A Service Oriented Architecture for Linked Data Integration

Pierre De Vettor, Michael Mrissa, Djamal Benslimane
Université de Lyon, CNRS
LIRIS, UMR5205, F-69622, France
Lyon, France
Email: firstname.surname@liris.cnrs.fr

Salim Berbar
Audience Labs
Place Louis Pradel, 69001,
Lyon, France
Email: sb@alabs.io

***Abstract*—Nowadays, the Web offers huge amounts of data sources for the benefit of the community. However, there is a lack of practical approach for converting and linking multi-origin data sources into one coherent smart data set. In this paper, we define a service-oriented architecture to attach explicit semantics to data, to solve heterogeneity issues, and to remove data inconsistencies in order to convert raw documents to quality Linked Data. We motivate the need for a service oriented architecture for smart data with a live scenario based on the Audience Labs company information system. We show how our service-oriented architecture adapts to the company needs and facilitates semantic annotation, data integration and exploitation of the resulting smart data.**

***Keywords*-service oriented architecture, data integration, data semantics, smart data**

I. Introduction

Nowadays, the Web offers advanced interaction perspectives where users are both data providers and consumers. Users comment and review cultural products, they instantly put together their interests and wish lists in profiles, blogs and social networks, producing a huge amount of data across the Web. On top of that, government and big companies open their databases to the world across the Web, especially thanks to initiative such as the open data project[1]. Data sources are typically exposed via Web APIs [2] that can be combined in service mashups [3] to produce highly valuable services. As an example, we could mention the sets of APIs provided by Twitter, Facebook, LinkedIn, Amazon, Youtube, Wordpress, Flickr, Dropbox and many more available, reused in thousands of mashups[1]. These facts allow organizations to open their information systems, enabling for example a better listening to customer needs, or improving their productivity. They have the possibility to integrate available data from Web sources together with their own data, in order to produce smart data[2]. We define smart data as significant, semantically explicit data, ready to be useful to fulfill the stakeholders' objectives.

A. Challenges

Despite the aforementioned advantages, the integration of data from diverse Web sources remains a complex and tedious process, relying mainly on human-based and error-prone tasks. Globally, there is a lack of practical approach for converting and linking multi-origin data pieces into one coherent smart data set. More specifically, the following scientific locks make the transformation of data into smart data a difficult task.

First of all, data usually comes in variable quality, unorganized or not described, and not linked to other sources on the Web. There is a need for providing machine-readable explicit description of data semantics, and linking data sets to each other and to ontologies[3].

Secondly, the combination of heterogeneous data from different sources generates issues. We classify these issues along three levels, which are **syntactic**, i.e. related to data formats and syntax; **structural**, due to differences in data organization; and **semantic** when different knowledge representations are used [6]. These data heterogeneity issues need to be solved to enable data integration.

Thirdly, the integration of multi-origin data sources generates duplicates and noisy, imprecise or inconsistent data, causing loss of meaning or misunderstandings. Specific algorithms are required to clean data from noise and to generate consistent, duplicate-free data.

Therefore, there is a need for a smart data architecture that enables the integration of heterogeneous data sources, by providing the means to annotate data with explicit semantics, merge the annotated data sets, and handle the cleaning and conflict resolution process.

B. Our Approach

In this paper, we define a data source independent and service-oriented architecture that offers the mechanisms

[1]See also http://www.programmableweb.com/

[2]Data integration is the task that consists in combining data from different sources to provide a unified view to the user [4]

[3]Ontologies are shared representations of a domain knowledge, concretely speaking, they are vocabularies to facilitate data interpretation [5]

to detect and resolve data heterogeneity issues, and provides tools to aggregate and process information in order to generate smart data from diverse Web data sources. Each operation in our architecture, as well as smart data access, is available through a linked data service. A linked data service is a Web resource identified via an URI and accessible with HTTP verbs. It exchanges its representations with JSON-LD data. We motivate the need for a service-oriented smart data architecture with a live scenario based on the Audience Labs company information system. We show how our service-oriented architecture adapts to the company needs, facilitates data integration and semantic annotation, and improves the exploitation of resulting data. In addition, we demonstrate how our solution adapts to any scenario and to large scale contexts thanks to its modular architecture and RESTful design.

