Author-produced version of the paper presented at 6ème atelier Recherche d'Information SEmantique RISE, SDNRI 2014, March 18th, 2014, Nancy, France.
The original publication is available at https://sites.google.com/site/frenchsemanticir/documents

An Attempt to Use Ontologies for Document Image Analysis

Bart Lamiroy[1] and Shabai Zheng[2]

[1] Université de Lorraine – LORIA (UMR 7503)
Campus Scientifique – BP 239, 54506 Vandoeuvre-lès-Nancy CEDEX, France
Bart.Lamiroy@loria.fr
[2] Université de Lorraine – École Nationale Supérieure des Mines de Nancy
Campus ARTEM – CS 14 234, 92 Rue Sergent Blandan, 54042 Nancy, France

Abstract. This paper presents exploratory work on the use of semantics in Document Image Analysis. It is different than existing semantics-aware approaches in the sense that it approaches the problem from a a very domain specific angle, and tries to incorporate an open model based on a reduced ontology. As presented here, it consists of enhancing an existing platform for Document Image Analysis benchmarking using off-the-shelf tools. The platform on which it is based hosts a wide variety of image interpretation algorithms as well as a wide range of benchmarking data. These data are stored in a relational database, as well as their type definition, the association between data and algorithms, *etc.* This work tries to provide an experimental indication whether ontologies and automated reasoning can provide new or alternative ways to extract relations among different stored facts, or infer dependencies between various user-defined types, based on their interactions with algorithms and other types of data.

1 Introduction

The aim of this paper is to report preliminary experimental setups related to introducing open semantics in Document Image Analysis (DIA). The topics described relate to a benchmarking platform, specifically conceived for DIA research, and stem from the need of intelligently handling a specific kind of semantics, as will be made clear later. The DAE Platform[1], on which this work is based, provides an experimental environment in which users can upload algorithms, store relevant data, conduct experiments and compare their results. This system is conceived around a flexible and open data model that links together document images, algorithms, user-defined interpretations and user information. All is stored in a relational database. Among these, the most important is a wide range of data related to document analysis algorithms that can generally be divided into user-defined data types as inputs of algorithms and automatic interpretations as results. However, the flexibility of this model in terms of interpretations and data types is one of its main weaknesses. One may define a type without considering that it may have other equivalent types in the system defined by others. Moreover, sometimes existing

types may form super types or subtypes of the one the user intended to create. The goal of this research consists in developing a semantic model of the knowledge stored in the database, which will allow automatic reasoning engines to infer dependencies between various user-defined types, based on their interactions with algorithms and other data types. One important step in this direction is to represent the underlying data model as an ontology.

The next section will explain in further details the goals of the benchmarking platform, the specific need for semantics it has, and how it differs from other existing platforms. In section 3 we explain the techniques used to integrate ontologies into the model. Section 4 provides some experimental results.

2 Context

2.1 DAE-platform

The DAE-Platform's [1] primary goal is to serve as a reference repository for reproducible experimental research in Document Image Analysis, and as such offers access to reference data, state-of-the-art algorithms, and benchmark annotations. It enables registered users to upload new reference material, to define, extend or correct the interpretation of stored data and aims to support experimental research through an open and evolving framework that can be extended by community driven contributions.

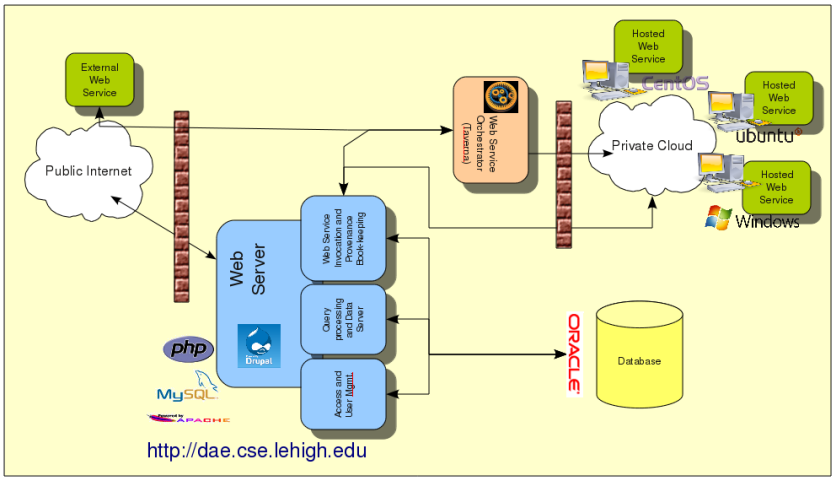

Fig. 1. Architecture of the DAE Platform

Author-produced version of the paper presented at 6ème atelier Recherche d'Information SEmantique RISE, SDNRI 2014, March 18th, 2014, Nancy, France.
The original publication is available at https://sites.google.com/site/frenchsemanticir/documents

