



## **LarKC**

*The Large Knowledge Collider  
a platform for large scale integrated reasoning and Web-search*

**FP7 – 215535**

---

### **D3.1**

## **Survey of Relevant Literature on Abstraction and Learning**

---

**Volker Tresp (Coordinator, Siemens), Yi Huang (Siemens), Emanuele Della Valle (CEFRIEL), Daniele Braga(CEFRIEL), Davide Barbieri(CEFRIEL) and Stefano Ceri(CEFRIEL)**

|                             |                          |
|-----------------------------|--------------------------|
| <b>Document Identifier:</b> | LarKC/2008/D3.1          |
| <b>Class Deliverable:</b>   | LarKC EU-IST-2008-215535 |
| <b>Version:</b>             | version 1.0              |
| <b>Date:</b>                | December 18, 2008        |
| <b>State:</b>               | final                    |
| <b>Distribution:</b>        | public                   |

## EXECUTIVE SUMMARY

In this document we explore some of the opportunities and challenges for machine learning and data streaming on the Semantic Web. The Semantic Web provides standardized formats for the representation of both data and ontological background knowledge. Semantic Web standards are used to describe meta data but also have great potential as a general data format for data communication and data integration. Within a broad range of possible applications machine learning and data streaming will play an increasingly important role:

- Machine learning solutions have been developed to support the management of ontologies, for the semi-automatic annotation of unstructured data, and to integrate semantic information into web mining.
- Data Streaming solutions consider a type of data common in many real-world applications (e.g. monitoring of network traffic, telecommunications management, clickstreams, manufacturing, sensor networks, etc.) that is not yet part of the Semantic Web.

Machine learning will increasingly be employed to analyze distributed data sources described in Semantic Web formats and to support approximate Semantic Web reasoning and querying. In this document we discuss existing and future applications of machine learning on the Semantic Web with a strong focus on learning algorithms that are suitable for the relational character of the Semantic Web's data structure. We discuss some of the particular aspects of learning that we expect will be of relevance for the Semantic Web such as scalability, missing and contradicting data, and the potential to integrate ontological background knowledge. In addition we review some of the work on the learning of ontologies and on the population of ontologies, mostly in the context with textual data.

In this document we also provide an overview of state-of-the-art data streaming technologies for data abstraction and we propose a way to plug data streaming within the algorithmic framework of LarKC through a continuous query language for the Semantic Web (named C-SPARQL). Such continuous query language extends SPARQL with data streams, aggregation operators, and a continuous semantics.



## DOCUMENT INFORMATION

|                           |   |                |       |
|---------------------------|---|----------------|-------|
| <b>IST Project Number</b> | FP7 – 215535  | <b>Acronym</b> | LarkC |
| <b>Full Title</b>         | Large Knowledge Collider                                |                |       |
| <b>Project URL</b>        | <a href="http://www.larkc.eu/">http://www.larkc.eu/</a> |                |       |
| <b>Document URL</b>       |   |                |       |
| <b>EU Project Officer</b> | Stefano Bertolo   |                |       |




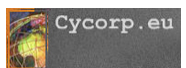







|                     |               |     |              |   |
|---------------------|---------------|-----|--------------|---|
| <b>Deliverable</b>  | <b>Number</b> | 3.1 | <b>Title</b> | Survey of Relevant Literature on Abstraction and Learning |
| <b>Work Package</b> | <b>Number</b> | 3   | <b>Title</b> | Abstraction and Learning                                  |

|                            |  |    |               |                                     |
|----------------------------|--|----|---------------|-------------------------------------|
| <b>Date of Delivery</b>    | <b>Contractual</b>   | M6 | <b>Actual</b> | 30-Sep-08                           |
| <b>Status</b>              | version 1.0  |    | final         | <input checked="" type="checkbox"/> |
| <b>Nature</b>              | prototype <input type="checkbox"/> report <input checked="" type="checkbox"/> dissemination <input type="checkbox"/> |    |               |                                     |
| <b>Dissemination Level</b> | public <input checked="" type="checkbox"/> consortium <input type="checkbox"/>                                       |    |               |                                     |


|                          |  |            |               |                          |
|--------------------------|--|------------|---------------|--------------------------|
| <b>Authors (Partner)</b> | Volker Tresp (Siemens AG), Yi Huang (Siemens AG), Emanuele Della Valle (CEFRIEL), Daniele Braga (CEFRIEL), Davide Barbieri(CEFRIEL), Stefano Ceri(CEFRIEL) |            |               |                          |
| <b>Resp. Author</b>      | Volker Tresp   |            | <b>E-mail</b> | volker.tresp@siemens.com |
|                          | <b>Partner</b>   | Siemens AG | <b>Phone</b>  | +49 (89) 636-49408       |

|  |  |
|--|--|
| <p><b>Abstract<br/>(for dissemination)</b></p> | <p>In this document we explore some of the opportunities and challenges for machine learning and data streaming on the Semantic Web. The Semantic Web provides standardized formats for the representation of both data and ontological background knowledge. Semantic Web standards are used to describe meta data but also have great potential as a general data format for data communication and data integration. Within a broad range of possible applications machine learning and data streaming will play an increasingly important role: a) Machine learning solutions have been developed to support the management of ontologies, for the semi-automatic annotation of unstructured data, and to integrate semantic information into web mining. b) Data Streaming solutions consider a type of data common in many real-world applications (e.g. monitoring of network traffic, telecommunications management, clickstreams, manufacturing, sensor networks, etc.) that is not yet part of the Semantic Web.</p> <p>Machine learning will increasingly be employed to analyze distributed data sources described in Semantic Web formats and to support approximate Semantic Web reasoning and querying. In this document we discuss existing and future applications of machine learning on the Semantic Web with a strong focus on learning algorithms that are suitable for the relational character of the Semantic Web's data structure. We discuss some of the particular aspects of learning that we expect will be of relevance for the Semantic Web such as scalability, missing and contradicting data, and the potential to integrate ontological background knowledge. In addition we review some of the work on the learning of ontologies and on the population of ontologies, mostly in the context with textual data. These discussions and analysis will be published soon as a chapter titled "<i>Towards Machine Learning on the Semantic Web</i>" authored by Volker Tresp et al. in a book titled "<i>Uncertainty Reasoning for the Semantic Web I</i>" which will be published by Springer Verlag in its Lecture Notes in Artificial Intelligence (LNAI) sub-series edited by Claudia d'Amato et al.</p> <p>In this document we also provide an overview of state-of-the-art data streaming technologies for data abstraction and we propose a way to plug data streaming within the algorithmic framework of LarKC through a continuous query language for the Semantic Web (named C-SPARQL). Such continuous query language extends SPARQL with data streams, aggregation operators, and a continuous semantics.</p> |
| <p><b>Keywords</b></p>                         | <p>machine learning, data mining, data streaming</p>   |

