Journal of Computer and Communications, 2016, 4, 1-9
Published Online May 2016 in SciRes. http://www.scirp.org/journal/jcc
http://dx.doi.org/10.4236/jcc.2016.46001

Scientific Research Publishing

Let Some Unforeseen Knowledge Emerge from Heterogeneous Documents

Maria Teresa Pazienza, Armando Stellato, Andrea Turbati

DII, University of Roma Tor Vergata, Roma, Italy
Email: PAZIENZA@INFO.UNIROMA2.IT, STELLATO@UNIROMA2.IT, TURBATI@INFO.UNIROMA2.IT

Received 12 March 2016; accepted 9 May 2016; published 12 May 2016

Abstract

Data production and exchange on the Web grows at a frenetic speed. Such uncontrolled and exponential growth pushes for new researches in the area of information extraction as it is of great interest and can be obtained by processing data gathered from several heterogeneous sources. While some extracted facts can be correct at the origin, it is not possible to verify that correlations among the mare always true (e.g., they can relate to different points of time). We need systems smart enough to separate signal from noise and hence extract real value from this abundance of content accessible on the Web. In order to extract information from heterogeneous sources, we are involved into the entire process of identifying specific facts/events of interest. We propose a gluing architecture, driving the whole knowledge acquisition process, from data acquisition from external heterogeneous resources to their exploitation for RDF trip lification to support reasoning tasks. Once the extraction process is completed, a dedicated reasoner can infer new knowledge as a result of the reasoning process defined by the end user by means of specific inference rules over both extracted information and the background knowledge. The end user is supported in this context with an intelligent interface allowing to visualize either specific data/concepts, or all information inferred by applying deductive reasoning over a collection of data.

Keywords

Computing Methodologies, Knowledge Representation and Reasoning, Information Extraction

1. Introduction

While there is a great expectation for new technologies enabling humans to directly access whatever produced information, there is a troubled attitude in respect of both the huge data volume and privacy matters. Especially in the Big Data and Web contexts, either data or specific analyses are becoming increasingly complex. In fact, while technology plays an important role in making information accessible to everyone, everywhere, in every

format, it is generally agreed that several issues still will have to be addressed to capture the full potential of Big Data over the Web. In fact, there are still open questions related to:

1) **size** (volume)
2) **freshness** (velocity)
3) **structured/unstructured** (variability)
4) **uncertainty/quality** (veracity)

that are generally referred as the famous **4 V** features that basically characterize Big Data applications development. Among these features, *veracity* is of great relevance for application systems. In fact, any user is interested in maintaining veracity of information derived through a complex extraction process from several heterogeneous documents. The danger is the inconsistency that can be generated by combining data that are true at different point in time and in different contexts. In fact, sometimes in the extraction process we correlate independent info was not created to be combined.

Indeed, we are convinced that Big Data pertains to how we can derive value from information, by answering ad hoc questions, in a timely fashion, by specializing an information extraction approach.

Independently of the final objective, what we are interested into is to acquire as much as possible relevant information to satisfy a specific question of interest: if the process concludes correctly in a timely fashion, it will produce added value. It is not only the task of extracting (both extensional and intensional) knowledge through massive data analysis and processing: persons are interested in the generation of concise models defined over the analyzed data to transform big data into actionable knowledge that is in applying the extracted knowledge in forthcoming decision processes. This process must be governed to let relevant information emerge. We need semantics to integrate different world models.

Big Data deal with multiple heterogeneous data sources (texts, documents and several kind of media) as well as with different types repositories. This information context is very common on the web and easily accessible: humans are interested in managing personally such information sources for numerous applications.

Governing heterogeneity is not an easy task also for experts; making it feasible for knowledge workers as well as private citizens requires developing algorithms for data transformation/aggregation/integration that must be enriched with further capabilities for managing inherent problems, such as provision of poor quality, inaccurate, irrelevant or fraudulent information, lack of knowledge models that are common in such an open environment [1]. Among others, ontologies assume an important role in supporting knowledge integration: in fact, different and heterogeneous information may relate to different world models that must be made explicit to the integration tools for the alignment of conceptual reference models (ontologies).