Previous accounts of the features and the potential uses of the platform [2,3] focused on its architecture, implementation choices, and its potential to impact experimental research, especially with respect to reproducibility and accountability. The platform has been implemented using a classical LAMP architecture [4] but also integrates a full WSDL interface [5], allowing it to be integrated in a fully distributed service-oriented architecture. The DIA reference data and annotations themselves are stored in an Oracle relational database back-end. An independent cluster of servers is used to execute stored algorithms. Fig 1 shows its structure.

2.2 Data model

Besides the technical architecture of the server, as depicted in Fig 1, the DAE Platform is an implementation of a data model [6] .The data model is based on the following claims: all data is typed; users can define new types;

- Data can be attached to specific parts of a document image (not obliged)
- Both data and algorithms are modelled; algorithms transform data from one type into data of another type;
- Full provenance of data history is recorded.

The result is that the platform can be used as a benchmarking repository. Since all provenance is recorded, one can leverage it to compare results from algorithms on identical data, and compare annotations by different users.
A simplified representation of the data model is represented in Fig 2. It consists of three key elements: `algorithms, data_items` and `algorithm_runs`.

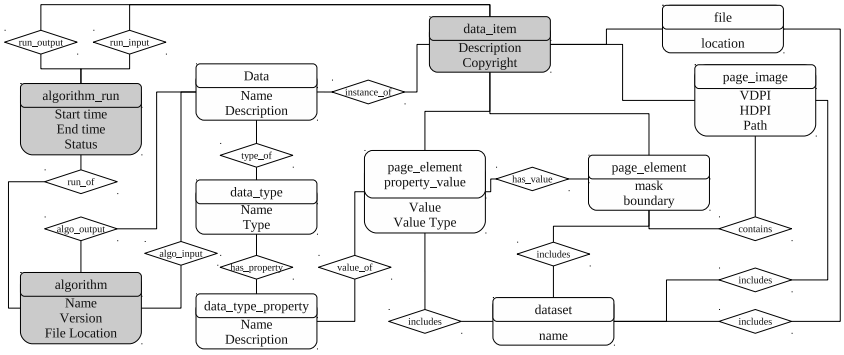

Fig. 2. The DAE Data Model (simplified)

The underlying reasoning is that data is transformed by algorithms. `data_items` are instances of data and are related to algorithms by explicit `algorithm_runs`. New `data_items` thus produced are stored in the

database with the exact information of how they were obtained. There are also pre-defined types of `data_items: page_images, page_elements, page_element_properties, files` and `datasets`. In this document we focus on the first three.

- `page_image` is an image file representing a physical page at a given resolution and with a given image quality. It is perfectly possible to have multiple `page_images` representing the same physical page, as shown in Fig 2;
- `page_element` is an area of a `page_image`, defined in as unconstrained a way as possible, either by a bounding box or a pixel map
- `page_element_property` are any kind of user-defined property or interpretation attached to a `page_element`.

3 Semantics and Ontologies

3.1 A Specific Kind of Semantics

As already mentioned, the flexibility of DAE platform causes polysemy. Unlike other image annotation initiatives, our main goal is not to try and relate visual representations to higher level known or fixed ontologies [7]. Although we will eventually benefit from connecting our ontology with it, it is currently not within the scope of this paper.
Our main concern comes from the fact that different users very likely operate in different interpretation contexts, which inevitably may result in several types for the same visual element. We have already shown that this is a normal, and necessary "feature" in machine perception and interpretation related topics [8]. The problem is, the system cannot automatically deduce or indicate the relation among these interpretations. Our goal is to try and investigate if an appropriate ontology, capturing the semantics of our benchmarking data can help resolve this.