The challenges mentioned previously are addressed as follows. For the semantic annotation challenge, domain ontologies and linked data tools provide us with the concepts that allow us to extend data with semantic annotations. Our solution relies on a scalable architecture that includes algorithms, available as linked data services, to semantically annotate data. For the data integration challenge, our proposal relies on our Data Mediation as a Service (DMaaS) approach [7], [8], that allows to perform on-the-fly adaptation of semantically annotated data, to resolve data conflicts that appear during the data integration process. For the data cleaning challenge, the data cleaning operations are exposed as linked data services to facilitate data manipulation.

C. Paper Organization

This paper is organized as follows. Section II illustrates our motivation through our company scenario and presents the use cases for the architecture. Section III details our approach and our service-oriented architecture for semantic annotation, filtering and integration of data from diverse sources. Section IV presents related work and highlights the advantages of our approach. Section V discusses our work and provides guidelines for future work.

II. Scaling the Solution to a Scenario

Our scenario comes from the Audience Labs company which has a need for a recommendation system that broadcasts information through communication media, essentially through Web and emails. The main goal of our solution stands in providing tools for studying the impact of a broadcasting campaign upon a customer database in the first place, and then to provide help decision tools as well as a recommendation systems for future campaigns. The objective is to broadcast documents of any kind, such as a newsletter or commercial campaign, to a set of customers. Typically, documents are email-embedded HTML pages containing texts and images and links to points of interest represented as keywords related to points of interest described in ontologies such as DBPedia[4]. The document includes transparent tracking mechanisms that allow us to harvest contextual data about users' interactions with the document. These contextual data about the customer, which are saved in our customer profile database for later use, includes browser information, action date and time, IP addresses, and helps us to build and extend a profile for each customer. Fig. 1 illustrates how our solution operates in the context of the Audiences Labs scenario. It shows the interaction patterns between the system user, customers and the architecture.

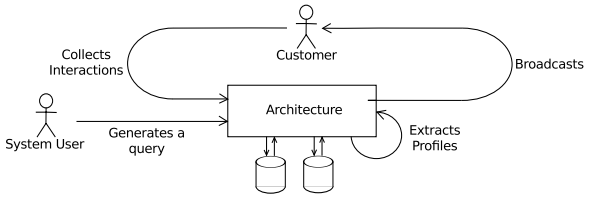

Figure 1. Querying the architecture

A. Querying the Architecture

It is possible to query the architecture through its GUI that offers a set of criteria. The system user creates a query through the GUI and extracts relevant profiles from the diverse data sources that are connected to the platform. The system user has access to several criteria including : age, gender, points of interest and related activities, socio-professional categories, localization (country, region), browser used, and so on. Once the query is formulated, the architecture goes through the semantic annotation, cleaning and integration tasks, as described in the following, and replies to the user query.

B. Annotating Data with Semantics

In order to enrich the raw profiles stored in our database and turn informal data to smart data, there is a need to collect additional information. We need to attach semantic concepts to each piece of data in the scenario to perform data integration. To do so, we link keywords to ontology concepts and reuse external sources such as DBpedia. To attach these descriptive concepts, we use different semantic extraction systems such as named entity extraction [9]. Each piece of data, profile or document is described with a set of criteria, keywords and concepts, also allowing to compute a similarity score between these objects.

[4]http://dbpedia.org/

In addition to the automated mechanisms, we provide the user with tools to manually add complementary information. The manual specifications add declarative information about the foreseen targeted populations, such as gender, age group, socio-professional categories. This declarative task allows the system user to add a *business oriented* input to the data, specifying its own keywords and concepts.