Last, but not least, let us consider visualization matters. Humans push for data visualization and analysis tools that simplify information access by enabling to define which aggregated data (returned from search processing tools) should be visualized, how, and whether an analysis of the aggregated data should be done.

To reach such a goal, we do not need only numerical/statistical analysis: we need tools “to understand” extracted data and identified relations: insights and foresights will easily follow!

All these aspects require new approaches to applications development and, as a consequence, new methodologies/platforms for the processing.

2. Pending Matters

As there is a wide interest of the scientific community for developing methodologies and technologies for extracting data from the Web, several progresses have been done in the direction of supporting humans’ involvement in the process of easily extracting relevant information. Nevertheless, several scientific challenges still remain; among others:

- algorithms don’t scale as expected;
- search and retrieval tasks could be improved;
- innovative approaches to integrate heterogeneous unstructured data must be defined;
- relevant visualization aspects are still neglected;
- multi-lingual and multi-cultural data require conceptual reference models for further processing;
- integration of different world models is still a utopia;
- role played by semantics is often neglected;
- new architectures for processing heterogeneous information sources in timely fashion and with limited end-user intervention are not so popular.

While such topics are of great interest, the need for architectures dedicated to the processing of heterogeneous info reveals to be of very high priority being necessary for dealing with the other ones.

Moreover, a not entirely negligible aspect of the search for info in the big mass of data is related to the need of awareness of the conceptual models of reference. Ontological knowledge representation and language processing tools may play an important role in this context. In fact, ontology-based models allow to:

- describe complex data;
- integrate heterogeneous information sources;
- use formal reasoning tools that, by combining reference conceptual knowledge with extracted data, are able to recognize unpredictable contexts;
- explain reasons behind the reasoning;
- integrate into lexicalized ontologies capabilities of both language and knowledge;
- support deep comprehension of extracted data and their interrelationships.

Related to the need of end-users to maintain personal data under control, ontological reasoning may be activated over several levels of abstraction and show details with numerous comments over reasoning process and modalities. It is of great interest for the user to become aware on the (possibly) complex path linking original data whit the inferred ones. This could have a couple of motivations:

- check for soundness of newly produced knowledge;
- be conscious of what kind of information exists and is possible to associate through different relations to produce the inferred result.

For developing this kind of intelligent systems, architectures play a relevant role. UIMA (Unstructured Information Management Architecture) [2] allows for developing applications dealing with heterogeneous complex data accessible on the web. UIMA offers an architecture and an associated framework supporting development and orchestration of analysis components—called Analysis Engines (AE)—for the extraction of information from unstructured content. UIMA is not limited, though naturally bound, to Text Analytics, and can be potentially adopted for processing any kind of media.

Over that, CODA [3] (an architecture and an associated framework) supports RDF trip lification of unstructured and semi-structured content. It extends UIMA with specific capabilities for populating RDF datasets and evolving ontology vocabularies with information mined from unstructured content. Both frameworks are commonly adopted for developing applications in a big data context, thus speeding the production of new inferred knowledge covering the entire spectrum of information content: from fully undetermined structure (the “content”) to fully documented and traditionally accessed structures. Efficient Information Management and, before that, Information Gathering, are thus becoming fundamental aspects of every task/application willing to gear with a considerable flow of (personal) data.

3. The Architecture for Heterogeneous Data Processing

To support easily development of applications for data elicitation over heterogeneous content, we propose an architecture, shown in **Figure 1**, composed of four distinct components; each one has specific abilities in order to solve a problem dealing with heterogeneous information extraction:

- ***File recognition***: identification of the type of file containing the relevant data. Then all data and metadata are more easily extracted and placed in a common structure, which can be further enriched by a user; such as tructure follows the XML specifications (for example the content of a text file, such as a pdf, is placed in the XML, along with all the metadata associated to that file);
- ***Filling the annotation structures***: produced XML is passed to a UIMA pipeline which, taking advantages of several dedicated UIMA Analysis Engines, is able to annotate all the relevant information, producing a JCAS (Java Common Analysis Structure);
- ***Populating the RDF dataset with new RDF triples***: JCAS is the input of CODA, where the rules specified in a PEARL (ProjEction of Annotations Rule Language) [4] document, are used to generate the new RDF triples that will populate the reference dataset;
- ***Knowledge extraction***: when all the files have been processed, the resulting ontology can be shown inside the ontology editor Semantic Turkey [5] and new knowledge is inferred thanks to the reasoner HORUS (Human-readable Ontology Reasoner Unit System) [6];

Hereafter each component of the proposed architecture will be described in details.