3.2 Example

In order to illustrate our point, let us develop the example depicted in Fig. 3. It represents the platform with various data objects and algorithms, created by different users (researchers).
We assume that one of the stored algorithms was contributed by researcher A, who defined its input as being of type X and its output of type Y. Another researcher B successfully execute this algorithm with, as input, a specific data item i whose type was previously defined as being Z.
It is straightforward to infer that:

$$\begin{cases} \mathrm{Type}(Z) \cap \mathrm{Type}(X) \neq \emptyset \\ i \in \mathrm{Type}(Z) \cap \mathrm{Type}(X) \end{cases}$$

The fact that types Z and X are considered being different may come from either having different users using different name labels for the same semantics (*e.g.* "text area" and "text zone"), or it may result

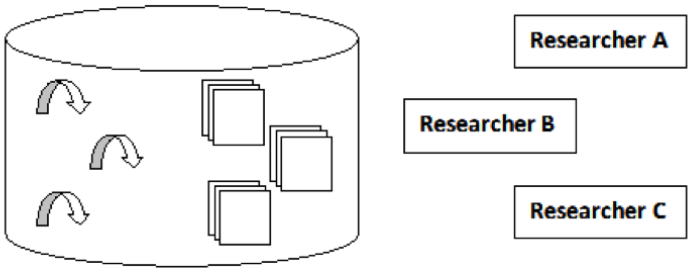

Fig. 3. Various Data Used by Different Users

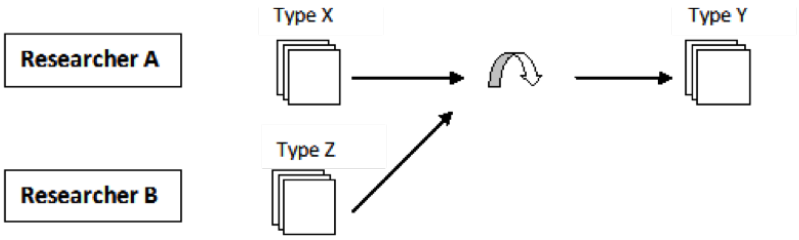

Fig. 4. Different Data Used by the Same Algorithm

from having both users using slightly different interpretation contexts with overlapping, yet not fully identical semantics. This situation is neither far-fetched, nor unique. Since the researches in domain of document analysis can be very specific (e.g. concentrate in translating a file from a particular form to another) and the domain itself is becoming more and more complex, some researchers may work on quite specialised subproblems, that may eventually prove to be instances of a more generic problem. In order to detect this, we need to extract the dependency between equivalent user-defined types.

3.3 Semantics of the data model

In the DAE Data Model, algorithms are declared having precise `data_type`s for their input and output values (called `data`). However, an algorithm can have any number of executions (`algorithm_run`s), associating specific `data_item`s. In other words, specific `data` (algorithm inputs/outputs) may possess various instances at run-time. The problem discussed above lies in the fact that, a given `data_item` can be sometimes applied in executions of different algorithms, which means it serves as `data` of potentially different types. Conversely, this means that once we find all the `data_type`s of a `data_item`, we can easily trace to every algorithms where this `data_item` is involved. Our study is based on this point. Therefore, one important issue is to establish the link from specific `data` to each of its instance `data_item`s.

Author-produced version of the paper presented at 6ème atelier Recherche d'Information SEmantique RISE, SDNRI 2014, March 18th, 2014, Nancy, France.
The original publication is available at https://sites.google.com/site/frenchsemanticir/documents

Given the architecture of the DAE platform, we could achieve this using SQL queries. However, this solution is difficult to generalise: every time when we want to check a particular `data_item`, we need to modify the query statement (e.g. `data_item`'s id). Since the number of `data_items` is enormous, using SQL complicated, tedious and error prone. Introducing semantic querying provides a more readable presentation for complicate hierarchical relationships between information. Furthermore, DL reasoners will allow us to infer a transitive chain of a series of relations between `data_type` and `data_item`.