C. Profile Extraction Process

Upon campaign broadcast feedback, we transparently collect customers' interaction traces simultaneously saving all contextual information about these interaction traces such as geolocalization, browser version, device type, and so on. By analyzing the customers' interaction traces, each customer is being attributed a specific profile that describes its points of interests. The interest profile is updated upon each of the customer's interaction. External data sources are also used to enhance the classification and evaluation of the user's points of interest. The semantic concepts that have been attached to data are reused to compute similarity between customers. The harvested data helps to retrieve all users that are the more likely to be interested in a document, when the user performs a new query on the interface, closing the usage loop.

III. Enabling our Service Oriented Architecture

Our approach relies on a service oriented architecture in which we define the different tasks required to prepare, semantically annotate and clean data so that it becomes a consistent "smart data" set.

A. Global Overview : A service oriented architecture

Erl et Al. [10] defined eight service design principles that form a methodology for engineering software components that are decoupled, cohesive and reusable, regardless of their location. These principles are standardized service contract, loose coupling, abstraction from implementation, autonomy, statelessness, discoverability and composability. Based on these principles, we design and implement a generic, scalable and modular architecture, divided into different layers, represented in figure 2. Each architecture layer handles a specific task of the data integration process.

We defined a set of layers according to the different tasks to be performed on data: the *data source management* layer, the *semantic annotation* layer, the *data integration* layer, the *data filtering* layer and the *data consumption* layer. The *data filtering* layer being a cross-cutting concern, between each task execution, we insert a cleaning and filtering task to filter data, removing duplicates and malformed pieces of information and correcting data inconsistencies when possible. We deploy each software component as a linked data service.

B. The Data Source Management Layer

The **data source management layer**, handles the data access tasks, and communicates with the different data sources available to the architecture. Data sources are available in various forms, such as databases, tabular files or web data services. Each sources is described in a data dource configuration resources which are used for data source administration and access. Each file contains the necessary informations for consuming the data source : a location (URI notation), a data type (CSV file, SQL database, etc), and the method for accessing data. Listing 1 illustrates data source configuration file structure.

```
"source":{
    "id" : "U1",
    "type":"database",
    "format":"mongodb",
    "uri":"mongodb://153.75.28.26:8080/myDBendpoint",
    "username":"user1",
    "password":"76ls6h",
    "databasename":"maindb"
}
```

Listing 1. A datasource configuration file examples

This layer benefits from "data source configuration" resources, allowing to perform the four CRUD (Create, Read, Update and Delete) operations to manage data sources. The CRUD operations allow to upload (PUT), modify (POST), retrieve (GET) and remove (DELETE) data source configuration resources. As an example, a GET request over the data source configuration resource with the id of a data source

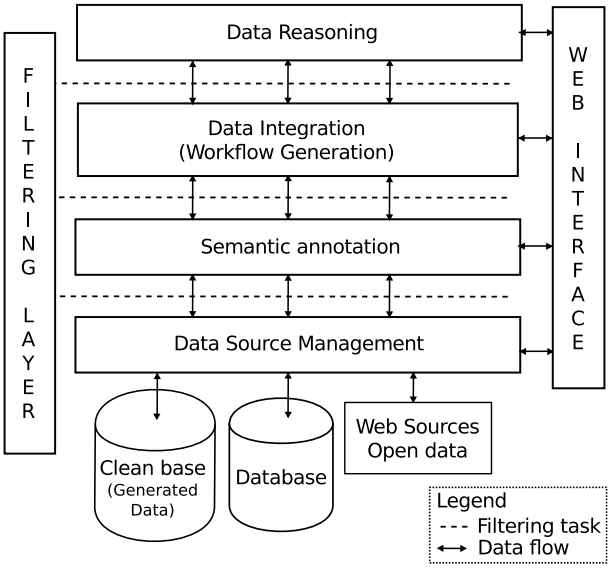

Figure 2. Architecture Layers

: `GET /datasource/get?id=U1` will return a JSON object representing the database source with the id $U1$ described in listing 1. This object contains the properties (`url`, `username`, `password`, `databasename`, etc...) which allows the architecture to connect and retrieve data from the database.