3.1. File Recognition

This architecture has been developed to deal also with big data over the web: it is expected input data be composed of files in different format and media types. Recognizing files characteristics are a preliminary problem to be solved (step 1 of **Figure 1**). Each file can contain several metadata. The metadata is associated, often automatically, by user or by the software which is used to generate or manage the file itself. For example, an audio file, containing a song (a CD file or an mp3), can have as metadata the name of the song, the author, the album name, etc. Each type of file has different type of metadata associated to itself (associate a song name to a file containing a picture does not have much sense). In our architecture, we are interested not only in the plain text (for text files), or in the transcript of the audio or the video; we want to both identify and extract all the associated metadata, as we believe that metadata contain relevant information to support the information extraction task. For example, by analyzing the metadata, we can immediately decide whether to invest resources in processing the content of the file or not, or to skip the file processing task. Since different format and standard are associated to different type of metadata, we produce an XML file, to have a standard structure that contains all the possible information found in the different formats. The information, stored in the XML structure, can be expanded using other data not immediately visible in the current file (for example it is possible to associate the same metadata information to a set of files, such as the metadata author, source location, etc.).

3.2. Filling the Annotation Structures

In the context of our framework, each file is associated to an XML structure (this structure can be serialized and stored for different uses). Such XML is then passed to a UIMA pipeline (step 2), in which several UIMA Analysis Engines (AEs) are active. The number and capabilities of these AEs depend on the task we wish to perform, given the input files. The idea behind each AE is that each one should deal with a specific aspect of the input data, being able to deal with a particular information stored in the XML file or using the annotation provided by the previous AE. For example, it is possible to have dedicated AEs for a specific type (to deal with text files, or to audio files, etc.). These AEs, thanks to what is expressed in the XML, do not care about how to extract the metadata from the original file, they already access to information associated to each specific file already extracted and normalized. Thanks to the modularity and scalability provided by UIMA, it is possible to add new AEs anytime, when new capabilities are required by the system. Once all the AEs have completed processing the input XML files, they produce a JCAS (Java Common Analysis Structure). In this particular structure, which is one of the main features of UIMA, all the annotations produced by the AEs are stored. Inside the JCAS, UIMA placed both the original content to be processed (in this case the whole XML, containing the extracted metadata in the previous step and the content of the file) and the annotations.

3.3. Ontology Enriching with RDF Triples

Once the current XML structure (corresponding to a particular input file) has been completely annotated, the resulting annotation structure, called JCAS is passed to CODA (step 3 of the architecture). CODA uses the annotation produced by the UIMA components, and transforms them into RDF triples. This transformation is carried on thanks to an input file written in PEARL (a language developed to specify how each annotation, and all the associated information, should be considered when trying to produce RDF triples). Using PEARL, it is possible to specify not only annotations to be used, but also which specific part of the annotation to extract to assemble the required RDF triples. Consider that a UIMA annotation is not just the identification of specific portion of the input file with a given meaning (according the to the AE which produced the annotation), it is based on feature structures (a list of attribute value couples, in which the value can be a feature structure as well and so on). This is the main advantage of having a dedicated language, as PEARL, that provides the possibility to specify which part of an annotation to use. These RDF triples are then added to an ontology. CODA does not just produce an ontology, but it is able to interact with an existing ontology to use already existing RDF resources, when producing new triples to be added (avoiding the duplication of already existing data).