## PROJECT CONSORTIUM INFORMATION

| Acronym   | Partner  | Contact  |
|---|--|--|
| Semantic Technology Institute Innsbruck<br><a href="http://www.sti-innsbruck.at">http://www.sti-innsbruck.at</a>                                      | STI<br>                     | Prof. Dr. Dieter Fensel<br>Semantic Technology Institute (STI)<br>Innsbruck, Austria<br>E-mail: dieter.fensel@sti-innsbruck.at |
| AstraZeneca AB<br><a href="http://www.astrazeneca.com/">http://www.astrazeneca.com/</a>   | ASTRAZENECA<br>             | Bosse Andersson<br>AstraZeneca<br>Lund, Sweden<br>E-mail: bo.h.andersson@astrazeneca.com                                       |
| CEFRIEL SCRL.<br><a href="http://www.cefriel.it/">http://www.cefriel.it/</a>  | CEFRIEL<br>                 | Emanuele Della Valle<br>CEFRIEL SCRL.<br>Milano, Italy<br>E-mail: emanuele.dellavalle@cefriel.it                               |
| CYCROP, RAZISKOVANJE IN EKSPERIMENTALNI RAZVOJ D.O.O.<br><a href="http://cyceurope.com/">http://cyceurope.com/</a>                                    | CYCROP<br>                  | Michael Witbrock<br>CYCROP, RAZISKOVANJE IN EKSPERIMENTALNI RAZVOJ D.O.O.,<br>Ljubljana, Slovenia<br>E-mail: witbrock@cyc.com  |
| Hchstleistungsrechenzentrum, Universitaet Stuttgart<br><a href="http://www.hlrs.de/">http://www.hlrs.de/</a>  | HLRS<br>                    | Georgina Gallizo<br>Hchstleistungsrechenzentrum, Universitaet Stuttgart<br>Stuttgart, Germany<br>E-mail : gallizo@hlrs.de      |
| Max-Planck-Institut fr Bildungsforschung<br><a href="http://www.mpib-berlin.mpg.de/index_js.en.htm">http://www.mpib-berlin.mpg.de/index_js.en.htm</a> | MAXPLANCKGESELLSCHAFT<br> | Michael Schooler,<br>Max-Planck-Institut fr Bildungsforschung<br>Berlin, Germany<br>E-mail: schooler@mpib-berlin.mpg.de        |
| Ontotext Lab, Sirma Group Corp.<br><a href="http://www.ontotext.com/">http://www.ontotext.com/</a>  | ONTO<br>                  | Atanas Kiryakov,<br>Ontotext Lab, Sirma Group Corp.<br>Sofia, Bulgaria<br>E-mail: atanas.kiryakov@sirma.bg                     |
| SALTLUX INC.<br><a href="http://www.saltlux.com/EN/main.asp">http://www.saltlux.com/EN/main.asp</a>   | Saltlux<br>               | Kono Kim<br>SALTLUX INC<br>Seoul, Korea<br>E-mail: kono@saltlux.com  |
| SIEMENS AKTIENGESELLSCHAFT<br><a href="http://www.siemens.de/">http://www.siemens.de/</a>   | Siemens<br><b>SIEMENS</b>  | Dr. Volker Tresp<br>SIEMENS AKTIENGESELLSCHAFT<br>Muenchen, Germany<br>E-mail: volker.tresp@siemens.com                        |
| THE UNIVERSITY OF SHEFFIELD<br><a href="http://www.shef.ac.uk/">http://www.shef.ac.uk/</a>  | Sheffield<br>             | Prof. Dr. Hamish Cunningham<br>THE UNIVERSITY OF SHEFFIELD<br>Sheffield, UK<br>E-mail: h.cunningham@dcs.shef.ac.uk             |
| VRIJE UNIVERSITEIT AMSTERDAM<br><a href="http://www.vu.nl/">http://www.vu.nl/</a>   | Amsterdam<br>             | Prof. Dr. Frank van Harmelen<br>VRIJE UNIVERSITEIT AMSTERDAM<br>Amsterdam, Netherlands<br>E-mail: Frank.van.Harmelen@cs.vu.nl  |
| THE INTERNATIONAL WIC INSTITUTE, BEIJING UNIVERSITY OF TECHNOLOGY<br><a href="http://www.iwici.org/">http://www.iwici.org/</a>                        | WICI<br>                  | Prof. Dr. Ning Zhong<br>THE INTERNATIONAL WIC INSTITUTE<br>Mabeshi, Japan<br>E-mail: zhong@maebashi-it.ac.jp                   |



|  |  |   |
|--|--|---|
| <p>INTERNATIONAL AGENCY FOR RESEARCH<br/>ON CANCER<br/><a href="http://www.iarc.fr/">http://www.iarc.fr/</a></p> | <p>IARC2</p>  The logo of the International Agency for Research on Cancer (IARC) is a blue emblem. It features a central caduceus (a staff with two snakes entwined around it and wings at the top) superimposed on a globe. The globe is surrounded by a laurel wreath, and the entire emblem is set within a circular border. | <p>Dr. Paul Brennan<br/>INTERNATIONAL AGENCY FOR RESEARCH ON<br/>CANCER<br/>Lyon, France<br/>E-mail: <a href="mailto:brennan@iarc.fr">brennan@iarc.fr</a></p> |
|--|--|---|

# TABLE OF CONTENTS

|       |  |    |
|-------|--|----|
| 1     | INTRODUCTION                                     | 1  |
| 2     | STATISTICAL MACHINE LEARNING ON SEMANTIC WEB     | 4  |
| 2.1   | Feature-based Statistical Learning on the SW     | 4  |
| 2.1.1 | Feature-based Statistical Learning               | 4  |
| 2.1.2 | Feature-based Statistical Learning on the SW     | 5  |
| 2.1.3 | Search for the Best Features                     | 6  |
| 2.1.4 | Discussion                                       | 7  |
| 2.2   | Inductive Logic Programming                      | 9  |
| 2.2.1 | ILP Overview                                     | 9  |
| 2.2.2 | Propositionalization, Upgrading and Lifting      | 10 |
| 2.2.3 | Discussion                                       | 11 |
| 2.3   | Learning with Relational Matrices                | 11 |
| 2.4   | Relational Graphical Models                      | 11 |
| 2.4.1 | Possible World Models on the SW                  | 12 |
| 2.4.2 | Directed RGMs                                    | 12 |
| 2.4.3 | Undirected RGMs                                  | 14 |
| 2.4.4 | Latent Class RGMs                                | 15 |
| 2.4.5 | Discussion                                       | 16 |
| 2.5   | Unstructured Data and the SW                     | 17 |
| 2.5.1 | Learning Ontologies from Text                    | 17 |
| 2.5.2 | Semantic Annotation                              | 19 |
| 2.6   | First Experiments in the Analysis of FOAF-data   | 21 |
| 2.7   | Conclusions                                      | 22 |
| 2.8   | Conclusions in the Context of LarKC              | 23 |
| 2.8.1 | Feature-based Statistical Learning on the SW     | 23 |
| 2.8.2 | Missing Data                                     | 24 |
| 2.8.3 | Search for Best Features                         | 24 |
| 2.8.4 | Integration of Ontological Background Knowledge  | 24 |
| 2.8.5 | RGM: IHRM  | 25 |
| 3     | DATA STREAMING TECHNOLOGIES FOR DATA ABSTRACTION | 26 |
| 3.1   | Introduction                                     | 26 |
| 3.2   | Well-Designed Queries for SPARQL                 | 28 |
| 3.3   | Aggregation in (C-)SPARQL                        | 28 |
| 3.4   | Stream Management in C-SPARQL                    | 30 |
| 3.5   | Urban Computing Scenario                         | 35 |
| 3.5.1 | Aggregation Queries                              | 36 |
| 3.5.2 | Stream Selection Query                           | 36 |
| 3.5.3 | Stream Composition Queries                       | 38 |
| 3.5.4 | Classification queries                           | 39 |
| 3.6   | Optimization Scenarios for C-SPARQL              | 40 |
| 3.7   | Previous Related Work                            | 41 |
| 3.7.1 | Data Streams                                     | 41 |
| 3.7.2 | SPARQL   | 42 |
| 3.8   | Conclusions                                      | 43 |



# 1 INTRODUCTION

The world wide web (WWW) represents an ever increasing source of information. Until now the WWW is mostly accessible to humans via search engines and browsers whereas computers only have a very rudimentary understanding of web content. The vision behind the Semantic Web (SW) is that computers should also be able to understand and exploit information offered on the web [10]. In the near future, a web representation might contain human-readable parts and sections made available in SW-formats to be accessible for automated processing. The SW is based on two concepts. First, a formal ontology provides domain specific background information that is shared by several parties: It provides a common vocabulary for a given domain and describes object classes, predicate classes and their interdependencies, as well as additional background information formalized in logical statements. Second, web information is annotated by statements readable and interpretable by machines via the common ontological background knowledge.

One of the prime SW applications will be context/user sensitive information retrieval where the result will still be in textual or multimedia format, to be interpreted by a human. But this information will be much more specific to the user's needs, since data can be integrated from multiple sites and smart information filters can be applied. Thus a search engine becomes more of an agent who knows the user, who has a deep understanding of the information request, who knows what to find where on the web and who presents the requested information in an appropriate user-friendly form. An immediate benefit might be that semantically annotated web pages obtain a higher search rank since the match between query and page content can be evaluated with high confidence. In a second group of applications, the items to be searched for are not human readable texts or multimedia data but are machine readable information about an item or a web service. Semantic web services are of great interest both for academia and industry [130, 50]. Service requests and service offerings can be formulated precisely based on SW standards and can be understood as precisely by semantic search engines and web applications. In the third family of applications the SW becomes the *web of data*. SW technologies will form the infrastructure for a standardized representation of information and for information exchange. Biomedicine is a forerunner here with almost 1000 databases publicly available today. If the data were published under a common SW ontological format, all this information would be accessible for querying and for analysis. As the WWW brought the knowledge of the world to our finger tips, the SW will bring the data of the world to our applications. Finally, in a fourth family of applications, SW technologies are being used in advanced expert systems to model complex industrial domains [110]. Here, SW technologies indeed realize answering machines since they not only point to information sources but also directly provide the diagnosis, solutions and answers.