3.4 D2R

To achieve this, we will be using D2R [9]. D2R Server is a tool for publishing the content of relational databases on the Semantic Web. Database content is mapped to RDF by a declarative mapping which specifies how resources are identified and how property values are generated from database content. D2R Server uses a specific language[3] to express mappings between application-specific database schemas and RDFS schemas or OWL. They specify how resources are identified and how property values are generated from database content. The central object in D2RQ is the *ClassMap*. A *ClassMap* represents a mapping from a set of entities described within the database, to a class or a group of similar classes of resources. Each *ClassMap* has a set of *PropertyBridges*, which specify how resource descriptions are created. Property values can be created directly from database values or by employing patterns or translation tables. D2RQ supports conditional mappings on *ClassMap* and *PropertyBridge* levels, mapping of multiple columns to the same property and the handling of highly normalised table structures where instance data is spread over multiple tables. It also includes a tool that automatically generates a D2RQ mapping from the database table structure, generating the appropriate RDF for each database, using table names as class names and column names as property names.

4 Experiments and Analysis

4.1 Matching `data` and `data_items`

As mentioned above, given a specific algorithm execution allows to relate a `data_item` to `data` properties. We know that an algorithm is defined as: ($output_1$, $ouput_2$, ... , $output_n$) = algorithm ($input_1$, $input_2$, ... , $input_m$) Where $input_i$ (i = 1 ... m) is defined as $\texttt{data}_i$, who possess a type $\texttt{data\_type}_i$. In the same way, $output_j$(j = 1 ... n) is defined as $\texttt{data}_j$ of $\texttt{data\_type}_j$. These data are associated directly with the algorithm in a strict order. This position information is available in our provenance data from the DAE platform. Fig. 5 we see the logical order of the inference. We have implemented this inference chain in Protégé[4]. Fig. 7 shows that we are capable of correctly inferring new types on data items.

[3] D2RQ Mapping Language: http://d2rq.org/d2rq-language
[4] http://protege.stanford.edu/

Author-produced version of the paper presented at 6ème atelier Recherche d'Information SEmantique RISE, SDNRI 2014, March 18th, 2014, Nancy, France.
The original publication is available at https://sites.google.com/site/frenchsemanticir/documents

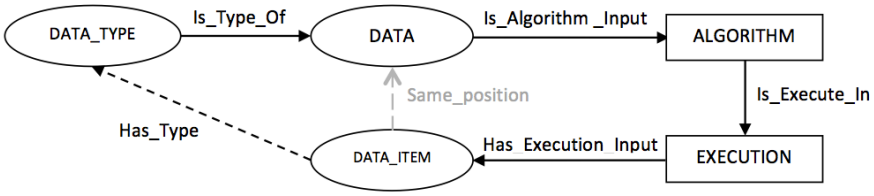

Fig. 5. Inference of Data Types from Algorithm Executions

Up to this point, we can conclude that if all relevant properties are properly defined(extracted), the desired deduction set by inference chain will be established automatically on the whole model after running the DL reasoning engine. However, after the implementation and tests, several issues left to be resolved:

- Complexity: a chain of 4 properties slows down the reasoning;
- Limited by the fact that the enforcement of argument positioning requires reification of the relations (our example is limited to two inputs/outputs only);
- We failed to realize the extraction by ordering: Because the D2RQ mapping language cannot yet function normally on conditional mapping.

Besides its inconveniences and technical limitations, this semantic model functions as expected, so the applied methods proved to be correct. Moreover, it proposes an important usage of the ontology modeling: verification of complex relations in relational database.

4.2 Simplification

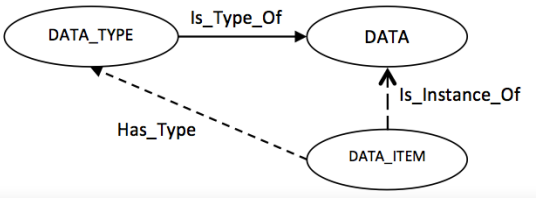

Fig. 6. Simplified Inference of Data Types from Algorithm Executions

In order to avoid all the problem occurred in the first approach and to provide a functional model, we decided to replace the ordering information by storing a direct link between `data_item` and `data_type` in our provenance database. Every time when an algorithm runs, we record each of the `data_items` used as its inputs/outputs, along with the information