3.4. Knowledge Extraction

The target ontology, with the new RDF triples (that can even be validated by a user, during a possible interactive process) is loaded and visualized inside Semantic Turkey. Semantic Turkey is able to show all the information

stored in the ontology (step 4 shown in **Figure 1**). For each ontological resource, Semantic Turkey shows all semantic relations, so the user can validate and appreciate the kind of formal data generated inside such an architecture. Then, by using the reasoner HORUS, through a Semantic Turkey extension, it is possible to infer new knowledge, from the explicit one. HORUS uses a set of inference rule for this task. A list of existing rules is provided with the reasoner (the standard inference rule, such as the transitive rule, the symmetric rule, the subclass rule, etc.). Thanks to a dedicated and easy to use GUI, new rules can be added at runtime and those already present can always be managed and made more suitable for specific knowledge extraction tasks. In fact, by adding ad-hoc rules, it is possible to guide knowledge extraction process. It represents the way in which the final user interacts with the system and finalizes the inferential process to his own information needs.

Another interesting feature of HORUS, which distinguishes it from the other existing reasoners, is the possibility to show the motivation behind each new inferred RDF triples. Two distinct and complementary visualizations have been made possible:

- all the triples used in the inference process may be shown as a plain list;
- a more complex graph visualization approach to make visible a larger inference context.

The original aspect of the proposed architecture relies on such an interaction between the ontology editor (Semantic Turkey) and the reasoner (HORUS). It is independent from previously existing general ontology of reference. By allowing users to write their own inference rules, we are providing a simple, yet powerful, approach in defining and enriching a personal ontology, in which usersareable to infer the kind of knowledge that has a meaning for them in a specific context/application. It could be possible to write the features, such as OWL restriction for example, in the ontology itself and then let an off-the-shelf reasoner do its task; it is like embedding in the ontology the users personal view of the world, so we are stating that their view is an ontological view, while, in our case, we are just trying to state that the particular inference rule has a meaning only in a specific context. Then, using the GUI provided by HORUS in the Semantic Turkey framework, the user is also able to see and understand which rule was used to generate the new knowledge and, if the new knowledge is disliked, he could just change the personalized rule, without the need to change the underneath ontology (which is always a delicate task). On the other hand, if a user needs to change a standard OWL inference rule to avoid a particular inference, then this mean that he needs to change the ontology itself, since the standard OWL inference rules are part of any inference process.

In the next section, to clarify our approach, we provide a few examples, in which we are using both generic inference rules and personalized ones.

More information about the single tools and framework could be found in: CODA and PEARL refer to [3] and [4] respectively, to check all the capabilities of ontology editor Semantic Turkey consult [5] and, to see what it is possible by adopting the reasoner HORUS see [6]. In this paper, we mainly focus our attention on the interaction between these tools (especially on the last part of the proposed architecture: the use of a dedicated and interactive reasoner to infer new knowledge from already annotated and existing one).

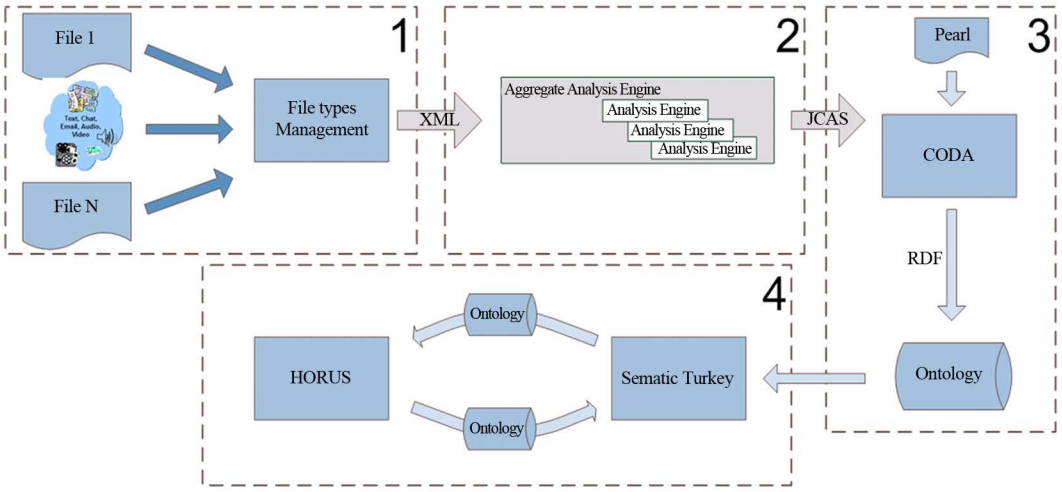

Figure 1. Heterogeneous data management architecture.