Reasoning plays an important role on the SW: Based on ontological background knowledge and the set of asserted statements, logical reasoning can derive new statements. But logical reasoning has its limitations. First, logical reasoning does not easily scale up to the size of the web as required by many applications; projects like the EU FP 7 Large-Scale Integrating Project LarKC are under way to address this issue [84]. Second, uncertain information is not suitable for logical reasoning. The representation of uncertain information on the SW and reasoning

with uncertainty on the SW have only recently been addressed [71]. Third, logical reasoning is completely based on axiomatic prior knowledge and does not exploit regularities in the data that have not been formulated as ontological background knowledge. In contrast, and as it has been demonstrated in many application areas, successful solutions can often be achieved by induction, i.e., by learning from data. The analysis of the potential of machine learning for the SW is the topic of this contribution.

The most immediate application of machine learning is SW mining, enhancing traditional web mining applications. Web content mining, web structure mining, web usage mining and the learning of ranking functions for retrieval will all benefit from the additional information available on the SW [9]. In another group of applications, machine learning serves the SW by supporting ontology construction and management, ontology evaluation, ontology refinement, ontology evolution, as well as the mapping, merging and alignment of ontologies [62, 63, 17, 22, 51]. Mostly these tasks are addressed on the basis of unstructured or semi-structured textual data. After all, most current web pages contain textual information; but other types of input data will become increasingly important, as well [102]. More generally we are concerned with learning of data in SW formats. As already mentioned, the current trend is that an increasing amount of information is made available in SW formats and machine learning and data mining will be the basis for the analysis of the combined data sources. A particular aspect here is the learning of logical constraints that can then be formulated in the language of the employed ontology [90, 92, 93, 91]. One can also contemplate that future ontologies should be extended to be able to represent learned information that cannot easily be formulated with current standards, e.g., represent the input-output mapping represented in probabilistic classifiers. The trained statistical models can furthermore be used to estimate the probability that statements are true that are neither explicitly asserted in the database nor can be proven to be true (or false) based on logical reasoning. Since the conclusions drawn from machine learning are typically probabilistic, this uncertainty needs to be represented. The weighted RDF-graphs, as explored in [52, 84], might lead to a suitable representation. Consequently, querying needs to support uncertain statements so that a query such as: *Find all female persons that live in the southeastern US, are older than 21 years, own a house and are highly ranked as being interested in buying a sailboat* could be implemented where the last information, i.e., the interest in buying a sailboat, was learned from data. Finally, in applications where the raw data is unstructured, machine learning can support the population of ontologies, i.e., the mapping of unstructured data to SW statements. Most work here concerns the population from textual data although the annotation of semi-structured data and multimedia data. e.g. images and video, is of great relevance as well. The goal here is to describe multimedia content semantically for fast content-based reasoning and retrieval.

In this paper we analyze algorithms from machine learning that are suitable for SW applications. First and foremost, SW data describe relationships between objects. Consequently, suitable learning approaches should be able to handle the relational character of the data. By far the majority of machine learning deals with non-relational feature-based representations (also referred to as propositional representation or attribute-value representation). Only recently statistical rela-

tional learning (SRL) is finding increasing interest in the ML community [55]. In Section 2.1 we present a novel discussion on feature-based learning in the SW and in Section 2.2 we relate this discussion to learning algorithms from inductive logic programming (ILP). In Section 2.3 we discuss matrix decomposition approaches and in Section 2.4 we present relational graphical models that are based on a joint probabilistic model of a relational domain. We discuss the machine learning approaches with respect to their applicability in a SW context, i.e., their scalability to the expected large size of the SW, their ability to integrate ontological background knowledge, their ability to handle the varying quality and reliability of data<sup>1</sup> and finally, their ability to deal with missing and contradictory data. In Section 2.5 we add a discussion on ontology learning and ontology population based on textual data. Ontology learning and ontology population are the most developed aspects of machine learning on the SW. In Section 2.6 we report first experiments based on the FOAF data set and in Section 2.7 we present conclusions. We will start the remaining part of the paper with an introduction into the SW as proposed by the W3C.

---

<sup>1</sup>Trust learning is an emerging field [119].

## 2 STATISTICAL MACHINE LEARNING ON SEMANTIC WEB

### 2.1 Feature-based Statistical Learning on the SW

#### 2.1.1 Feature-based Statistical Learning

Based on a long tradition, statistical learning has developed a large number of powerful analytical tools and it is highly desirable to make these tools available for the SW. Figure 2.1 (top) shows the main steps that are performed in statistical learning, analyzing, as example, students in a university. First, a *statistical unit* is defined, which is the entity that is the source of the variables or features of interest [49, 27, 129]. The goal is to generalize from observations on a few units to a statistical assembly of units. Typically a statistical unit is an object of a given type, here a student. In general one is not interested in all statistical units but only in a particular subset, i.e., the *population*. The population might be defined in various ways, for example it might concern all students in a particular country or, alternatively, all female students at a particular university.

In a statistical analysis only a subset of the population is available for investigation, i.e. a *sample*. Statistical inference is dependent on the details of the sampling process; the sampling process essentially defines the random experiment and, as a stationary process, allows the generalization from the sample to the population. In a *simple random sample* each unit is selected independently. Naturally, sometimes more complex sampling schemes are used, such as stratified random sampling, cluster sampling, and systematic sampling.

The quantities of interest of the statistical investigation are the *features* (or variables) that are derived from the statistical units. In the example, features are a student's IQ and a student's age. In the next step the *data matrix* is formed where each row corresponds to a statistical unit and each column corresponds to a feature. Finally, an appropriate statistical model is employed for modeling the data, i.e., the analysis of the features and the relationships between the features, and the final result is analyzed by the user. Naturally, all of this is typically an iterative process, e.g., based on a first analysis new features might be added and the statistical model might be modified.

In a supervised statistical analysis one partitions the features in explanatory variables (a.k.a. independent variables, predictor variables, regressors, controlled variables, input variables) and dependent variables (a.k.a response variables, the regressands, the responding variables, the explained variables, or the outcome/output variables). Note that it is often a design choice if one either defines a population based on the state of a variable or if one uses that variable as an independent variable. Consider a binary variable male/female. One choice might be to partition the population into males and females and learn separate models for each population. Another option is to simply use gender as an independent variable and consider a joint population of males and females. The second choice is for example more appropriate if the sample is small. Hierarchical Bayesian modeling is a compromise in which statistical inference in different populations is coupled.

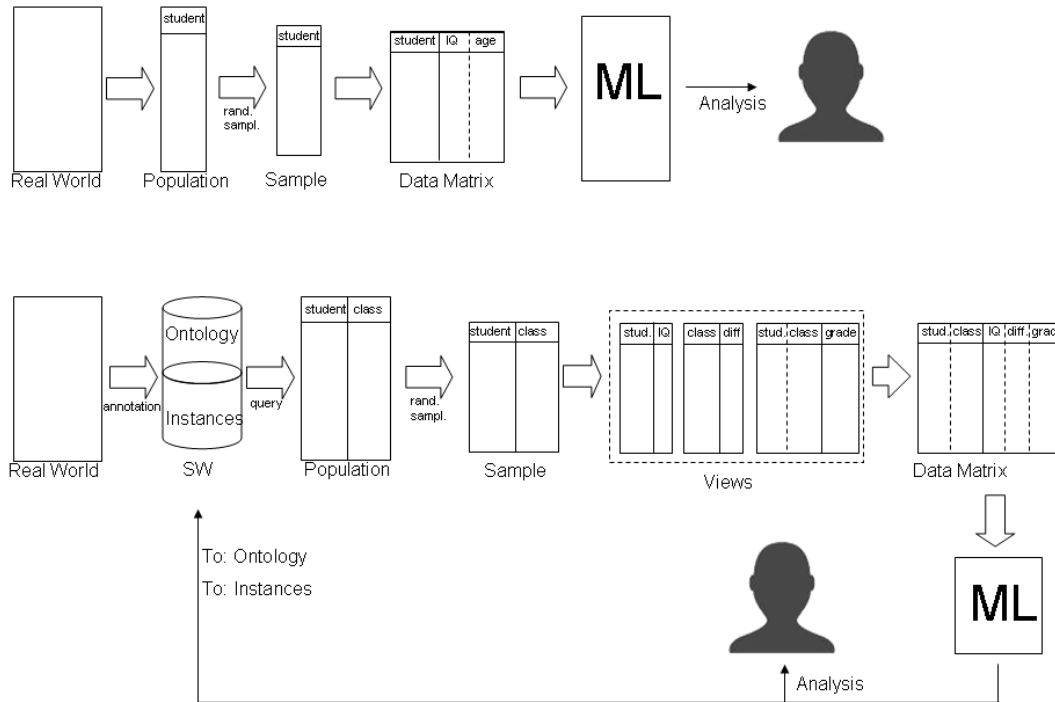


Figure 2.1: Top: Standard machine learning. Bottom: Machine learning applied to the SW.

