MetaExtractor: A System for Metadata Extraction from Structured Data Sources

Alexandra Pomares-Quimbaya, Miguel Torres-Moreno, Fabián Roldán

MetaExtractor: A System for Metadata Extraction from Structured Data Sources

Alexandra Pomares-Quimbaya, Miguel Eduardo Torres-Moreno, and Fabián Roldán

Pontificia Universidad Javeriana, Bogotá, Colombia

Abstract. The extraction of metadata used during the planning phase in mediation systems assumes the existence of a metadata repository that in most cases must be created with high human involvement. This dependency rises complexity of maintenance of the system and therefore the reliability of the metadata itself. This article presents MetaExtractor, a system which extracts structure, quality, capability and content metadata of structured data sources available on a mediation system. MetaExtractor is designed as a Multi-Agent System(MAS) where each agent specializes in the extraction of a particular type of metadata. The MAS cooperation capability allows the creation and maintenance of the metadata repository. MetaExtractor is useful to reduce the number of data sources selected during query planning in large scale mediation systems due to its ability to prioritize data sources that better contribute to answer a query. The work reported in this paper presents the general architecture of MetaExtractor and emphasizes on the extraction logic of content metadata and the strategy used to prioritize data sources accordingly to a given query.

Key words: Large Scale Data Mediation, Metadata Extraction, Source Selection, Multi-Agent System, Mediation Systems

1 Introduction

The processing of queries in mediation systems involves processes of logic and physical planning taking into account the characteristics of the data sources available in the system. This information is typically stored in metadata repositories that are accessed whenever a query is evaluated. Unfortunately, mediation systems assume the existence of previously constructed metadata repositories [1] or focuses only on the structure of data sources, and basic statistics such as the number of records, the number of null data, etc.[2] Although advanced mediation strategies contemplate the existence of more comprehensive metadata such as the number of objects contained in a source [3], the way in which this metadata is obtained has not been well defined. Parallel to mediation systems of structured sources, several proposals have been made to summarize the contents of available sources in the Web: using trees [4] to describe the dependency relationships between sources [5]. These alternatives are useful for increasing the richness of metadata. However, their generation requires high human interaction or includes only general aspects that do not allow a real differentiation among sources, especially when the sources are dependent and data fragmentation is not disjoint. Faced with these limitations of mediation systems this article proposes MetaExtractor a system designed for the automatic extraction of metadata from structured distributed data sources that are registered in a mediation system. MetaExtractor's goal is to increase the quality of metadata repositories through methods that do not require intensive human involvement. MetaExtractor design is based on multi-agent systems that uses data sources as resources to meet its goal of extracting a particular type of metadata. MetaExtractor considers the extraction of structure, capability, quality and content metadata. The structure of this article presents in Section 2 the analysis of related works on the area of mediation systems that allow to put into context the contribution of MetaExtractor. Subsequently, Section 3 presents an overview of MetaExtractor that is detailed in Section 4. Section 5 reports evaluation results on priorization of data sources using the extracted metadata. Finally, Section 6 presents the conclusions and future work.

2 Related Work

Different approaches to support data queries in the context of heterogeneous and distributed sources have been studied for over 15 years. Most of them are based on mediation systems [6] that aims to provide an integrated view of heterogeneous and distributed data sources respecting their own autonomy [7]. The architecture of mediation systems [7,8,9,10] consists of four levels of abstraction: i) the data sources level, ii) the adaptation (wrappers) level, iii) the mediation level and iv) the application level. Query processing in this type of systems is coordinated at the mediation level and takes place in two stages: planning and execution. In the planning stage, sources that can fully or partially answer the query are selected and then the query is rewritten into subqueries, using the external model of each source. To select appropriate sources, the mediator uses a metadata catalogue that includes the external model and other information sources, whose level of detail and accuracy varies according to the type of mediator. In the execution stage, the mediator sends the subqueries to each source and is responsible for coordinating the execution of the queries, and receiving and processing responses in order to integrate them.

One factor that affects the accuracy and efficiency of a mediation system is its planning strategy, which is directly related to the quantity and quality of available metadata. In the first generation of mediation systems, such as: Information Manifold [11], TSIMMIS [9], DISCO [8] and Minicon [12], the planning process uses as metadata information the processing capabilities of sources (e.g. the number of required conditions, or the attributes that could be defined in the predicate). Another group of strategies such as: iDrips and Streamer [1], navigation paths [3] perform the planning process by means of metadata that describes the content of the sources. Other proposals such as QPIAD [13] and the quality-oriented presented in [14] use detailed statistics on the quality of the data contained in the sources to reduce the number of plans. Meanwhile, proposals for large-scale mediation: PIER [15], PinS [16], Piazza [17], PeerDB [18], and SomeWhere [19] use location of metadata, content summary and description of reputation sources, among others.

