas module (e.g fragmentation). This splitting step (a priori) includes several quality constraints: splitting criteria, reliability of the fragments obtained characteristics of schemas (structure, format), etc.... This operation can be either automatic or manual.

The data mining approach used for pre-matching is summarized in three phases. We will first formally define what a tree is. Then, we will show how to find and extract frequent sub-trees. Finally, we will describe the schema reduction process.
A. Modeling XML Schemas as Trees
An XML schema Tree \( A = (r, N, E, \varphi) \) is an acyclic connected graph, where \( r \) is the root, \( N \) is a set of nodes, \( E \) is the set of edges, and \( \varphi \) is a labeling application \( \varphi : N \rightarrow L \) assigning a label to each node of the tree, where \( L \) is the set of labels of nodes.

B. Identifying and mining frequent sub-trees
This stage identifies inter and intra schemas structures. These XML sub-trees appear frequently in a set of trees. This means determining, given a set of trees (\( F \)) and an arbitrary threshold \( \sigma \), if the sub-trees is included at least \( \sigma \) trees in the set of trees (\( F \)).

We define a tree inclusion as identical inclusion which must preserve the nodes label and parent relationship. Identical inclusion has been defined in [19] as induced inclusion or strict, exact, unordered and strong in [22].

1) Identical Tree Inclusion
Let \( A_1 = (r_1, N_1, E_1, \varphi_1) \) be a labelled unordered sub-tree and \( A_2 = (r_2, N_2, E_2, \varphi_2) \) be a labelled unordered tree. \( A_1 \) is included into \( A_2 \) (noted \( A_1 \subseteq A_2 \)) if there exists an injective mapping \( \mathcal{M} : N_1 \rightarrow N_2 \) verifying the following rules:

\[ R_1 : \mathcal{M} \text{ preserves the labels} : \forall u \in N_1, (u) = \varphi_2(\mathcal{M}(u)) \ (\varphi : N \rightarrow L \text{ is an application that assigning a label to each node}). \]

\[ R_2 : \mathcal{M} \text{ preserves the parent relationship} : \forall u, v \in N_1, (u, v) \in E_1 \Leftrightarrow (\mathcal{M}(u), \mathcal{M}(v)) \in E_2 \]

Inclusion is said to be identical if \( R_1 \) and \( R_2 \) rules.

\[ R_3 : \text{Child nodes are not ordered, Inclusion is said to be relaxed} : \exists u \in N_1, \text{Child}(u) = \{uc / (u, uc) \in E_1\} \]

\[ \exists u' \in N_2, \text{Child}(u') = \{u'c / (u', u'c) \in E_2\} \]

where \( \text{Child}(u) \) is a function returning the child of a node \( u \).

\( R_1 \) implies that \( \text{Child}(u) = \text{Child}(u') \) without conserving child ordering

\( A_1 \subseteq A_2 \) involves that inclusion must verify: \( R_1 \cdot R_2 \cdot R_3 \)

We consider the identical inclusion as shown in figure 2. \( A_1 = (r_1, N_1, E_1, \varphi_1) \) and \( A_2 = (r_2, N_2, E_2, \varphi_2) \) are two trees.

\( A_1 \subseteq A_2 \) means that:

Following \( R_1 : \exists u \in N_1, \varphi_1(u) = \text{Contact} \Rightarrow \varphi_2(u) = \text{Contact} \)

\( (M(u)) = \text{Contact} \)

Following \( R_2 : \exists u, v \in N_1, (u, v) \in E_1 \Leftrightarrow (M(u), M(v)) \in E_2 \)

\( R_1 \) and \( R_2 \) are verified.

Child nodes are ordered \( \Rightarrow \) the inclusion is identical, no relaxed.

2) Frequent sub-trees
Given a set of trees (\( F \)) and an arbitrary threshold \( \sigma \), the problem is to find frequent sub-trees (\( AF \)). We identify two kinds of frequent sub-trees: intra-frequent and inter-frequent (corresponding respectively to intra and inter-schemas). The frequency is computed using the notion of frequency support. The support of a frequent sub-tree is noted Support (\( \sigma, F \)).

Frequency Support
Let \( F = \{A_1, A_2, \ldots, A_n\} \) the set of trees and \( A' \subseteq F \) a frequent sub-tree.

The frequency Support of a sub-tree \( A' \) noted \( \text{Support}(A',F) \) is defined as the set of occurrences of \( A' \) in \( F \).

Intra and inter-frequent sub-trees share the same definition of frequency support.

Intra-frequent Sub-tree
Let \( A \) a tree and \( A' \subseteq A \)

\( \sigma \) is a threshold corresponding to the minimum occurrences of \( A' \) within \( A \).

\( A' \) is an intra-frequent sub-tree if \( \text{Support}(A',A) \geq \sigma \).

Inter-frequent Sub-tree
Let \( F = \{A_1, A_2, \ldots, A_n\} \) the set of trees, \( A' \subseteq F \) a sub-tree.

\( \sigma \) is a minimum threshold corresponding to the minimum trees including the sub-tree \( A' \).

\( A' \) is an inter-frequent sub-tree if \( A' \) is included at least \( \sigma \) trees in \( F \): \( \text{Support}(A',F) \geq \sigma \).

Mining Algorithm
The first objective of our work is to identify and extract common shared structures. We adapt to this aim DryadeParent algorithm proposed in [16] that finds identical included sub-trees.

C. Processing of XML schemas for Reduction
By reducing the XML schema, the task of finding matched schemas is simplified. The main goal of this approach is 1) Reducing the size of large matching problems into small sub-problems. By reducing the size of the match problem we not only aim at better performance but also at improved match quality. 2) Improving the visualization of schemas and match results.

![Figure 2: Example of identical inclusion trees](image)

![Figure 3: Schemas before reduction processing](image)
As illustrated in Figure 3 and 4, frequent sub-trees have been calculated by the reducing process. Then, the labels (AF1 and AF2) have been assigned to the reduced trees.

Matching: After pretreatment schemas phase, we use a pair-wise matcher to find the relations between elements. Pair-wise Matcher module may include elementary matcher or combinations of matchers [21] depending on the application. We can employ an auxiliary semantic resource to find these correspondences.

Post-Matching: In the final phase, we group (grouping a posteriori), the matching results, to select the highly ranked matchings that represent the most pertinent results. The grouping module includes clustering algorithms to merge these most important results in the same clusters. We test, then, the quality of these results to satisfy the accuracy criterion. These results will be saved for a forthcoming use.

IV. CONCLUSION AND FUTURE WORKS

We have presented in this paper a state of the art study covering existing approaches: Pair-wise and Holistic matching techniques. Our proposition concerns the design of a matching system that provides features described as: formalizing quality metrics, mining, splitting, and grouping (e.g. clustering) techniques (in a priori and posteriori phases). Hence, we presented how to scale down the size of schemas in pre-matching stage by using data mining techniques. A synthetic description of the whole system is given as well.

Finally, this work is still in progress in terms of validation on real world schemas.

ACKNOWLEDGMENT

Special thanks to Alexandre Termier, associate professor at the university Joseph Fourier (Grenoble, France) for providing the sources of DryadeParent mining Algorithm.

REFERENCES