### 2.1.2 Feature-based Statistical Learning on the SW

The main steps for statistical learning on the SW are displayed in Figure 2.1 (bottom). The first new aspect is that the statistical analysis is based on the world as it is represented on the SW and that all quantities of interest, i.e., statistical unit, population, sample and features, are defined in context of the SW.<sup>1</sup> As before, a *statistical unit* might be defined to be an object of a given type, e.g., a student. More generally a statistical unit might be composed of several objects that have a particular relationship to each other. In Figure 2.1, as an example, a statistical unit might be a composed entity consisting of a student and a class that the student attends, i.e., a registration.

A *population* might now be defined by a SW query that produces a table whose tuples (i.e., variable bindings) correspond to the objects that identify a statistical unit. In the example in Figure 2.1 we might define a query to generate a population table with objects student and class; a tuple then stands for the statement that a student registered in a particular class. Sampling, as before, selects a proper random subset of the population. A particular aspect of SW data is the dominance of relationships between objects. Thus, features that are calculated for a statistical unit might reflect this relationship structure.

Technically, one first generates a data matrix. The number of rows in the data matrix is identical to the number of tuples in the sample table, i.e., the number of statistical units in the sample. A statistical unit is a primary key for the table.

<sup>1</sup>Technically one needs to be aware that the generation of a sample with the help of a search engine or a crawler might introduce a bias, for example, if snowball sampling is employed.

The data matrix has a fixed number of columns corresponding to the number of features, which are derived for each unit. All matrix entries are initialized to be N/A (not available or missing) and will (partially) be replaced by feature values as described in the following two steps.

Next, database views<sup>2</sup> are generated that contain as attributes the objects in a statistical unit (respectively a subset of those objects) plus additional attributes. In Figure 2.1, the first view contains the student’s ID and the student’s IQ, the second view contains the class ID and the class difficulty and the third view contains the student ID, the class ID and the grade the student obtained in a class. Note that views can be generated from rather complex queries.

In the next step, relational features are calculated based on these views. In the simplest case each statistical unit is represented exactly in one tuple in each view and features are calculated based on the tuple attributes. The situation becomes more complex if a statistical unit is not represented in a view or if it is represented more than once. In the first case, i.e., a statistical unit is not represented in a view, one either enters zero or another default entry (e.g., the number of a person’s children is zero) or one does not overwrite the corresponding N/A entry in the data matrix (e.g., when a student’s IQ is unknown). In the second case, i.e., a statistical unit is represented in more than one tuple in a particular view—in the example if a student attended a class twice and got two grades—some form of aggregation can be applied (number-of, average, max, min, etc.). In domains like the SW, many-to-many relations often play a significant role and can lead to a large number of sparse features: The number of items a customer has acquired is typically still very small if compared to the total number of items. In the case that such features are used, the learning algorithm needs to be able to handle the high-dimensional sparse data. Technically, it might be possible to execute the described steps, i.e., the generation of the sample, the views and the data matrix, in one SQL/SPARQL operation.

Finally, the statistical model can be applied beyond the sample to the population. It is important to note that we have a well-defined statistical problem as long as we restrict the analysis to the world in as much it is represented in the SW. Of course the SW can grow (and shrink) such that online learning and transfer learning might become applicable. To what degree the statistical model can be generalized to the real world needs to be analyzed carefully since sometimes the SW data are generated by multiple parties for their own reasons and not for the purpose of a statistical analysis.

### 2.1.3 Search for the Best Features

So far it was assumed that the user would be able to define the features of interest. In particular in supervised learning one is often interested to automate the selection of the best input features. Popescul and Ungar [113] describe a relational learning approach based on a greedy search for optimal relational features derived from SQL queries (see also [74]). Features are dynamically generated by a refinement style search over SQL queries including aggregation, statistical operators, groupings, richer join conditions and argmax based queries. The features are used to predict

---

<sup>2</sup> A view is a stored query accessible as virtual table composed of the result set of a query. Alternatively, one could also work with a temporary or persistent table.

the target relation using logistic regression. Additional features are generated by clustering, which leads to new “invented” attributes. The authors obtain good results on citation prediction and document classification. It is straightforward to implement a similar search procedure on SW data. Note that the automatic generation of candidate features is certainly attractive; on the other hand the computational burden is quite large; feature definition based on the experimenters insight and some input pruning might be adequate in many applications.

#### 2.1.4 Discussion

Statistical learning on the SW, as presented, is highly scalable since the determining factor is the number of statistical units in the sample, which basically is independent of the size of the SW. One needs to be aware that sampling with the help of a search engine or a crawler might introduce a bias. The queries, which need to be executed for the calculation of the features, can be executed efficiently with current technology [77]. Ontological background knowledge can be integrated in different ways. The most obvious one would be to perform complete or partial materialization, which would derive statements from reasoning prior to training. Recall that total materialization is only feasible with less expressive ontologies. Second, since the ontology is part of the RDF-graph, features can be defined including ontological concepts of a statistical units, respecting the subclass restrictions. This has effectively been employed in [118]. It is conceivable that the trained statistical models could be added to an extended “probabilistic” ontology, indicated by the arrow at the bottom of Figure 2.1. In addition, the statistical models derive probabilistic statements about the truth values of triples. For example, if —based on a trained model— it can be derived that a person has a high IQ, this information could be added to the SW as a weighted RDF-triple, the weight reflecting the likelihood that this statement is true. Moreover, if it was found that particular features generated during learning are valuable, one could define corresponding statements and add those to the SW as “invented predicates”. The same is true for the latent variables introduced in a cluster analysis or in a principle component analysis (PCA). We should emphasize again that statistical inference strictly speaking is only applicable within the experimental setting of a particular statistical unit, population and sampling approach. Thus if a statistical model allows the conclusion that *statement X is true with 90% probability*, this is only valid in a particular statistical context. Experiments have shown, for example, that predictive performance can depend to some degree on the object selected as statistical unit. An interesting view is that the results from a number of statistical models could be combined in a committee machine [128].

Feature generation is nontrivial and might exploit prior knowledge that is partially available in the domain ontology. For example it is relevant that a person only has exactly one age, exactly one mother, but zero or more children. In fact it would be desirable that the ontological information could be exploited in a way such that the statistical framework is automatically constructed requiring a minimum of additional domain background knowledge from the user. A problem with less expressive ontologies might be that one cannot express negation. Consider the example of gene-gene interactions where the literature primarily reports positive results, i.e., positive evidence for gene-gene interactions. Evidence that two genes

do not interact would be important to report but might be difficult to represent in less expressive ontologies.

Maybe the most important issue concerns missing or incomplete data. We can make a closed-world assumption and postulate that the world only exists in as much as it is represented in the SW: besides the statements that are known to be true or can be derived to be true, all statements are assumed false. Naturally, in many cases we are really interested to perform inference in the real world and it is more appropriate to assume that the truth values of some statements are unknown. Here we should distinguish, first, the case that statistical units are missing and, second, the case that due to missing information features cannot be calculated or features are biased. The first case is not a problem if statistical units are missing at random, e.g., if some of the students at a university are unknown. The situation is more complex if the fact that a statistical unit is missing is dependent on features of interest, e.g., if only smart students are in the data base. Then the missing data mechanism should be included in the statistical model. For the second case consider that the age of a person's father is an important feature that is not available: Either the age of a person's father might be unknown or a person father's ID might be unknown. Another example is that if the number of transactions is an important feature, the feature might be biased if not all transactions are recorded. If a closed-world assumption is not appropriate, one could deal with missing features using the appropriate procedures known from statistics [94]. Again, the missing data mechanism should be included in the statistical model. Also note that ontological information can be quite relevant for dealing with missing data. For example if we know that a person has brown eye color we know that all other statements about eye color must be false, since a person has only one eye color. Note that there are statistical models that can easily deal with missing data if they occur at the input of a supervised model such as naive Bayes, many nearest neighbor methods, or kernel smoothers.

Naturally there are cases where simple missing data models are not appropriate, since missing data can render the independent sample assumption invalid. Consider objects of type *Person* and the properties *friendOf* and *income* and *age*. Furthermore assume that from the age of a person and from the income of a person's friends we can predict the income of a person with some certainty. If all features are available, then training an appropriate classifier is straightforward. If in training and testing the income of a person and of a person's friends are partially unknown, we have the situation that the income prediction for one person depends on the income prediction of the person's friends. The situation, where for the prediction of features of a statistical unit (here the income) the same features of linked statistical units are required, is typical for data defined on networks. In the analysis of social networks, this situation is referred to as a collective classification problem and a mechanism is added to propagate information using, e.g., Gibbs sampling, relaxation labeling, iterative classification or loopy belief propagation. Recent overviews are presented in [123, 97]. One of the first papers demonstrating the benefits of collective classification is [29] and some important contributions are described in [106, 96, 56, 127]. It is likely that collective classification will also concern SW applications. Interestingly, many social networks have been shown to exhibit homophily, which means that objects with similar attributes (e.g., persons with similar income) are linked (e.g., are friends).