The metadata used in these systems are an essential input for the performance of the mediation process. However, obtaining such metadata from structured data sources is limited to the manual (human) description of the structure of the sources and some quality statistics obtained during the operation of the system. Even if the owners or managers of each source of information can define some metadata manually, to ensure maintainability, efficiency, consistency and cost of the mediation system, it is necessary to generate strategies for automatically obtain metadata from the mediation repositories.

The existing methods for generating metadata can be classified into three groups: i) extraction, ii) collection and iii) hybrid generation [20]. In the first case the aim is to obtain the metadata based on the execution of techniques for analyzing, mining, evaluating, etc. In the case of metadata collection the goal is to gather pre-built metadata. Finally, hybrid generation obtains existing metadata, and from its analysis generates new metadata [21]. The aim of the MetaExtractor project is focused on metadata extraction from structured data sources that may contain narrative text in some of their attributes.

Literature reviews show that most research efforts suggests strategies to extract metadata from unstructured information available on the web, the vast majority of available methods use processing algorithms that analyze natural language texts in order to generate structured descriptions [22]. These algorithms are part of strategies for information retrieval that assess text in a semantic and syntactic way to get a list of descriptors for the content of the sources [23]. Although there are methods of metadata generation from structured sources its extraction and / or collection is focused on basic attributes such as size or scheme that cannot act as a true differentiator of data sources.

This article presents MetaExtractor a system that modifies and extends these existing methods and techniques to enhance the extraction of metadata from structured sources. MetaExtractor obtains metadata using different techniques according to the type of metadata that must be extracted. In addition, given a query, MetaExtractor prioritizes data sources that better contribute to answer it. The proposed MetaExtractor architecture includes the use of data mining techniques to identify the type of content from a source. The intention is to explore a sample of each data source to establish whether there is a tendency to have some type of instances of a business object (e.g. client, patient).

3 MetaExtractor Overview

As Section 2 illustrates, current mediation system strategies for source selection assume the existence of the required metadata for query planning, extracts only basic metadata, or require an important manual effort to feed metadata repos-

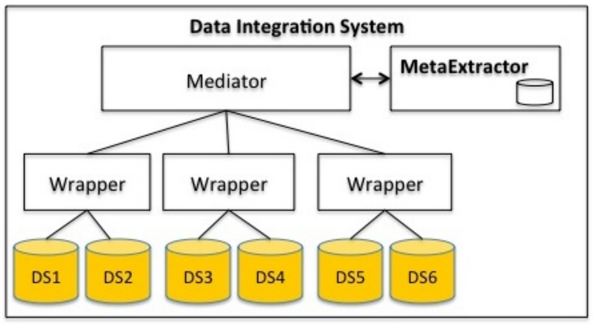

Fig. 1. MetaExtractor in a Mediation System

itories. In order to fulfill this gap this paper proposes MetaExtractor a system that allows to extract structure, quality, capability and content metadata from structured data sources. MetaExtractor architecture is based on the interaction of software agents in charge of extracting different types of metadata from distributed and heterogeneous structured data sources. The main idea behind the agent architecture is for MetaExtractor to have agents located in the same location as the data source and watching over the sources' logs to identify any important or relevant changes on the structure or data contained in each source. This section presents an overview of MetaExtractor functionality and architecture. The following section emphasizes on key aspects of the agents in charge of metadata extraction.

3.1 MetaExtractor Preliminaries

The goal of MetaExtractor is to provide the metadata required for query planning in a mediation system. Figure 1 illustrates the relationship between MetaExtractor and a mediation system, the goal is to act as the knowledge base during the planning phase of a query.

MetaExtractor extracts and stores metadata based on the assumption that data sources contain pieces of information of the relevant objects in a specific domain (e.g. patient, client, campaign) from now on called VDO (see Definition 1). The VDO illustrated in Figure 2 represents a *Patient* composed by four concepts (Person, Medical Act, Medical History, Affiliation). This VDO pertains to the medical domain, which is going to be used from now on to describe MetaExtractor's functionality.

Definition 1 ***Virtual Data Object (VDO)***. *A VDO is a logical set of related concepts relevant to a group of users. Figure 2 illustrates a VDO joining four concepts. Each concept has data type properties whose values are data literals. Concepts are related through object type properties whose values are instances. VDOs are virtual because their instances are partitioned and distributed on several data sources.*

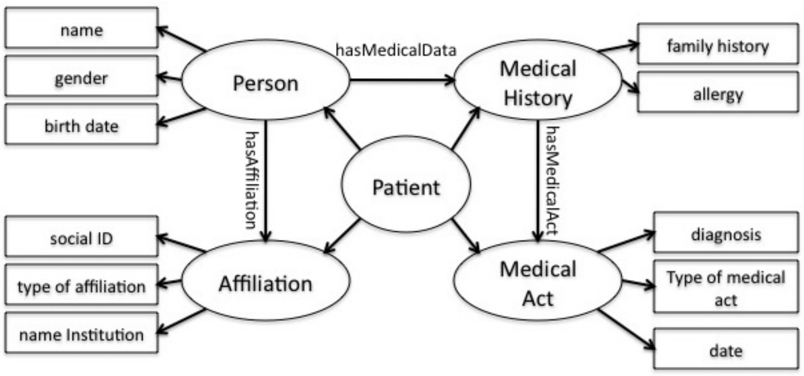

Fig. 2. VDO Patient.