C. The Semantic Annotation Layer

The **semantic annotation layer** handles the second task of the data integration process. It adds explicit semantics to data. This layer helps to extract semantics from data, analyzing data without pre-existing semantic annotation. The system relies on different contextual elements, such as data source, field name, plain text data analysis, comparison and matching with existent annotated data. If the data is provided by a service, the algorithm analyzes its description, and detect whether or not the source type is specified. In case of databases, contextual information coupled with tuple analysis, provides us with the necessary information to describe it as linked data. For all other use cases, such as data files, our mechanisms rely on file structure to detect and identify concepts. These tools identify raw data concepts and link fields to concepts from ontologies, when available, or from DBpedia. Each piece of data is then annotated with the help of these concepts. This layer is one of the most important component of the system, since it provides a machine-readable description of data semantics to be manipulated.

Here we defined different linked data services to semantically annotate flowing data. The first semantic annotation tools generates an annotation process from a data source configuration resource. We defined another linked data service which allows to add (PUT), update (POST), retrieve (GET) and delete (DELETE) a semantic annotation process for each data source. As an example, the request `GET /annotation?id=U1` provides the process (mapping) for the datasource with the id $U1$.

D. The Data Integration Layer

The **data integration layer** enhances the interconnection of the data piece, relying on query that has been created, to automatically connect concepts. On complex query execution, data which has been prepared and semantically annotated through the previously described layer, could be put in correlation to construct the request response. The architecture analyses the different concepts in the query and prepares data for the merging process, relying on metadata to connect data from the different sources. It generates a composition that provides the requested pieces of information.

To detect possible heterogeneity problems, could they be syntactic or structural, and to reconcile them, we rely on our previous DMaaS approach [8], which is an automated solution for resolving data heterogeneity problems between semantically described data, using a decentralized (peer to peer) repository of mediation services[5]. The DMaaS approach classifies data heterogeneity issues according to the syntactic, structural and semantic levels, and provides adapted mediation along these levels. We set up the automatic conflict detection mecanism, which analyses input metadata, and intercepts data responses to performs reconciliation. These data mediation services are published into a peer to peer repository, distributed around a chord ring. Discovery task is made possible thanks to the publication method, each data mediation service is published by its input and output concepts. The architecture submit a composition, defined by a list of resource URIs, to DMaaS which analyses data homogeneity. DMaaS core analyzes input and output metadata of the composition and veriy the semantic matching of each data field. On conflicts detection, the system performs discovery request to find the data mediation service that will handle the transformation of the heterogeneous data. DMaaS returns the new generated composition.

E. The Data Filtering Layer

In our architecture, each time data flows from one service to another, a *cross-cutting* layer called the **data filtering layer** analyzes data. The layer provides cleaning and filtering linked data services that cleanup and format data samples, remove duplicate tuples and reformat malformed data, including eventual encoding issues. In addition, it is possible to specify filtering rules, allowing to ignore specific data values, or to limit a request to a range domain. We defined a linked data service which performs the management of filtering rules, allowing to create (POST), retrieve (GET), modify (UPDATE) and remove (DELETE) a filtering rule.

F. The Data Reasoning Layer

Our architecture also integrates an inference engine which we call **statistics engine**, which runs in background, regularly analyzing data, generating rules and producing statistical knowledge. This statistical machine learning engine constantly updates a specific database with additional knowledge. We integrate such an engine to deal on-the-fly with complex user queries and get an answer in a reasonable time. We also add the possibility, for an agent, to manually add business rules. In this layer, we use data and customer traces, to infer these rules and create data. We use classical IA mechanisms and algorithms such as collaborative filtering (Apache

[5]Mediation services are Web services dedicated to data conversion.