In networks exhibiting homophily, simple propagation models, for example based on Gaussian random field models employed in semi-supervised learning [140], give very competitive results. Collective classification is highly related to the relational graphical model approaches described in Section 2.4, in particular dependency networks [108, 107]. Note, that in collective classification, the sample is not independent and a statistical analysis becomes more involved. In contrast, statistical units should not be selected independently, rather the statistical units (for both training and test) would be defined by the complete RDF-graph or a connected RDF-subgraph (compare, Section 2.4).

## 2.2 Inductive Logic Programming

Inductive logic programming (ILP) encompasses a number of approaches that attempt to learn logical clauses. In the view of the discussion in the last section, ILP uses logical (binary) features derived from logical expressions, typically conjunctions of (negated) atoms. Recent extensions on probabilistic ILP have also addressed uncertain domains.

### 2.2.1 ILP Overview

This section is on “strong” ILP, which covers the majority of ILP approaches and is concerned with the classification of statistical units and on predicate definition<sup>3</sup>. Strong ILP performs modeling in relational domains that is somewhat related to the approach discussed in the previous section. Let’s consider FOIL (First Order Inductive Learner) as a typical representative [115]. The outcome of FOIL is a set of definite clauses (a particular if-then-rule) with the same head (then-part).

Here is an example (modified from [45]). Let the statistical unit be a customer with ID  $CID$ .  $VC = 1$ , indicates that someone is a valuable customer,  $GC = 1$  indicates that someone owns a golden credit card and  $SB = 1$  indicates that someone would buy a sailboat. The first rule that FOIL might have learned is that a person is interested in buying a sailboat if this person owns a gold card. The second rule indicates that a person would buy a sailboat if this person is older than 30 and has at least once made a credit card purchase of more than 100 EURO:

$$\begin{aligned}
 sailBoat(CID, SB = 1) &\leftarrow customer(CID, GC = 1) && (2.1) \\
 sailBoat(CID, SB = 1) &\leftarrow customer(CID, Age) \\
 &\quad \wedge purchase(CID, PID, Value, PM) \\
 &\quad \wedge PM = credit-card \wedge Value > 100 \wedge Age > 30.
 \end{aligned}$$

In rule learning FOIL uses a covering paradigm. Thus the first rule is derived to correctly predict as many positive examples as possible (covering) with a minimum number of false positives. Subsequent rules then try to cover the remaining positive examples. The head of a rule (then-part) is a predicate and the body (the if-part) is a product of (negated) atoms containing constants and variables.<sup>4</sup> Naturally, there

<sup>3</sup>A predicate definition is a set of program clauses with the same predicate symbol in their heads.

<sup>4</sup>FOIL learning is called learning from entailment in ILP terminology.

are many variants of FOIL. FOIL uses a top down search strategy for refining the rule bodies, PROGOL [103] a bottom up strategy and GOLEM [104] a combined strategy. Furthermore, FOIL uses a conjunction of atoms and negated atoms in the body, whereas other approaches use PROLOG constructs. The community typically discusses the different approaches in terms of language bias (which rules can the language express), search bias (which rules can be found) and validation bias (when does validation tell me to stop refining a rule). An advantage of ILP is that also non-grounded background knowledge can be taken into account (typically in form of a set of definite clauses that might be part of an ontology).

In view of the discussion in the last section, the statistical unit corresponds to a customer, and FOIL introduces a binary target feature (1) for the target predicate *sailBoat(CID, SB)*. The second feature (1) is one if the customer owns a golden credit card and zero otherwise. Then a view is generated with attribute CID. A CID is entered in that view each time the person has made a credit card purchase of more than 100 EURO, but only if that person is older than 30 years. The third feature (3) is binary and is equal to one if the CID is present in the view at least once and zero otherwise. FOIL then applies a very simple combination rule: if feature (2) or feature (3) is equal to one for a customer, then the target feature (1) is true.

## 2.2.2 Propositionalization, Upgrading and Lifting

ILP approaches like FOIL can be decomposed into the generation of binary features (based on the rule bodies) and a logical combination, which in case of FOIL is quite simple. As stated before, ILP approaches contain a complex search strategy for defining optimal rule bodies. If, in contrast, the generation of the rule bodies is performed as a preprocessing step, the process is referred to as propositionalization [79]. Instead of using the simple FOIL combination rule, other feature-based learners are often used. It has been proven that in some special cases, propositionalization is inefficient [40]. Still, propositionalization has produced excellent results. The binary features are often collected through simple joins of all possible attributes. An early approach to propositionalization is LINUS [85].

*Upgrading* (or *lifting*) of a feature-based learner [131] is an alternative to propositionalization. The main differences to propositionalization is that the optimization of the features is guided by the improvement of the performance of the overall system. It turns out that many strong ILP systems can be interpreted as upgraded propositional learners: FOIL is an upgrade of the propositional rule-induction program CN2 and PROGOL can be viewed as upgrading the AQ approach to rule induction. Additional upgraded systems are Inductive Classification Logic (ICL [42]) that uses classification rules, TILDE [16] and S-CART that use classification trees, and RIBL [47] that uses nearest neighbor classifiers. nFOIL [83] combines FOIL with a naive Bayes (NB) classifier by changing the scoring function and by introducing probabilistic covering. nFoil was able to outperform FOIL and propositionalized NB on standard ILP problems. kFoil [82] is another variant that derives kernels from FOIL-based features.

### 2.2.3 Discussion

ILP algorithms can easily be applied to the SW if we identify atoms with basic statements. ILP fits well into the basically deterministic framework of the SW. In many ways, statistical SW learning as presented in Section 2.1 is related to ILP's propositionalization; the main difference is the principled statistical framework of the former. Thus most of the discussion on scalability in Section 2.1 carries over to ILP's propositionalization. When ILP's complex search strategy for defining optimal rule bodies is applied, training time increases but is still proportional to the number of samples. An interesting new aspect is that ILP produces definite clauses that can be integrated, maybe with some restrictions, into the Semantic Web Rule Language. ILP approaches that consider learning with description logic (and clauses) are described, for example, in [38, 121, 90, 92, 93, 91]. An empirical study can be found in [46].

## 2.3 Learning with Relational Matrices

Another representation of a basic statement (RDF-triple) is a matrix entry. Consider the triple  $(User, buys, Item)$ . Recall that a standard relational representation would be the table *buys* with attributes *User* and *Item*. A relational adjacency matrix on the other hand has as many rows as there are users and as many columns as there are items and as many matrix entries as there are possibly true statements. A matrix entry is equal to one if the item was actually bought by a user and is equal to zero otherwise. Thus SW data can be represented as a set of matrices where the name of the matrix is the property of the relation under consideration. Matrix decomposition/reconstruction methods, e.g., the principle component analysis (PCA) and other more scalable approaches have been very successful in the prediction of unknown matrix entries [126]. Lippert *et al.* [89] have shown how several matrices can be decomposed/reconstructed jointly and have shown that this increases predictive performance if compared to single matrix decompositions. By filling in the unknown entries via matrix decomposition/reconstruction, the approach has an inherent way of dealing with data that is missing at random. Care must be taken if missing at random is not justified. In [89], one type of statement concerns gene-gene interactions where only positive statements are known. Reconstructed matrix entries can, as before, be entered into the SW, e.g., as weighted triples. Scalability of this approach has not been studied in depth but the decomposition scales approximately proportional to the number of known matrix entries. Note that the approach performs a prediction for all unknown statements in one global decomposition/reconstruction step. In contrast, the previous approaches would learn separate models for each statistical unit under consideration. Other approaches, which learn with the relational adjacency matrix, are described in [137] and [138].

## 2.4 Relational Graphical Models

The approaches described in Sections 2.1 and 2.2 aim at describing the statistical, respectively logical, dependencies between features derived from SW data. In contrast the matrix decomposition approach in the last section and the relational

graphical models (RGMs) in this section predict the truth values of all basis statements (RDF-triples) in the SW. Unlike the matrix decomposition techniques in the last section, the RGMs are probabilistic models and statements are represented by random variables. RGMs can be thought of as upgraded versions of regular graphical models, e.g., Bayesian networks, Markov networks, dependency networks and latent variable models. RGMs have been developed in the context of frame-based logical representations, relational data models, plate models, entity-relationship models and first-order logic. Here, we attempt to relate the basic ideas of the different approaches to the SW framework.