MetaExtractor considers queries with selection conditions on the properties of the concepts composing the VDO. For example, the query hereafter uses the *VDO Patient* and selects the id of patients with a ”Buphtalmos” diagnosis.

Query 1 $Q(VDO_{Patient}, properties(id), conditions(diagnosis = Buphtalmos))$

In the following sections we will work with queries which include several conditions, such as follows:

$$Q(VDO_{name}, properties(id), conditions(condition_1, condition_2, ..., condition_n))$$

When a mediator receives a query it can issue a request to MetaExtractor asking about the facts related to the VDO involved in the query; these facts may be given directly to the mediator or they can be used by MetaExtractor to prioritize the data sources that better contribute to answer the query.

3.2 MetaExtractor Processes

Figure 3 illustrates the agents and processes involved in the execution of MetaExtractor. The GUI process, the Server process and the Source process include a set of sub-processes (agents), which corresponds to activities of an agent with a defined behaviour. The GUI agents represent the interactions of the users, the Server agents manage persistence of metadata and users, and include the logic to prioritize data sources given a particular query. To accomplish their tasks they use the metadata extracted by the Source agents that are in charge of the actual extraction of metadata and are the ones watching over the source itself.

The metadata extraction starts with a communicative act between the Metadata Manager Agent, who proactively and periodically activates the metadata extraction. The activation event triggers the activation of the Source Manager

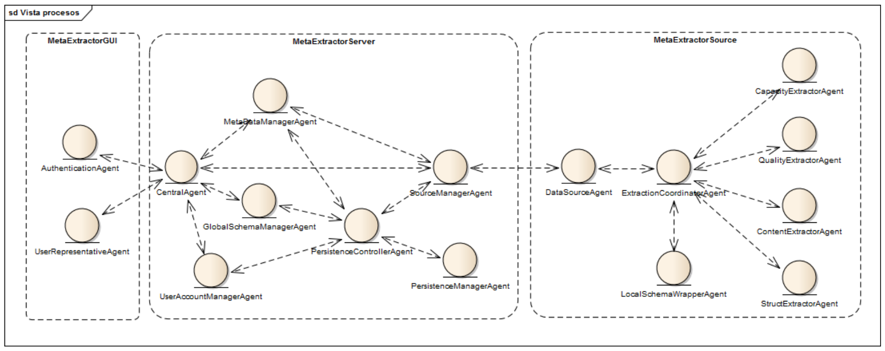

Fig. 3. MetaExtractor Agents Interaction

Agent and this, in turn, sends an event to the related Data Source Agents. Finally, each Data Source Agent communicates with the Extraction Coordination Agent that is in charge of coordinating the extraction of each type of metadata. Currently, MetaExtractor provides four agents in charge of extracting diferent metadata types: structure, quality, content and capability metadata. The following subsection describes each one of metadata types and Section 4 the internal logic of each metadata extractor agent. It is important to notice that according to the requirements of the mediation system, different kinds of metadata extractor agents can be added later on to the system.

Extraction functions Once a data source is registered MetaExtractor is able to extract and/or obtain from them the following metadata (by means of the Source Agent):

1. Structure Metadata: This type of metadata describes the relationship between the objects of a data source with the domain VDOs. For instance, a data source in the medical domain may have a table called *Attention* that contains all the attentions of a patient in a medical institution, this table may be the equivalent of *Medical Act* in the description of the VDO patient. In this case the metadata structure contains the equivalent relationship between the table *Attention* and the concept *Medical Act*. Structure metadata includes the name of the tables, attributes and relationships, and their relationship with the domain VDOs by using the operators Equivalence and Inclusion.
2. Quality Metadata: MetaExtractor provides information about the quality state of a data source. This type of metadata is very important to distinguish which data source is more promising to answer a query when two or more data sources have similar structural and content characteristics. The quality state is described using the following attributes:
   – Null percentage: It serves to identify the level of deterioration of an attribute according to the frequency of nulls related to the size of a