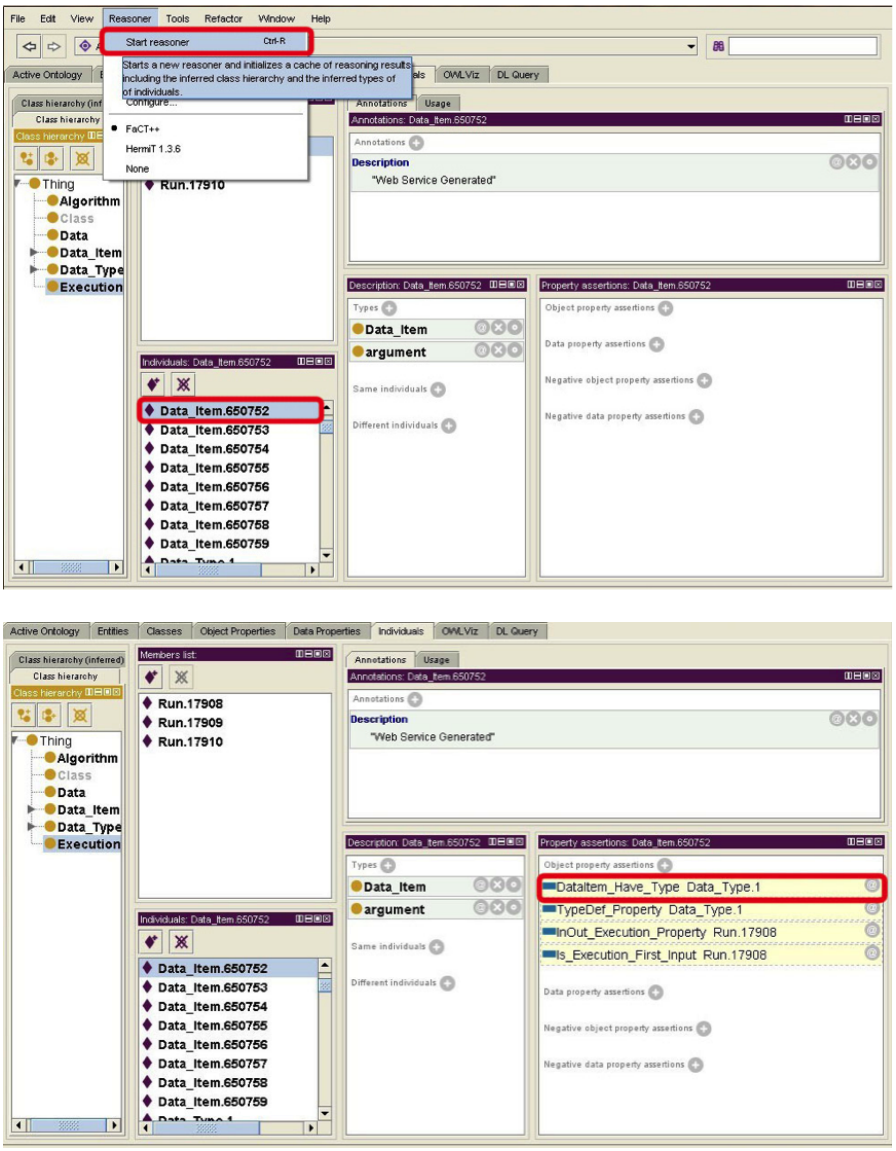

Fig. 7. Results of Type Inference from Algorithm Executions using Protégé

of this algorithm in the database. Besides, the corresponding data are also recorded. The resulting semantics is shown in Fig. 6.
This model provides the solution in a most simplified way with only three classes and is more efficient than the one in the first approach.