### 2.4.1 Possible World Models on the SW

Consider all constants in the SW (i.e., all objects and literal values) and all statements that can possibly be true. Now one introduces a binary random variable  $U$  for each possibly true statement (grounded atom), where  $U = 1$  if the corresponding statement is true and  $U = 0$  otherwise. In a graphical model,  $U$  would be identified with a node. These nodes should not be confused with the nodes in the RFD-graph, which represent URIs; rather  $U$  stands for a potential link in the RDF-graph. We can reduce the number of random variables if type constraints are available and if the truth value of some statements are assumed known in each world under consideration (e.g., if object-to-object statements are all assumed known, as in the basic PRM model in Subsection 2.4.2). If statements are mutually exclusive, e.g., the different blood types of a person, one might integrate several statements into one random variable using, e.g., multi-state multinomial variables or continuous variables (to encode, e.g., a person's height). An assignment of truth values to all random variables defines a possible world or Herbrand interpretation<sup>5</sup>. RGMs assign a probability distribution to each world in the form  $P(\vec{U} = \vec{u})$ .<sup>6</sup> The approaches differ in how these probabilities are defined and mapped to random variables, and how they are learned.

### 2.4.2 Directed RGMs

The probability distribution in a directed RGM, i.e., relational Bayesian model, can be written as

$$P(\vec{U} = \vec{u}) = \prod_{U \in \vec{U}} P(U | \text{par}(U)).$$

$U$  is represented as a node in a Bayesian network and arcs are pointing from all parent nodes  $\text{par}(U)$  to the node  $U$ . One now partitions all elements of  $\vec{U}$  into node-classes. Each  $U$  belongs to exactly one node-class. The key property of all  $U$  in the same node-class is that their local distributions are identical, which means that  $P(U | \text{par}(U))$  is the same for all nodes within a node-class and can be described by a truth-table or more complex representations such as decision trees. For example, all nodes representing the IQ-values of students in a university might form a node class, all nodes representing the difficulties of university courses might

<sup>5</sup>RGM modeling would be termed learning from interpretation in ILP terminology. The set of all possible worlds is also called Herbrand base.

<sup>6</sup>Our discussion includes the case that we are only interested in a conditional distribution of the form  $P(\vec{U} = \vec{u} | \vec{V} = \vec{v})$ , as in conditional random fields [81]

form a node class, and the nodes representing the grades of students in courses might form a node-class. Care must be taken, that no directed loops are introduced in the Bayesian network in modeling or structural learning.

### Probabilistic Relational Models (PRMs):

PRMs were one of the first published RGMs and found great interest in the statistical machine learning community [78, 55]. PRMs combine a frame-based logical representation with probabilistic semantics based on directed graphical models. The nodes in a PRM model the probability distribution of object attributes whereas the relationships between objects are assumed known. Naturally, this assumption simplifies the model greatly. In context of the SW object attributes would primarily correspond to object-to-literal statements. In subsequent papers PRMs have been extended to also consider the case that relationships between objects (in context of the SW these would roughly be the object-to-object statements) are unknown, which is called *structural uncertainty* in the PRM framework [55]. The simpler case, where one of the objects in a statement is known, but the partner object is unknown, is referred to as *reference uncertainty*. In reference uncertainty the number of potentially true statements is assumed known, which means that only as many random nodes need to be introduced. The second form of structural uncertainty is referred to as *existence uncertainty*, where binary random variables are introduced representing the truth values of relationships between objects.

For some PRMs, regularities in the PRM structure can be exploited (encapsulation) and exact inference is possible. Large PRMs require approximate inference; commonly, loopy belief propagation is being used. Learning in PRMs is likelihood based or based on empirical Bayesian learning. Structural learning typically uses a greedy search strategy, where one needs to guarantee that the ground Bayesian network does not contain directed loops.

### More Directed RGMs:

A *Bayesian logic program* is defined as a set of Bayesian clauses [76]. A Bayesian clause specifies the conditional probability distribution of a random variable given its parents on a template level, i.e. in a node-class. A special feature is that, for a given random variable, *several* such conditional probability distributions might be given. As an example,  $bt(X) \mid mc(X)$  and  $bt(X) \mid pc(X)$  specify the probability distribution for blood type given the two different dispositions  $mc(X)$  and  $pc(X)$ . The truth value for  $bt(X) \mid mc(X)$ ,  $pc(X)$  can then be calculated based on various combination rules (e.g., noisy-or). In a Bayesian logic program, for each clause there is one conditional probability distribution and for each Bayesian predicate (i.e., node-class) there is one combination rule. *Relational Bayesian networks* [72] are related to Bayesian logic programs and use probability formulae for specifying conditional probabilities. *Relational dependency networks* [108] also belong to the family of directed RGMs and learn the dependency of a node given its Markov blanket using decision trees.

### 2.4.3 Undirected RGMs

The probability distribution of an undirected graphical model or Markov network can be written as

$$P(\vec{U} = \vec{u}) = \frac{1}{Z} \prod_k g_k(u_k)$$

where  $g_k(\cdot)$  is a potential function,  $u_k$  is the state of the  $k$ -th clique and  $Z$  is the partition function normalizing the distribution. One often prefers a more convenient log-linear representation of the form

$$P(\vec{U} = \vec{u}) = \frac{1}{Z} \exp \sum_k w_k f_k(u_k)$$

where the feature functions  $f_k$  can be any real-valued function and where  $w_i \in \mathbb{R}$ .

We will discuss two major approaches that use this representation: Markov logic networks and relational Markov models.

#### Markov Logic Networks (MLN):

Let  $F_i$  be a formula of first-order and let  $w_i \in \mathbb{R}$  be a weight attached to each formula. Then a MLN  $L$  is defined as a set of pairs  $(F_i, w_i)$  [120] [44]. One introduces a binary node for each possible grounding of each predicate appearing in  $L$  (i.e., in context of the SW we would introduce a node for each possible statement), given a set of constants  $c_1, \dots, c_{|C|}$ . The state of the node is equal to 1 if the ground atom/statement is true, and 0 otherwise (for an  $N$ -ary predicate there are  $|C|^N$  such nodes). A grounding of a formula is an assignment of constants to the variables in the formula (considering formulas that are universally quantified). If a formula contains  $N$  variables, then there are  $|C|^N$  such assignments. The nodes in the Markov network  $M_{L,C}$  are the grounded predicates. In addition the MLN contains one feature for each possible grounding of each formula  $F_i$  in  $L$ . The value of this feature is 1 if the ground formula is true, and 0 otherwise.  $w_i$  is the weight associated with  $F_i$  in  $L$ . A Markov network  $M_{L,C}$  is a grounded Markov logic network of  $L$  with

$$P(\vec{U} = \vec{u}) = \frac{1}{Z} \exp \left( \sum_i w_i n_i(\vec{u}) \right)$$

where  $n_i(\vec{u})$  is the number of formula groundings that are true for  $F_i$ . MLN makes the unique names assumption, the domain closure assumption and the known function assumption, but all these assumptions can be relaxed.

A MLN puts weights on formulas: the larger the weight, the higher is the confidence that a formula is true. When all weights are equal and become infinite, one strictly enforces the formulas and all worlds that agree with the formulas have the same probability.

The simplest form of inference concerns the prediction of the truth value of a grounded predicate given the truth values of other grounded predicates (conjunction of predicates) for which the authors present an efficient algorithm. In the first phase, the minimal subset of the ground Markov network is returned that is required to calculate the conditional probability. It is essential that this subset

is small since in the worst case, inference could involve alle nodes. In the second phase Gibbs sampling in this reduced network is used.

Learning consists of estimating the  $w_i$ . In learning, MLN makes a closed-world assumption and employs a pseudo-likelihood cost function, which is the product of the probabilities of each node given its Markov blanket. Optimization is performed using a limited memory BFGS algorithm.

Finally, there is the issue of structural learning, which, in this context, defines the employed first order formulae. Some formulae are typically defined by a domain expert *a priori*. Additional formulae can be learned by directly optimizing the pseudo-likelihood cost function or by using ILP algorithms. For the latter, the authors use CLAUDIAN [41], which can learn arbitrary first-order clauses (not just Horn clauses, as many other ILP approaches).

### Relational Markov Networks (RMNs):

RMNs generalize many concepts of PRMs to undirected RGMs [127]. RMNs use conjunctive database queries as clique templates. By default, RMNs define a feature function for each possible state of a clique, making them exponential in clique size. RMNs are mostly trained discriminately. In contrast to MLN, RMNs, as PRMs, do not make a closed-world assumption during learning.