sample. An attribute with a value 65% in Null Percentage can mean that the quality is too low.
   - Null density: It calculates the density of null values in a table, its goal is to differentiate tables that have few attributes with high frequency of null values and tables that have a larger number of attributes with low frequency of null values.
   - Maximum null frequency: Identify the attribute with the highest frequency of nulls.
3. Content Metadata Extraction: Most of the queries over mediation systems involve a condition over a set of attributes of the VDO; for instance in the medical domain, a query over the patient VDO may have a condition over the diagnosis (e.g. diagnosis = cancer). These conditions include a restriction of value over an attribute. The content metadata allows to identify the data sources that better could answer a query according to the role it can play as a contributor of instances of a VDO in the domain where the restriction is true. Roles reflect the ability of data sources to resolve query predicates. Section 4 describes the roles and the strategy used by MetaExtractor to describe the content of a data source.
4. Capability Metadata Extraction: This type of metadata is extracted to identify data sources that have restrictions over the type of queries that they can answer. This metadata is specially important in data sources that can only be accessed through a set of services or an interface that limits the attributes that may be acquired. It is also relevant when data sources demand a specific number of query conditions to answer a query. In the medical domain for instance, a data source may restrict the access of the patient's name or may demand the inclusion of at least one condition in the query.

MetaExtractor provides three functions for querying the metadata required for planning purposes. These functions are:

- Query data source metadata: Allows to query the values of metadata of a specific data source.
- Prioritize data sources: According to the properties and conditions related to a VDO query it delivers the list of prioritized data sources to answer it. In order to prioritize MetaExtractor uses the logic proposed in [24]. This strategy is based on combinatorial optimization techniques.
- Metadata subscription: Allows to obtain changes on the metadata of a data source or a set of data sources automatically.

4 Extractor Agents

This section presents the logic within content, quality and capability metadata extractor agents.

4.1 Content Metadata Extractor Agent

The goal of this agent is to identify the role that a data source may play during the execution of a query taking into account its content. This task is not straightforward in structured data sources, because they may contain different records, each one of them containing different content. As a consequence, the *Content Metadata Extractor Agent* must identify the better way to describe a data source given the content of its records. Lets look an example. Consider three hospital databases DS1, DS2 and DS3. According to the kind of hospital these databases may be specialized on a specific type of patients. For example DS1 is specialized on information about patients with cancer. DS2 is specialized on pediatrics and, as a consequence, it contains only child patients. And DS3 contains both, patients with cancer, child patients, as well as other type of patients because is the database of a general hospital. In this domain the agent must describe the role according to the attribute "diagnosis" and "age" differentiating the specialization of DS1 (i.e. diagnosis = Cancer) and DS2 (age < 18), and indicating that even if DS3 is not a specialist it may contain records with this content.

Roles reflect the ability of sources to solve conditions. Given the analysis of the roles played by a database in a mediation system, we propose the following roles: *specialist* and *container*. The definition of each source role is described in Definition 2 and 3. In these definitions all the instances of *VDOj* stored in data source *DSi* are noted $ext(DSi, VDOj)$ [1]. U designates all the data sources participating in the system. All the instances of *VDOj* available in the system are denoted $ext(U, VDOj)$. The subset of $ext(DSk, VDOj)$ corresponding to the instance that verifies a condition p is denoted $ext(DSk, VDOj)^p$ and $card()$ is the cardinality function.

Definition 2 ***Specialist Role.*** *A data source DSi plays a specialist role w.r.t. a condition p in a query on VDOj iff most instances of VDOj stored in DSi match p.* $IsSpecialist(DSi, VDOj, p) \Longrightarrow card(ext(DSi, VDOj)^p) \geq card(ext(DSi, VDOj)^{\neg p})$

Definition 3 ***Container Role.*** *A data source DSi plays a container role w.r.t. a condition p in a query on VDOj iff DSi contains at least one instance of VDOj that matches p.* $IsContainer(DSi, VDOj, p) \Longrightarrow ext(U, VDOj)^p \cap ext(DSi, VDOj)^p \neq \varnothing$

The process of content metadata extraction is divided into two phases, the Domain Training phase and the Content Description. The following subsections describe both of them.

Domain Training The content of a data source must be described in terms of its domain. For instance, a data source containing electronic health records (EHR) must be described in terms of diagnosis, treatments, medicines, or in terms of the properties of patients like age and gender. Typically, a data source of EHR contains attributes that store all this information; however, some of them, and frequently the most important, are narrative text attributes that contain these terms hidden in lots of text.

[1] This extension contains the VDO identifiers.

The goal of this phase is to record and train the system to recognize what kind of VDOs are contained in a data source. In order to do that MetaExtractor must be trained to identify when the content of a data source is related to a VDO with a specific value in one or more properties. For instance, when a data source contains patients with a specific diagnosis. This identification is straightforward when data sources contain mostly well coded attributes; however, as mentioned, in some domains (e.g. medical) this information may be within texts.

This phase works as follows, for each one of the VDO properties that are usually used as query conditions (e.g. diagnosis) and their possible values according to well known thesaurus or terminology dictionaries (e.g. SNOMED CT [**?**]), MetaExtractor searches on the internet (using an API created for this purpose) in order to identify what are the common words used when talking about the condition with a specific term (e.g. diagnosis and diabetes). The search involves the property and its possible terms (e.g. disease, diabetes). According to the domain of the mediation system, the properties, as well as their possible terms, must be parametrized in the system. At the end of this training phase each one of the terms related to selected properties has associated a dictionary of related words used in conjunction with the term. The outputs are called training dictionaries, and are used later to identify the terms that better describe a data source.