4. HORUS as an Interactive Intelligent System

We will introduce HORUS facilities step by step willing to stress how different kind information may be extracted in previously described framework. Let us start by applying HORUS to an ontology and using only the standard inference rules, defined in RDF, RDFS and OWL. Then we show how, by defining *ad hoc* rules, which have a certain meaning only in the current domain for the particular user, who is defining and applying such rule, he is able to extract and motivate new knowledge. Each HORUS rule consists of a list of premises and at least one conclusion. When all the premises are true, then the conclusion is inferred and the newly generated triples are added to the ontology in a separated graph; in such a way they can be immediately distinguished from the original RDF triples or even deleted with a single request from HORUS itself. Each premise can be either an RDF triple or a Boolean condition (this language is inspired by SPARQL, with its RDF triples and its filters; sorry for being a little bit cryptic).

As a first case, consider that HORUS relies on the functional property, which has the following premises:

- ?p rdf:type owl:FunctionalProperty
- ?x ?p ?y
- ?x ?p ?z
- ?y != ?z

and the conclusion:

- ?y owl:sameAs ?z

This mean that if a property is defined as functional, and two different resources are the objects of the functional property for the same subject, then a reasoner will always infer that the two objects are the same resource. If we have the following RDF triples in the ontology:

- :spouse rdf:type owl:FunctionalProperty
- :Maria_Teresa_Pazienza :spouse :Gigi_Fusco
- :Maria_Teresa_Pazienza :spouse :Luigi_Fusco

the reasoner will infer

- :Gigi_Fusco owl:sameAs :Luigi_Fusco

as show in **Figure 2**.

This is a first important result: we are *unifying* different entities that, possibly, appear on the web in very different contexts (e.g. Luigi Fusco in formal frameworks, while Gigi Fusco in familiar informal ones). This inference relies on the fact that CODA generated the exact same resource for the two subjects in the used triples (:Maria_Teresa_Pazienza). If the two subjects were two different RDF resources, then this inference would have failed. In the current implementation, there is no specific component which is able to perform the Named Entity Recognition (NER) task, so it is possible that the same Named Entity, could be represented by two distinct RDF resources (for example, :M_T_Pazienza and :Maria_Teresa_Pazienza are two distinct resources). By adding to this architecture a Named Entity Recognition (NER) system, such as OKKAM [7] (or its implementation inside Semantic Turkey, Maskkot [8]) this problem could be avoided.

As a second case, consider info published usually by any company in Linked Open Data standard and related to their employs; we show how a user-defined rule can be used to infer new knowledge. Suppose we want to state that when a person earns more than a certain amount of money (1000 in this case), he can be classified as a RichMan (it belongs to the class RichMan). First of all, we define the rule with the following premises:

- ?x :salary ?salary

  ?salary >= "1000"^^xsd:integer

and the conclusion:

- ?x rdf:type :RichMan.

Having this ad-hoc inference rule, and the single following triple:

- :John_Doe :salary “5000”^^xsd:integer

HORUS will infer the triple:

- :John_Doe rdf:type :RichMan

as shown in **Figure 3**.

Consider that these rules are independent while specific to an application, they do not relate to ontological features. They may be defined for *ad hoc* reasoning purposes. The system remains as it is, and may be used for different applications just changing the set of rules.