Mahout) and semantic Web inference engine (Jena, Pellet, Euler EYE, HermiT)[6]. We defined a linked data services allowing the agent to add (POST), modify (UPDATE) or delete (DELETE) business rules. We also provide the possibility to search for a business rule according to different criteria, these criteria passed as JSONLD objects. As an example, the following request will list all the business rules of type *SameAs*.

```
GET /reasoner/search?rule=type&query=sameas
```

G. Architecture Orchestration

Each component of our architecture is available through a Web resource, the architecture handles resource orchestration. The architecture disposes from a user interface, which generates data requests (i.e. queries) from customer activities.

We identify two global cases in our architecture: data queries and data import. Our architecture adapts to our scenario as follows. Data queries come from system users, when they interact with the user interface, for example, when users want to generate a list of profiles corresponding to a set of criterias. Data import comes from customers when they interact with broadcasted documents. These two tasks represent read and write actions, not completely independent in our architecture. The write task will be computed by descending in the layers, ending up the actual write operation in the data source, whereas the read task begins by reading from data sources, and then ascending through the layers.

When the architecture receives a query, the architecture selects the appropriate data sources to generate query results. The architecture creates a composition of these data sources, adds semantic annotation and filters data. Then the architecture executes the composition, invokes resources, analyzes the data produced and checks whether or not data mediation is required. The generated smart data is then returned to the system user.

IV. Related Work

Smart data architecture is currently a hot research topic explored by the community. **Dustdar et al.** present a *peer data network* architecture in [11], where data sources are independent databases and where tasks are separated into levels, isolating data management and service integration. Their solution provides service-based optimization, such as peer replication, to resolve data issues. However, the paper does not address data heterogeneity problems, assuming that schema mapping is sufficient. **QuerioCity** [12] presents a smart platform to index and query heterogeneous information from complex systems such as city data portals[7]. The approach clearly distinguishes between the data integration and data consumption tasks. In order to harmonize data usage, data fields are converted to a standard format, annotated with metadata and aligned with public ontologies (Dublin Core[13] or FOAF[14]). We rely on these approaches to build our proposal, improving the reusability and loose coupling through usage of linked data services, automating the linked data efforts and restructuring the different layers of the platform according to our needs that require reasoning about data and proposing a loosely-coupled approach for the different tasks to perform on data.

In another context, some approaches propose techniques for semantically annotating raw data from heterogeneous sources. **Furth et Al.** [15] propose a technique based on natural language processing applied on technical documents for extracting a set of ranked concepts that represent the main subject of the document. **Venetis et Al.** [16] describe a system that recovers semantics from tables on the Web. Their solution relies on similarity computation with help from a set of millions of other tables. Han et Al. present in **RDF123**[17] an open-source tool for translating spreadsheet data to RDF, relying on a column-based mapping, where a set of expressions represents the structure and orchestration of cells in a tabular row. We rely on this latter approach to perform our semantic annotation task, improving it by automating the creation of the mapping expression with help from external and third party services for semantics extraction and concept recognition.

V. Conclusion

In this paper, we present a service-oriented architecture that provides a layered approach for combining, converting and linking multi-origin data sources into one coherent smart data set. We structured our architecture to be as generic as possible, independent from data sources, and adaptable to any use case. We demonstrate the applicability of our architecture in the context of a scenario that answers the needs of our partner company. Future work includes performing additional evaluation over large data sets and exploring issues related to data management such as data quality and freshness issues, and reasoning about inconsistent or imprecise data.

Acknowledgment

This work was supported by the Région Rhône Alpes (ARC6), France.

References

[1] T. Heath and C. Bizer, *Linked Data: Evolving the Web into a Global Data Space*, ser. Synthesis Lectures on the Semantic Web. Morgan & Claypool Publishers, 2011.

[6]See http://www.semantic-web-journal.net/sites/default/files/swj120_2.pdf for a comparison of reasoners.