Content Description This phase identifies the role played by a data source within the system. The algorithm proposed to identify the role is illustrated in Algorithm 1. The principle of this algorithm is to use narrative text attributes and a subset of structured attributes from the data source to identify its main content. For example, the narrative text attributes of an EHR that contain the evolution and the treatment of the patient are very important to describe the type of patients the data source has. Additionally, some of the structured attributes (i.e. attribute that contains a limited number of possible values) may provide hints on the content of the data source.

In case the size of a data source exceeds a threshold, the first part of the algorithm reduces its size applying a stratified sample technique [I]. This technique uses a structured attribute to identify the strata.

Then, for each one of the records in the data source sample [II], and using text mining techniques, it annotates the narrative texts taking into account the training dictionaries created in the *Domain Training* phase [III]. As an example we can obtain for a record the annotation (diagnosis = heart failure, sign = pulmonary edema).

Next, analyzing the structured attributes it identifies its value and adds it to the annotations [IV]; After this step the annotation may include (diagnosis = heart failure, sign = pulmonary edema, gender = F).

Once MetaExtractor analyzes all the records it applies a neighborhood based clustering algorithm [25] to identify if there is a specific group of similar records [V] in the data source. The annotations that better describe each cluster are used to specify the role of the data source. In case a cluster contains more than 50%

Algorithm 1 Role Identification

```
Input:  DS //DataSource
Cat(cat1{v1,...,vn},...,catm{v1,...,vn})
sizeThreshold
Output: QRoles(role(catj),...,role(catk))
Begin
    QRoles={}, annotations={}, clusters={}
[I] If (size(DS) > sizeThreshold)
   s = getSample(DS)
 Else
    s = DS
[II] ForAll (reg in s)
   ForAll (att in reg)
          If (narrativeAtt(att))
[III]      annotations[reg] = annotations[reg] + annotate(att)
  Else
[IV]      annotations[reg] = annotations[reg] + idValue(att)
[V] clusters = Cluster(annotations)
    ForAll (c in clusters)
[VI]    If (size(c) > size(s)/2)
QRoles = QRoles(specialist(c.cat))
   Else
    QRoles = QRoles(container(c.cat))
   Return(QRoles)
End
```

of the records of the data source, the data source will be stored as a specialist with respect to the property with the value that better describes the cluster (e.g. diagnosis:heart failure) [VI]; otherwise, it will be stored as a container.

4.2 Quality Metadata Extractor Agent

This agent evaluates the quality state of a data source based on the analysis of nullity of its attributes. To obtain this state MetaExtractor evaluates the frequency of null values in each one of the attributes and the general density of nullity of the data source. The density is very important because some attributes may have high frequency of nullity not because of the quality, but because of the domain. For instance, an attribute middle name may have a lot of nulls because is an optional value in the domain, its null frequency does not reveal a quality problem, but a business rule. In order to assure the performance of MetaExtractor when data sources contain high number of records, it provides the possibility of obtaining quality metadata over a sample from the original data source. This sample follows a systematic approach using as sample size a parameter that can be configured in the system.

4.3 Capability Metadata Extractor Agent

Capability metadata extractor goal is to identify the restrictions of data sources according to the following characteristics:

1. Attributes that can be specified in the query
2. Attributes that can be included in the predicate of the query
3. Attributes that cannot be specified in the query
4. Attributes that cannot be included in the query predicate.
5. Minimum number of attributes that must be in the query predicate
6. Maximum number of attributes that must be in the query predicate

The strategy to obtain this information is based on the execution of prove-queries. These queries are light queries that belong to a defined set of queries each one of them intended to validate one or more of the restrictions. An example of a prove-query ask for a specific attribute without any predicate, this query contributes to the validation of restriction characteristics 1,3 and 5. For evaluating restriction number 2, MetaExtractor tries to execute a query with one condition in the predicate including the attribute compared to a valid value according to the type of the attribute (e.g. numeric, string). If the query does not return an error, the attribute is included in the set of safe attributes in the predicate. The last restriction is evaluated through the incremental of the number of conditions in the predicate. The attributes used in this case are only the ones that passed the restriction 2. Once a data source is registered the defined set of prove-queries is executed over it.

5 Prototype and Functionality Evaluation

In order to evaluate the behavior of MetaExtractor a prototype has been constructed and used to evaluate its extractors and query capabilities. This section presents the main results obtained during this evaluation. Section 5.1 presents the characteristics of the prototype. Section 5.2 details the experimental results.