In the third example, we stress how to deal with a common problem when using an ontology: time passing

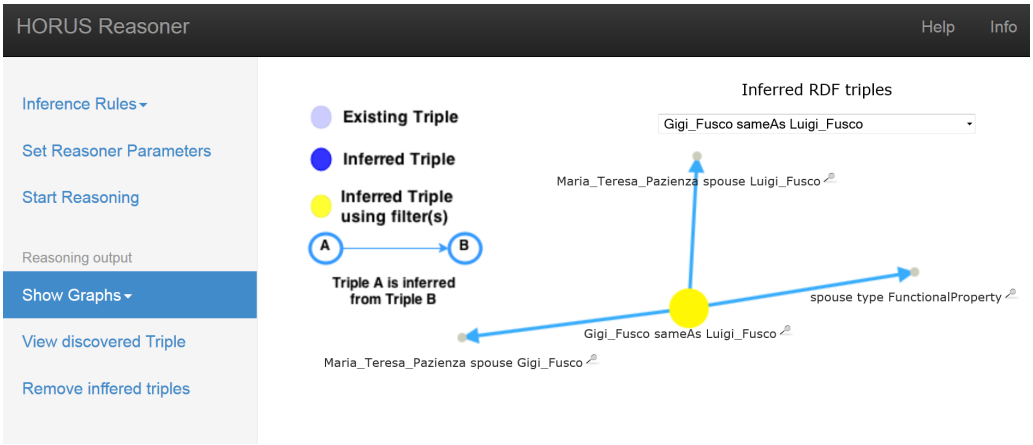

Figure 2. HORUS example: functional rule.

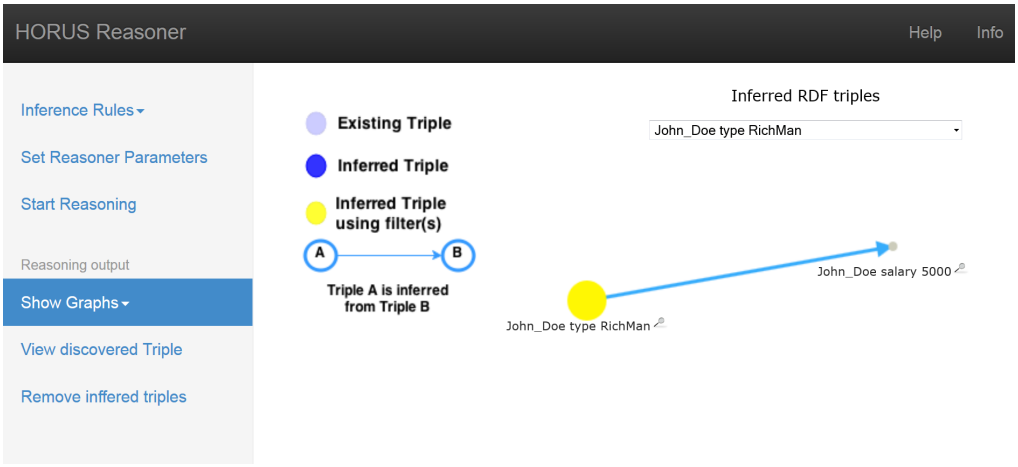

Figure 3. HORUS example: RichMan rule.

and how the stored knowledge could change when certain conditions are met. Nowadays it is common for everyone to have more than one email address, some of them are personal (so they tend to remain always the same) while others are associated to a specific job. When someone retires, his work-email is no longer valid, but it can still be found on several documents on the web. By simply removing it from the ontology, it can happen that when processing other documents, the work email can be added a second time as a result of the knowledge extraction process.

A better solution will be to mark as current-email the personal email, which is more likely to be the right one, when someone is retired. By defining in HORUS a rule with the following premises:

- ?x :workEmail ?we
- ?x :personalEmail ?pe
- ?x rdf:type :RetiredPerson

and the conclusion:

- ?x: currentEmail ?pe

HORUS is able to infer what we have just said, that for a retired person, his current email should be his personal email. By applying this rule to the following RDF triples:

- :Luigi_Fusco :workEmail “luigi.fusco@esa.int”
- : Luigi_Fusco :personalEmail “fusco.luigi@gmail.com”
- : Luigi_Fusco rdf:type :RetiredPerson

HORUS will infer:

- Luigi_Fusco :currentEmail “fusco.luigi@gmail.com”

As can be seen (with the used triple) in **Figure 4**.

This case is different from unification of entities: it emerges that information is "*changing over time*" for (semantic) validity while not being associated to an explicit "time" feature.