#### 2.4.4 Latent Class RGMs

The infinite hidden relational model (IHRM) [134] presented here is a directed RGM (i.e., a relational Bayesian model) with latent variables.<sup>7</sup> The IHRM is formed as follows. First, we partition all objects into classes  $K_1, \dots, K_{|K|}$ , using, for example, ontological class information. For each object in each class, we introduce a statement  $(Object, hasHiddenState, H)$ . If  $Object$  belongs to class  $K_i$ , then  $H \in \{1, \dots, N_{K_i}\}$ , i.e., the number of states of  $H$  is class dependent. As before, we introduce a random variable or node  $U$  for each grounded atom, respectively potentially true basic statement. Let  $Z_{Object}$  denote the random variables that involve  $Object$  and  $H$ .  $Z_{Object}$  is a latent variable or latent node since the true state of  $H$  is unknown.  $Z_{Object} = j$  stand for the statement that  $(Object, hasHiddenState, j)$ .

We now define a Bayesian network where the nodes  $Z_{Object}$  have no parents and the parents of the nodes for all other statement are the latent variables of the objects appearing in the statement. In other words, if  $U$  stands for the fact that  $(Object_1, property, Object_2)$  is true, then there are arcs from  $Z_{Object_1}$  and  $Z_{Object_2}$  to  $U$ . The object-classes of the objects in a statement together with the property define a node-class for  $U$ . If the property value is a literal, then the only parent of  $U$  is  $Z_{Object_1}$ .

In the IHRM we let the number of states in each latent node to be infinite and use the formalism of Dirichlet process mixture models. In inference, only a small number of the infinite states are occupied, leading to a clustering solution where the number of states in the latent variables  $N_{C_i}$  is automatically determined during inference.

Since the dependency structure in the ground Bayesian network is local, one might get the impression that only local information influences prediction. This is

<sup>7</sup>Kemp et al. [75] presented an almost identical model independently.

not true, since in the ground Bayesian network, common children  $U$  with evidence lead to interactions between the parent latent variables. Thus information can propagate in the network of latent variables. Training is based on various forms of Gibbs sampling (e.g., the Chinese restaurant process) or mean field approximations. Training only needs to consider random variables  $U$  corresponding to statements that received evidence, e.g., statements that are either known to be true or known not to be true; random variables that correspond to statements with an unknown truth value (i.e., without evidence) can completely be ignored.

The IHRM has a number of key advantages. First, no structural learning is required, since the directed arcs in the ground Bayesian network are directly given by the structure of the SW graph. Second, the IHRM model can be thought of as an infinite relational mixture model, realizing hierarchical Bayesian modeling. Third, the mixture model allows a cluster analysis providing insight into the relational domain.

The IHRM has been applied to recommender systems, for gene function prediction and to develop medical recommender systems. The IHRM was the first relational model applied to trust learning [119]. In [118] it was shown how ontological class information can be integrated into the IHRM.

### 2.4.5 Discussion

RGMs have been developed in the context of frame-based logical representations, relational data models, plate models, entity-relationship models and first-order logic but the main ideas can easily be adapted to the SW data model. One can distinguish two cases. In the first case, an RGM learns a joint probabilistic model over the complete SW or a segment of the SW. This might be the most elegant approach since there is only one (SW-) world. The draw back is that the computational requirements scale with the number of statements whose truth value is known or even the number of all potentially true statements. More appropriate for large-scale applications might be the second case where one applies the sampling approach as described in Section 2.1. As an example consider that the statistical unit is a student. A data point would then not correspond to a set of features but to a local subgraph that is anchored at the statistical unit, e.g., the student. As before sampling would make the training time essentially independent of SW-size. Ontological background knowledge can be integrated as discussed in Section 2.1. First, one can employ complete or partial materialization, which would derive statements from reasoning prior to training. Second, an ontological subgraph can be included in the subgraph of a statistical unit [118]. Also note that the MLN might be particularly suitable to exploit ontological background information: ontologies can formulate some of the first-order formulas that are the basis for the features in the MLN. PRMs have been extended to learn class hierarchies (PRM-CH), which can be a basis for ontology learning.

The RGM approaches typically make an open world assumption.<sup>8</sup> The corresponding random variables are assumed missing at random such that the approaches have an inherent mechanism to deal with missing data. If missing at random is not justified, then more complex missing data models need to be ap-

<sup>8</sup>There are some exceptions, e.g., MLN make a closed-world assumption during training.

plied. As before, based on the estimated probabilities, weighted RDF-triples can be generated and added to the SW.

## 2.5 Unstructured Data and the SW

The realization of the SW heavily depends on (1) available ontologies and (2) the annotation of unstructured data with ontology-based meta data. Manual ontology development and manual annotation are two well known SW bottlenecks. Thus learning-based approaches for both tasks are finding increasing interest [130, 63]. In this section, we will concentrate on two important tasks, namely ontology learning and semantic annotation (for a compilation of current work on ontology learning and population see, e.g., [24]). A particularly important source of information for these tasks is unstructured or semi-structured textual data. Note that there is a close relationship between textual data and SW data. Textual data describes, first, ontological concepts and relationships between concepts (e.g., a text might contain the sentence: *We all know that cats are mammals*) and, second, instances and relationships between instances (e.g., a document might inform us that: *Marry is married to Jack*). However, the input data for ontology learning and semantic annotation will not be limited to textual data; especially once the SW will be realized to a greater extent, other types of input data will become increasingly important. Learning ontologies from e.g., XML-DTDs, UML diagrams, database schemata or even raw RDF-graphs is also of great interest [34], but is out of scope here. The outline of this section is as follows: first, we consider the case, where a text corpus of interest is given and the task is to infer a prototype ontology. Second, given a text corpus and an ontology, we want to infer instances of the concepts and their relations.

### 2.5.1 Learning Ontologies from Text

Ontology learning, in general, consists of several subtasks. This includes the identification of terms, synonyms, polysems, concepts, concept hierarchies, properties, property hierarchies, domain and range constraints and class definitions. These tasks can be illustrated as the so-called ontology learning layer cake [34]. Different approaches differ mainly in the way a concept is defined and one distinguishes between formal ontologies, terminological ontologies and prototype-based ontologies [124]. In prototype-based ontologies, concepts are represented by collections of prototypical instances, which are arranged hierarchically in subclusters. An example would be the concept disease, which is defined by a set of diseases. Since prototype-based ontologies are defined by instances, they lack definitions and axiomatic grounding. In contrast, typical examples for terminological ontologies are WordNet and the Medical Subject Headings (MeSH<sup>9</sup>). Terminological ontologies are described by concept labels and both nouns and verbs are organized into hierarchies, defined by hypernym or subclass relationships. For example a disease is defined in WordNet as an impairment of health or a condition of abnormal functioning. Terminological ontologies typically also lack axiomatic grounding. A formal ontology such as OWL, in contrast, is seen as a conceptualization, whose

---

<sup>9</sup><http://www.nlm.nih.gov/mesh/>

categories are distinguished by axioms and definitions [13]. Most of the state-of-the-art approaches focus on learning prototype-based ontologies. Work on learning terminological or formal ontologies is still quite rare. Here, the big challenge is to deal with uncertain and often even contradicting extracted knowledge, introduced during the ontology learning process. This is addressed in [132], which presents a system that is able to transform a terminological ontology to a consistent formal OWL-DL ontology.

Prototype ontologies are often learned based on some type of hierarchical clustering techniques such as single-link, complete-link or average-link clustering. According to Harris' distributional hypothesis [67], semantic similarity between words can be assessed via the syntactic context, which they are sharing in a corpus. Thus most approaches base the semantic relatedness between words on some distributional similarity between the words. Usually, a vector-space model is used as input and the linguistic context of a term is described by, e.g., syntactic dependencies, which the term establishes in a corpus [69] The input vector for a term to be clustered can be, e.g., composed of syntactic expressions such as prepositional phrases following a verb or adjective modifiers. See [33] for an illustrative example for assessing the semantic similarity of terms. Hierarchical clustering, in its classical form, distinguishes between agglomerative (bottom-up) and divisive (top-down) clustering, whereas the agglomerative form is most commonly used due to its computational efficiency. Somewhat different from hierarchical clustering is the divisive bi-section-Kmeans algorithm, which yielded competitive results for document clustering [125] and has been applied to the task of learning concept hierarchies as well [98, 35]. Another variant is the the Formal Concept Analysis (FCA) [53]. FCA is closely related to bi-clustering and tries to build a lattice of so-called formal concepts from a vector space model. FCA thereby makes use of order theory and analyzes the covariance between objects and their features. The reader is referred to [53] for more information.