4.3 About Performance Analyisis

Knowing that using the ontology model for queries can provide an alternative to relational database, we would like to examine its performance comparing to traditional SQL in a given case, which is described below. The approach described here is fairly naive and would require a more thorough experimental investigation. For one, using a reasoner is unfair, and definitely not the optimal approach. Further work will necessarily need to include benchmarking with triple-store databases [10,11].
However, to show the importance of correctly choosing tools and approaches, we will develop the comparative experimentation we have conducted, even though it is open to criticism. As mentioned before [6], a `page_image` consists of several `page_element`s (*cf.* Fig. 2). A `page_element` itself is a `data_item`, but it usually contains "sub" `data_item`s. In fact, it is defined as "a physical sub-part of an image" and its basic description is only its geometry. Furthermore, attached to them are `page_element_property_value`s to record "interpretations" (*e.g.* OCR results, layout labels...). Meanwhile, these interpretation also have a `data_type`, *etc.* ending up by composing a complex network of semantic relations.
Let us now consider the following example: suppose there is a `page_element` whose `data_type` is "textline" (an area containing a line of text), and it have several `page_element_property_value`s such as "number of the next line" and "transcription" (*i.e.* its literal content). In that case, the `data_type` of "number of the next line" would be "page_element id" or simply "integer", for instance; the `data_type` of "transcription" would be "String"; "transcription" would also be considered as a specific property.
So what if we need to know all annotations and their types attached to a given `page_image`? Seen from a part of the data model shown in Fig. 8, this logic is not directly reflected in the current database and requires a series of SQL join operations to extract the list of all `data_type`s attached to a given `page_image`.

```
SELECT pi.ID AS PAGE_IMAGE_ID,dt.ID AS Data_Type_ID
FROM PAGE_IMAGE pi
LEFT JOIN CONTAINS_PAGE_ELEMENT cpe ON
           pi.ID = cpe.PAGE_ELEMENT_ID
LEFT JOIN PAGE_ELEMENT_UNDERLYING pe ON
           cpe.PAGE_ELEMENT_ID = pe. ID
LEFT JOIN HAS_VALUE hv ON pe.ID = hv.PAGE_ELEMENT_ID
LEFT JOIN PAGE_ELEMENT_PROPERTY_VALUE pe_pv ON
           hv.PAGE_ELEMENT_PROPERTY_VALUE_ID  = pe_pv.ID
LEFT JOIN VALUE_OF vo ON
```

Author-produced version of the paper presented at 6ème atelier Recherche d'Information SEmantique RISE, SDNRI 2014, March 18th, 2014, Nancy, France.
The original publication is available at https://sites.google.com/site/frenchsemanticir/documents

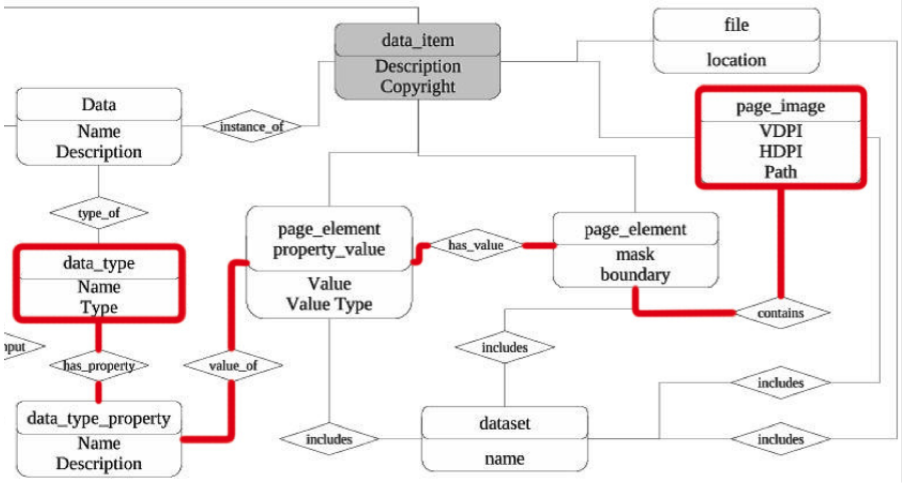

Fig. 8. Representation of required join operations to extract all annotations (and their types) of a `page_image`

```
          pe_pv.ID = vo.PAGE_ELEMENT_PROPERTY_VALUE_ID
LEFT JOIN DATATYPE_PROPERTY dp ON
          vo.DATA_TYPE_PROPERTY_ID = dp.ID
LEFT JOIN HAS_PROPERTY hp ON dp.ID = hp.DATA_PROPERTY_ID
LEFT JOIN DATATYPE dt ON hp.DATA_TYPE_ID = dt.ID
```

We have tested this query with eight join operation between nine tables. With about 20,000 `page_image`s, the wall clock execution time (inter-continental network traffic included) of this query is around 5 seconds.

We can also express the join using a property chain in Protégé and use reasoners to infer the information.

DataType_Have_Property o DataType_Property_Have_Value o Value_Of_Element o Element_Of_Image → DataType_Appears_In_Image

It is notable that the running speed of launching the reasoner largely depends on the size of the extracted initial dataset. For example, in a predigest version of the database with 50 `page_image`s on a personal computer, the running time is around 1.4 seconds; but if the number of `page_image`s increase to 2000, the reasoning will take more than 20 minutes. Also, the choice of reasoner in this experiment can influence the result. During the experiment, we could obtain the expected test result with FaCT++ [12] and Hermit (1.3.6) [13]. However, with RacerPro [14], the property chain failed to work.