5.1 Prototype

In order to evaluate the behavior of MetaExtractor we developed the components presented in Figure 4. Components were written in Java. the Global schema and metadata repository are stored in Virtuoso [26]. The communication and interaction of agents uses the multiagent framework BESA[27]. Queries are accepted in SPARQL [28]. The content metadata extractor agent uses the natural language processing API provided by LingPipe [29] and the Weka environment.

Structural Metadata and Global Schema One of the main issues in mediation systems is how to control the heterogeneity of data sources that contribute to the system. Approaches that allow each data source to manage its

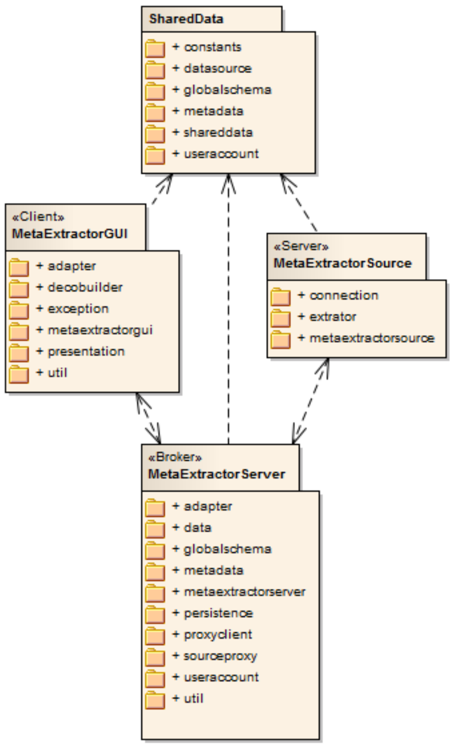

Fig. 4. MetaExtractor Prototype Components.

own schema have demonstrated to scale well, but have important drawbacks in the expressiveness of domain concepts [30]. Because of this, MetaExtractor follows a global schema approach that allows to relate the structure of each one of the data sources to the schema of the domain that allows to create VDOs. This decision was made considering that mediation systems are used to create virtual environments of data sharing around a domain. The global schema defines this domain, the relationship between a data source and the VDOs global schema represents the contribution of this data source to the domain.

The Global Schema in MetaExtractor is created as an ontology in OWL [31]. It can be created by importing it using a *.owl* extension or automatically based on a seminal data source. In the latter case, the tables of the data source are interpreted as *Classes*, its attributes as *Data Type Properties* and the foreign keys as *Object Type Properties*. Although the extraction from a structured data source does not allow to include all the elements of a OWL ontology, they can be added manually after its creation.

In order to create the relationship between data sources and global schema, MetaExtractor provides a method that identifies synonyms between the names of tables and attributes with the names of classes and properties of the global

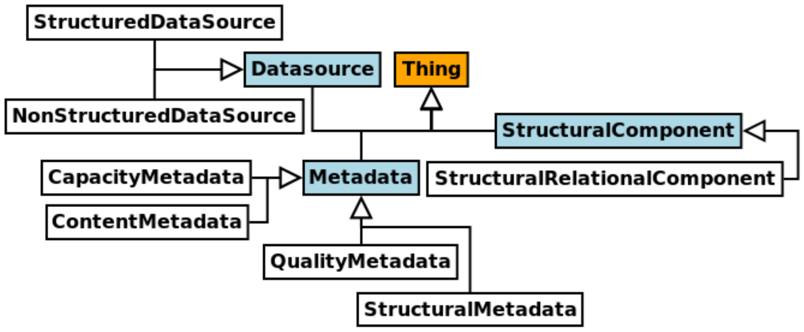

Fig. 5. Metadata Ontology

schema. The execution of this method is not mandatory, but it facilitates the process of data source registration in MetaExtractor.

Metadata Repository Representation Similarly to the Global Schema, Metadata elements in MetaExtractor are described using the syntax and semantics of an ontology. This ontology includes three main classes: *Data Source, Metadata, Structural Component*. The class *Data Source* has two specializations: *Strucured* and *Non-Structured*, the former one includes a restriction that specifies that an individual of this class must have at least one structural component related. *Metadata* class has four specializations, *Structure*, *Quality*, *Capacity* and *Content*. Each one of them has specializations and related properties. Finally, The Structural Component class and its subclasses define the specialization and elements of structured data sources like tables, attributes, etc. Figure 5 illustrates an excerpt of this ontology.

Metadata pieces are stored as triples *(Subject, Predicate, Object)*, respecting the semantics and syntaxis of metadata ontology and global schema ontology. In these triples *Subject* is an individual of the classes defined in the ontology, *Predicate* is a property and *Object* is the value associated to the *Subject-Predicate*.