[7]Such as Dublinked http://www.dublinked.ie/

[2] M. Maleshkova, C. Pedrinaci, and J. Domingue, "Investigating web apis on the world wide web," in *ECOWS*, A. Brogi, C. Pautasso, and G. A. Papadopoulos, Eds. IEEE Computer Society, 2010, pp. 107–114.

[3] D. Benslimane, S. Dustdar, and A. P. Sheth, "Services mashups: The new generation of web applications," *IEEE Internet Computing*, vol. 12, no. 5, pp. 13–15, 2008.

[4] M. Lenzerini, "Data integration: A theoretical perspective," in *PODS*, L. Popa, S. Abiteboul, and P. G. Kolaitis, Eds. ACM, 2002, pp. 233–246.

[5] T. Gruber, "Ontology," in *Encyclopedia of Database Systems*, L. Liu and M. T. Özsu, Eds. Springer US, 2009, pp. 1963–1965.

[6] M. Nagarajan, K. Verma, A. Sheth, J. Miller, and J. Lathem, "Semantic interoperability of web services - challenges and experiences," in *Web Services, 2006. ICWS '06. International Conference on*, Sept 2006, pp. 373–382.

[7] P. D. Vettor, M. Mrissa, and C. Pedrinaci, "Context mediation as a linked service," in *ESOCC*, ser. Lecture Notes in Computer Science, F. D. Paoli, E. Pimentel, and G. Zavattaro, Eds., vol. 7592. Springer, 2012, pp. 210–211.

[8] M. Mrissa, M. Sellami, P. D. Vettor, D. Benslimane, and B. Defude, "A decentralized mediation-as-a-service architecture for service composition," in *WETICE*, S. Reddy and M. Jmaiel, Eds. IEEE, 2013, pp. 80–85.

[9] E. F. Tjong Kim Sang and F. De Meulder, "Introduction to the conll-2003 shared task: Language-independent named entity recognition," in *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, ser. CONLL '03. Stroudsburg, PA, USA: Association for Computational Linguistics, 2003, pp. 142–147. [Online]. Available: http://dx.doi.org/10.3115/1119176.1119195

[10] T. Erl, *Service-Oriented Architecture (SOA): Concepts, Technology, and Design*. Prentice Hall PTRs, Aug. 2005.

[11] S. Dustdar, R. Pichler, V. Savenkov, and H. L. Truong, "Quality-aware service-oriented data integration: requirements, state of the art and open challenges." *SIGMOD Record*, vol. 41, no. 1, pp. 11–19, 2012.

[12] V. Lopez and S. Kotoulas, "Queriocity: A linked data platform for urban information management." in *International Semantic Web Conference (2)*, ser. Lecture Notes in Computer Science, vol. 7650. Springer, 2012, pp. 148–163.

[13] S. Weibel and T. Koch, "The dublin core metadata initiative : Mission, current activities, and future directions," *D-Lib Magazine*, vol. 6, no. 12, 2000.

[14] D. Brickley and L. Miller, "The Friend Of A Friend (FOAF) vocabulary specification," November 2007, http://xmlns.com/foaf/spec/. [Online]. Available: http://xmlns.com/foaf/spec/

[15] S. Furth and J. Baumeister, "Towards the semantification of technical documents," in *FGIR'13: Proceedings of German Workshop of Information Retrieval (at LWA'2013)*, 2013.

[16] P. Venetis, A. Y. Halevy, J. Madhavan, M. Pasca, W. Shen, F. W. 0003, G. Miao, and C. Wu, "Recovering semantics of tables on the web." *PVLDB*, vol. 4, no. 9, pp. 528–538, 2011.

[17] L. Han, T. Finin, C. Parr, J. Sachs, and A. Joshi, "Rdf123: From spreadsheets to rdf," in *The Semantic Web - ISWC 2008*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2008, vol. 5318, pp. 451–466.