As a final case, we consider how to apply more than one inference rule. The rules that have been applied are either standard OWL rule or *ad hoc* rule. The standard rule states that an instance of a subclass can be inferred to be an instance of the super class as well. This rule has the following premises:

- ?x rdf:type ?class1
- ?class1 rdf:subClassOf ?class2

And one conclusion:

- ?x rdf:type ?class2

The *ad hoc* rule is used, for example, to define when a particular instance belongs to the class FirstLady. Is has the premises:

- ?x :hasHusband ?y
- ?y rdf:type :President

And the conclusion:

- ?x rdf:type :FirstLady

Let's assume that, in the ontology, we have the following RDF triples:

- :Barack_Obama rdf:type :CurrentPresidentUSA
- :Michelle_Obama :hasHusband :Barack_Obama

From these two RDF triple, HORUS, applying the first inference rule, is able to infer:

- :Barack_Obama rdf:type : President

This means that, since Barack Obama is the current president of the USA, he belongs also to the class President (which, according to the knowledge in the ontology, represents any past and present President of any country). By exploiting this new knowledge, HORUS is now able to use the second rule to infer that Michelle Obama (being the wife of a President) can be associated to the class FirstLady. The inference graph, for this example, can be seen in **Figure 5**. This use case is useful to understand and show two main aspects of applying inferences rules:

- It is possible to use already inferred RDF triple to infer new knowledge and it is almost mandatory to use reasoners, since the discovery of new knowledge can be a complex task for end users, as they should consider also the knowledge which has not been inferred yet.
- By looking at the inference graph, the end user is able to see all the used knowledge in the reasoning process and in which order they were used.

5. Conclusions

In this paper, we have described an architecture for managing heterogeneous data to extract new info as well as unforeseen knowledge. The core idea behind this architecture is to provide final users with an easy to use, yet

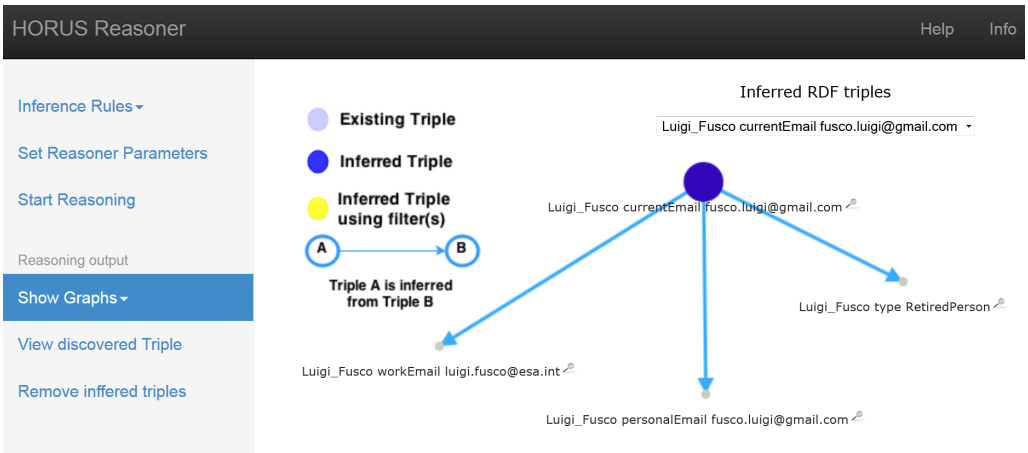

Figure 4. HORUS example: email rule.