When a data source is registered in MetaExtractor the first performed action is to create a triple that specifies that the name of the data source is a *Data Source*. Then, MetaExtractor proceeds creating the triple that describes the data source. The following expression is an example on how to create a triple:

```
Triple triple = new Triple(
Node.createURI(systemOntologyBaseName + ‘‘\#’’ + tableName),
Node.createURI(systemOntologyBaseName + ‘‘\#hasAttribute’’),
Node.createLiteral(systemOntologyBaseName + ‘‘\#’’ + attribute))
```

5.2 Experimental Results

Eventhough MetaExtractor was created for being used in any context, the motivational context was the integration of medical data sources containing electronic

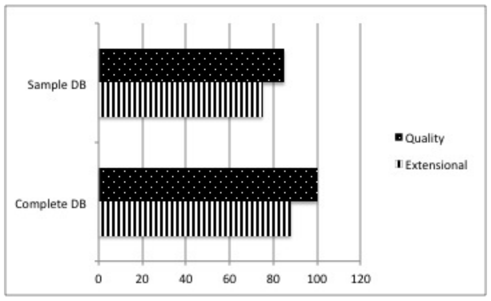

Fig. 6. Precision Quality and Content Metadata with Samples.

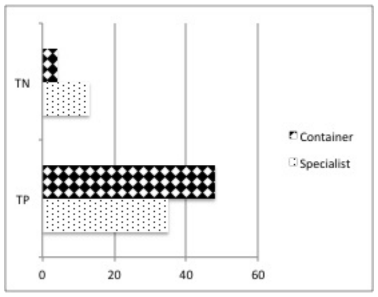

Fig. 7. True Positives and True Negatives Specialists and Containers.

health records in a country. Tests with MetaExtractor were executed in this context. The environment involves 100 simulated data sources containing EHRs. Data sources were created in three different DBMS: PostgreSQL, MySQL and SQL Server 2012. The records contained in each data source were extracted from a database from a general hospital; however, for testing purposes, the database was divided into subsets in order to have different types of specializations (e.g. data sources specialized on obstetrics and gynecology) or general data sources. Categories were trained based on the SNOMED terms.

Tests involve the extraction of structure, content, quality and capability metadata as well as the prioritization using the extracted metadata. Due to space limitations we are going to focus in content metadata extraction tests. Figure 6 shows the results comparing the precision of metadata when the extraction of content and quality metadata used samples instead of the complete databases. Although the extraction of samples reduces the precision of the extracted metadata, the tests allows us to conclude that the precision of metadata is stable and can be used to extrapolate the quality and content of the complete data source.

Figure 7 illustrates the differences on True Positives (TP) and False Positives (FP) taking into account the Container and Specialist roles. The main conclusion of these tests is that the rate of TP and FP confirms the reliability of the system.

6 Conclusions and Future Work

This paper presents MetaExtractor a system that extracts, stores and maintains metadata. MetaExtractor is able to extract content, quality, capability and structure metadata from structured data sources that contain high volume of narrative texts. To the best of our knowledge, this is the first attempt to support formally the extraction of this type of metadata in mediation systems.

MetaExtractor architecture is based on the interaction of software agents in charge of extracting different types of metadata from distributed data sources. It stores metadata in a repository that follows the semantics and syntax of an ontology that allows to obtain precise metadata given a user query. MetaExtractor was designed to be used in architectures where the proportion of distribution with respect to the structure of an object (VDO) is higher than the distribution of its instances, making the metadata useful to reduce the number of sources required for a query. It improves the evaluation of queries that involve predicates with high selectivity and guarantees the dynamic adaptability of the system.

MetaExtractor is being implemented as part of a large scale mediation system. The current prototype is a proof-of-concept. Performance evaluation will be tested in the next stage of development. Short term work involves the analysis of redundancy between data sources to avoid the use of data sources that may provide the same set of instances.

Acknowledgements

This work was supported by the project "Extracción semi-automática de metadatos de fuentes de datos estructuradas: Una aproximación basada en agentes y minería de datos" funded by Banco Santander S.A. and Pontificia Universidad Javeriana.

References

1. Doan, A., Halevy, A.Y.: Efficiently ordering query plans for data integration. In: ICDE '02, Washington, DC, USA, IEEE Computer Society (2002) 393
2. Akbarinia, R., Martins, V.: Data management in the appa system. Journal of Grid Computing (2007)
3. Bleiholder, J., Khuller, S., Naumann, F., Raschid, L., Wu, Y.: Query planning in the presence of overlapping sources. In: EDBT. (2006) 811–828
4. Hayek, R., Raschia, G., Valduriez, P., Mouaddib, N.: Summary management in p2p systems. In: EDBT. (2008) 16–25
5. Sarma, A.D., Dong, X.L., Halevy, A.: Data integration with dependent sources. In: Proceedings of the 14th International Conference on Extending Database Technology. EDBT/ICDT '11, New York, NY, USA, ACM (2011) 401–412
6. Wiederhold, G.: Mediators in the architecture of future information systems. Computer **25** (1992) 38–49
7. Roth, M., Schwarz, P.: A wrapper architecture for legacy data sources. In: VLDB'97, Morgan Kaufman (1997) 266–275
8. Tomasic, A., Raschid, L., Valduriez, P.: Scaling access to heterogeneous data sources with DISCO. Knowledge and Data Engineering **10** (1998) 808–823
9. Garcia-Molina, H., Papakonstantinou, Y., Quass, D., Rajaraman, A., Sagiv, Y., Ullman, J.D., Vassalos, V., Widom, J.: The tsimmis approach to mediation: Data models and languages. Journal of Intelligent Information Systems **8** (1997) 117–132
10. Kossmann, D.: The state of the art in distributed query processing. ACM Comput. Surv. **32** (2000) 422–469
11. Levy, A.Y., Rajaraman, A., Ordille, J.J.: Querying heterogeneous information sources using source descriptions. In: VLDB. (1996) 251–262
12. Pottinger, R., Halevy, A.Y.: Minicon: A scalable algorithm for answering queries using views. VLDB Journal. **10** (2001) 182–198