Although we are aware of the limited scope of the above experiment, it does highlight some interesting points. The SQL query is complicated, requires in depth knowledge of the data model and not easily reusable. However, it is very efficient. The reasoner approach is much less efficient and dependant on the underlying reasoner implementation or technique. However, it reformulates the intricate table structures into a concise graph. Moreover, the query expression of model is highly reusable as it is very simple to define new properties based on existing ones.

5 Conclusion

In this work, we have demonstrated that the use of semantics opens new perspectives of looking at document image annotation and benchmarking repositories, since it allow to infer and extract new kinds of relations between data. This can be used as a basis of trying to uncover whether different algorithms or different experimental setups share similar interpretation contexts, and thus discover new relations between various interpretations of the same data. Our approach offers advantages compared to the previously existing tool. Relations can be more easily created, modified and integrated using the appropriate an ontologies. In the domain of document analysis research, the ontology modelling is helpful in inferring the dependency between existing algorithms developed by individual researchers. Moreover, it can be considered as an efficient way to verify existing relationships between tables in the database.

Acknowledgements

Part of this work was based on previous, unpublished work on configuring a D2RQ server, by students A. Hombourger, P. Lauffenburger and J. Pruvost. Without their technical help, this work would never have been possible.

References

1. Lamiroy, B., Lopresti, D., Korth, H., Heflin, J.: How Carefully Designed Open Resource Sharing Can Help and Expand Document Analysis Research. In Gady Agam, C.V.G., ed.: Document Recognition and Retrieval XVIII - DRR 2011. Volume 7874., San Francisco, United States, SPIE, SPIE (January 2011)
2. Lamiroy, B., Lopresti, D.: An Open Architecture for End-to-End Document Analysis Benchmarking. In: 11th International Conference on Document Analysis and Recognition - ICDAR 2011, Beijing, China, International Association for Pattern Recognition, IEEE Computer Society (September 2011) 42–47
3. Lamiroy, B., Lopresti, D.: A Platform for Storing, Visualizing, and Interpreting Collections of Noisy Documents. In: Fourth Workshop on Analytics for Noisy Unstructured Text Data - AND'10. ACM International Conference Proceeding Series, Toronto, Canada, IAPR, ACM (October 2010)

4. Wikipedia: Lamp (software bundle) — Wikipedia, the free encyclopedia (2008) [Online; accessed 01-Sep-2008].
5. Chinnici, R., Moreau, J.J., Ryman, A., Weerawarana, S.: Web services description language (wsdl) version 2.0 part 1: Core language. World Wide Web Consortium, Recommendation REC-wsdl20-20070626 (June 2007)
6. Korth, H.F., Song, D., Heflin, J.: Metadata for structured document datasets. In: Proceedings of the 8th IAPR International Workshop on Document Analysis Systems. 547–550
7. Halaschek-Wiener, C., Golbeck, J., Schain, A., Grove, M., Parsia, B., Hendler, J.: Photostuff - an image annotation tool for the semantic web. In: 4th International Semantic Web Conference - Poster Paper. (2005)
8. Lamiroy, B.: Sur les limites de la perception artificielle et de l'interprétation. Hdr, Université de Lorraine (December 2013)
9. Bizer, C., Cyganiak, R.: D2r server-publishing relational databases on the semantic web. 5th international Semantic Web conference (2006) 26
10. Bizer, C., Schultz, A.: Benchmarking the performance of storage systems that expose sparql endpoints. World Wide Web Internet And Web Information Systems (2008)
11. Voigt, M., Mitschick, A., Schulz, J.: Yet another triple store benchmark? practical experiences with real-world data. In: SDA. (2012) 85–94
12. Tsarkov, D., Horrocks, I.: Fact++ description logic reasoner: System description. In: Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2006). Volume 4130 of Lecture Notes in Artificial Intelligence., Springer (2006) 292–297
13. Motik, B., Shearer, R., Horrocks, I.: Hypertableau Reasoning for Description Logics. Journal of Artificial Intelligence Research **36** (2009) 165–228
14. Haarslev, V., Hidde, K., Möller, R., Wessel, M.: The racerpro knowledge representation and reasoning system. Semantic Web **3**(3) (2012) 267–277