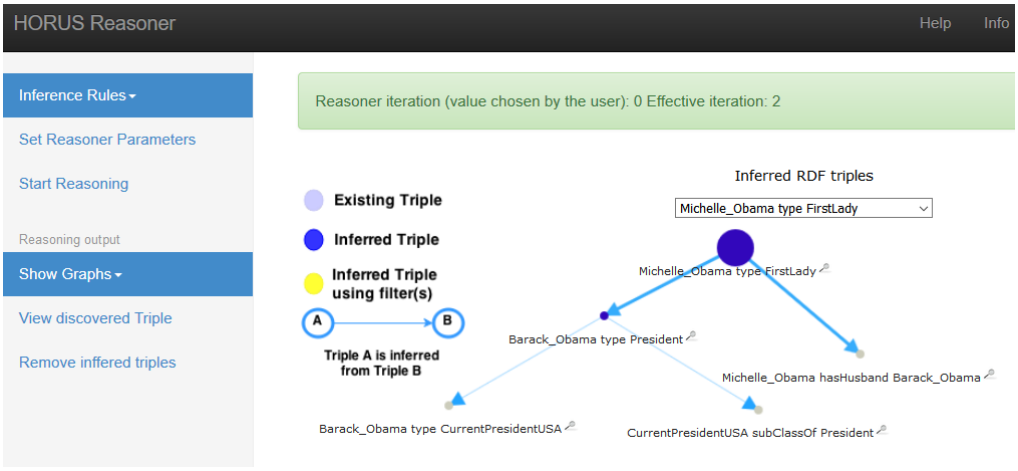

Figure 5. HORUS example: subClass and FirstLady rules.

powerful set of tools to assist them in the extraction process of relevant data (given a specific domain) from a set of heterogeneous documents officially published by different entities. The extracted data is then formalized and placed in an ontology.

The innovation of this architecture resides in using a customizable reasoner to infer new knowledge, which represents a specific meaning just for specific users. This is achievable using HORUS and its feature of defining *ad-hoc* inference rules, through a dedicated GUI. Thanks to such an architecture (and mainly to HORUS) users, starting from the same initial documents, may develop an ontology, which satisfies their own personal requirements to check for correctness and validity of data freely accessible on the web thus maintaining also a constant control on them.

References

[1] Knap, T., Michelfeit, J. and Necaský, M. (2012) Linked Open Data Aggregation: Conflict Resolution and Aggregate Quality. *COMPSAC Workshops*, Izmir, 16-20 July 2012, 106-111. http://dx.doi.org/10.1109/compsacw.2012.29

[2] Ferrucci, D. and Lally, A. (2004) Uima: An Architectural Approach to Unstructured Information Processing in the Corporate Research Environment. *Natural Language Engineering*, **10**, 327-348. http://dx.doi.org/10.1017/S1351324904003523

[3] Fiorelli, M., Pazienza, M.T., Stellato, A. and Turbati, A. (2014) CODA: Computer-Aided Ontology Development Architecture. *IBM Journal of Research and Development*, **58**, 14: 1, 14: 12. http://dx.doi.org/10.1147/jrd.2014.2307518

[4] Pazienza, M.T., Stellato, A. and Turbati, A. (2012) PEARL: ProjEction of Annotations Rule Language, a Language for Projecting (UIMA) Annotations over RDF Knowledge Bases. *International Conference on Language Resources and Evaluation* (*LREC* 2012), Instanbul, 2012, 3828-3835.

[5] Pazienza, M.T., Scarpato, N., Stellato, A. and Turbati, A. (2012) Semantic Turkey: A Browser-Integrated Environment for Knowledge Acquisition and Management. *Semantic Web*, **III**, 279-292.

[6] Napoleoni, G.L., Pazienza, M.T. and Turbati, A. (2014) HORUS: A Configurable Reasoner for Dynamic Ontology Management. *COGNITIVE* 2014, Venice, 2014, 66-71.

[7] Bouquet, P., Stoermer, H. and Xin, L. (2007) Okkam4P—A Protégé Plugin for Supporting the Re-Use of Globally Unique Identifiers for Individuals in OWL/RDF Knowledge Bases. *Proceedings of the 4th Italian Semantic Web Workshop* (*SWAP* 2007), Bari, 18-20 December 2007.

[8] Stellato, A., Stoermer, H., Bortoli, S., Scarpato, N., Turbati, A., Bouquet, P. and Pazienza, M.T. (2010) MASKKOT: A Tool for Annotating Entities through the OKKAM Service. *Proceedings of the 6th Workshop on Semantic Web Applications and Perspectives* (*SWAP* 2010), Bressanone, 21-22 September 2010.