13. Khatri, H., Fan, J., Chen, Y., Kambhampati, S.: Qpiad: Query processing over incomplete autonomous databases. In: ICDE. (2007) 1430–1432
14. Naumann, F., Freytag, J.C., Leser, U.: Completeness of integrated information sources. Inf. Syst. **29** (2004) 583–615
15. Huebsch, R., Hellerstein, J.M., Lanham, N., Loo, B.T., Shenker, S., Stoica, I.: Querying the internet with pier. In: VLDB. (2003) 321–332
16. Villamil, M.D.P., Roncancio, C., Labbe, C.: Pins: Peer-to-peer interrogation and indexing system. In: IDEAS '04: Proceedings of the International Database Engineering and Applications Symposium, Washington, DC, USA, IEEE Computer Society (2004) 236–245
17. Tatarinov, I., Ives, Z., Madhavan, J., Halevy, A., Suciu, D., Dalvi, N., Dong, X.L., Kadiyska, Y., Miklau, G., Mork, P.: The piazza peer data management project. SIGMOD Rec. **32** (2003) 47–52
18. Ooi, B.C., Tan, K.L., Zhou, A., Goh, C.H., Li, Y., Liau, C.Y., Ling, B., Ng, W.S., Shu, Y., Wang, X., Zhang, M.: Peerdb: peering into personal databases. In: Proceedings of the 2003 ACM SIGMOD international conference on Management of data. SIGMOD '03, New York, NY, USA, ACM (2003) 659–659
19. Rousset, M.C., Adjiman, P., Chatalic, P., Goasdoué, F., Simon, L.: Somewhere: A scalable peer-to-peer infrastructure for querying distributed ontologies. In: OTM Conferences (1). (2006) 698–703
20. Greenberg, J.: Metadata extraction and harvesting: A comparison of two automatic metadata generation applications. Journal of Internet Cataloging **6** (2004) 59–82
21. Margaritopoulos, M., Margaritopoulos, T., Kotini, I., Manitsaris, A.: Automatic metadata generation by utilising pre-existing metadata of related resources. Int. J. Metadata Semant. Ontologies **3** (2008) 292–304
22. Hripcsak, G., Knirsch, C., Zhou, L., Wilcox, A., Melton, G.B.: Using discordance to improve classification in narrative clinical databases: An application to community-acquired pneumonia. Comput. Biol. Med. **37** (2007) 296–304
23. Kowalski, G.: Information Retrieval Systems: Theory and Implementation. 1st edn. Kluwer Academic Publishers, Norwell, MA, USA (1997)
24. Pomares-Quimbaya, A., Roncancio, C., Cung, V.D., Villamil, M.D.P.: Improving source selection in large scale mediation systems through combinatorial optimization techniques. T. Large-Scale Data- and Knowledge-Centered Systems **3** (2011) 138–166
25. Everitt, B.S., Landau, S., Leese, M., Stahl, D.: Cluster Analysis. 5 edn. John Wiley and Sons (2011)
26. Software, O.: Openlink software virtuoso open-source (vos) versión 1.7.2.1 (2011)
27. González, E., Bustacara, C.J., Avila, J.A.: Besa: Arquitectura para construcción de sistemas multiagentes. In: CLEI - Conferencia Latinoamericana de Estudios en Informática Ponencia. (2003)
28. Eric Prud, A.S.: Sparql query language for rdf, http://www.w3.org/tr/rdf-sparql-query/ (2007)
29. Carpenter, B., Baldwin, B.: Text Analysis with Ling Pipe 4. Ling Pipe Publishing (2011)
30. Ooi, B.C., Tan, K.L., Zhou, A., Goh, C.H., Li, Y., Liau, C.Y., Ling, B., Ng, W.S., Shu, Y., Wang, X., Zhang, M.: Peerdb: Peering into personal databases. In: SIGMOD Conference. (2003) 659
31. OWL Working Group, W.: OWL 2 Web Ontology Language: Document Overview. W3C Recommendation (27 October 2009) Available at http://www.w3.org/TR/owl2-overview/.