Recently, [34] set up a benchmark to compare the above mentioned clustering techniques for learning concept hierarchies. While each of the methods had its own benefits, FCA performed better in terms of recall and precision. All the methods just mentioned, face the problem of not being able to appropriately label the resulting clusters, i.e., to determine the name of the concept. To overcome this limitation and to guide the clustering process, [36] either use hyponyms extracted from WordNet or use Hearst patterns [68] derived either from the corpus under investigation or from the WWW.

Another type of technique for learning prototype ontologies, comes from the topic modeling community, an active research area of machine learning [111, 63]. Topic models are generative models based upon the idea that a document is made of a mixture of topics, where a topic is represented by a distribution over words. Powerful techniques such as Latent Semantic Analysis (LSA) [43], Probabilistic Latent Semantic Analysis (PLSA) [70] or Latent Dirichlet Allocation (LDA) [15] have been proposed for the automated extraction of useful information from large document collections. Applications include document annotation, query answering, document summarization, automatic topic extraction as well as trend analysis. Generative statistical models such as the ones mentioned, have been proven effective in addressing these problems. In general, the following advantages of topic models are highlighted in the context of document modeling: First, topics can be

extracted in a complete unsupervised fashion, requiring no initial labeling of the topics. Second, the resulting representation of topics for a document collection is interpretable and last but not least, each document is usually expressed by a mixture of topics, thus capturing the topic combinations that arise in documents [70, 15, 60]. When applying topic modeling techniques in an ontology learning setting, a topic is referred to as concept. To satisfy the hierarchical structure of prototype ontologies, [111] extends the PLSA method to an hierarchical version, where super concepts are introduced. While yielding already impressive results with this kind of techniques, [111] concentrates on learning prototype ontologies, where no labeling of the concept is needed. Furthermore, the hierarchy of the ontology is assumed to be known a priori. Learning the hierarchical order in topic models is an area of growing interest. Here, [14] introduced hierarchical LDA, which models the setup of the tree-structure of the topics as a Chinese Restaurant Process (CRP). As a consequence, the hierarchy is not fixed a priori, instead it is a part of the learning process. To overcome the limitation of unlabeled topics or concepts, [101] tries to automatically infer an appropriate label for multinomial topic models. [63] discusses ontology learning based on topic models in context of the SW.

### **Ontology Merging, Alignment and Evolution:**

In many cases no dominant ontology will exist, which leads to the problem that several ontologies need to be merged and aligned. In [22] these tasks have been addressed with the support of machine learning. Another aspect is that an ontology is not a rigid and fixed construct — ontologies will evolve with time. Thus, the structure of an ontology will change and new concepts will be needed to be inserted into an existing ontology. This leads to another task, where machine learning can play a role in ontology engineering: ontology refinement and ontology evolution. This task is usually treated as classification task [13]. The reader is referred to [13, 17] for more information.

## **2.5.2 Semantic Annotation**

Besides ontological support, a second prerequisite to put the SW into practice, is the availability of machine-readable meta data. Producing human readable text from SW data is simple since an RDF triple can easily be formulated as a textual statement. However, even though the statement won't be powerfully eloquent, it will still serve its purpose. The inverse is much more difficult, i.e., the generation of triples from textual data. This process is called semantic annotation, knowledge markup or meta data generation [19]. Hereby, we are following the notion of semantic annotation as linguistic annotations (such as named entities, semantic classes, etc.) as well as user annotations like tags (see the ECIR 2008 workshop on 'Exploiting Semantic Annotations in Information Retrieval'<sup>10</sup>).

The Information Extraction (IE) community provides a number of approaches for these tasks. IE is traditionally defined as the process of filling the fields and records of a database from unstructured text and is seen as precursor to data mining [100]. Usually, the fields are filled with named entities (i.e., Named Entity

---

<sup>10</sup>[http://www.yr-bcn.es/dokuwiki/doku.php?id=ecir08\\_entity\\_workshop\\_proposal](http://www.yr-bcn.es/dokuwiki/doku.php?id=ecir08_entity_workshop_proposal)

Recognition (NER)), such as persons, locations or organizations. IE first populates a database from unstructured text and data mining then aims to find patterns. IE is, dependent on the task, made up of five subtasks: segmentation, classification, finding associations and last but not least normalization and deduplication [100]. Segmentation refers to the identification of text phrases, which describe entities of interest. Classification is the assignment to predefined types of entities, while finding associations is the identification of relations between the entities (i.e., relation extraction). Normalization and deduplication describe the task of merging different text descriptions with the same meaning (e.g., mapping entities to URIs).

NER is an active field of research and several evaluation conferences such as the Message Understanding Conference (MUC-6)[61], the Conference on Computational Natural Language Learning (CoNLL-2003) [48] and in the biomedical domain, the Critical Assessments of Information Extraction systems in Biology (BioCreAtIvE I+II<sup>11</sup>) [136] have attracted a lot of interest. While in MUC-6 the focus was NER for persons, locations, organizations in an English newswire domain, CoNLL-2003 focused on language-independent NER. BioCreAtIvE focused on the recognition of biomedical entities, in this case gene and protein mentions. The methods proposed for NER vary, in general, in their degree of reliance on dictionaries, and their different emphasis on statistical or rule-based approaches. Numerous machine learning techniques have been applied to NER tasks such as Support Vector Machines [99], Hidden Markov Models [117], Maximum Entropy Markov Models [88] and Conditional Random Fields [81].

An F-measure in the mid-90s can now be achieved for extracting persons, organizations and locations in the newswire domain [100]. For extracting gene and protein mentions, however, the F-measure lies currently in the mid- to high 80s (see the BioCreAtIvE II conference for details). So NER can provide high accuracy solutions for the SW, but typically only for a small number of classes, mostly because of a limited amount of labeled training data. However, when populating an existing ontology, there will often be the need to be able to extract hundreds of classes of entities. Thus, systems which are able to scale to a large number of classes on a large amount of unlabeled data are needed. Also flexible and domain-independent recognition of entities is an important and active field of research. State-of-the-art approaches try to extract hundreds of entity classes in an unsupervised fashion [37], but so far with a fairly low accuracy. Promising areas, which could help to overcome current limitations of supervised IE systems, are semi-supervised learning [30, 2] as well as active learning [80].

The same entities can have different textual representation (e.g., ‘Clark Kent’, ‘Kent Clark’ and ‘Mr. Clark’ refer to the same person). Normalization is the process of standardizing the textual expressions. This task is usually also referred to as entity resolution, co-reference resolution or normalization and deduplication. The Stanford Entity Resolution Framework (SERF), e.g., has the goal to provide a framework for generic entity resolution [8]. Other techniques for entity resolution employ relational clustering [12] as well as probabilistic topic models [11].

Another important task is the identification of relations between instances of concepts (i.e., the association finding stage in the traditional IE workflow). Up to now, most of research on text information extraction has focused on tagging named entities. The Automatic Content Extraction (ACE) program provides annotation

---

<sup>11</sup><http://biocreative.sourceforge.net/biocreative.2.html>

benchmark sets for the challenging task of relation extraction. At ACE, this task is called Relation Detection and Characterization (RDC). A representative system using an SVM with a rich set of features, reports results for Relation Detection (74.7% F-measure) and 68.0% F-measure for the RDC task [139]. Co-occurrence based relation extraction is a simple, effective and popular method [116], but usually suffers of a lower recall, since entities can co-occur for many other reasons. Other methods are kernel-based [26] or rule-based [109]. Recently, [25] propose a new method that treats relation extraction as sequential labeling task. They extend Conditional Random Fields (CRFs) towards the extraction of semantic relations. Hereby, they focus on the extraction of relations between genes and diseases (five types of relations) as well as between disease and treatment entities (eight types of relations). The work applies the authors' method to a biomedical textual database and provides the resulting network of genes and diseases in a machine-readable RDF graph. Thereby, gene and disease entities are normalized to Bio2RDF<sup>12</sup> URIs.

## 2.6 First Experiments in the Analysis of FOAF-data

The purpose of the FOAF (Friend of a Friend) project [21] is to create a web of machine-readable pages describing people, the relationships between people and people's activities and interests, using W3C's RDF technology. The FOAF ontology is defined using RDFS/OWL and is formally specified in the FOAF Vocabulary Specification 0.91 [20]. In our study we employed the IHRM model as described in Section 2.4. The trained IHRM can, for instance, recommend new friendships, the affiliations of persons, and their interests and projects. Furthermore one might want to predict attributes of certain persons, like their gender or age. Finally, by interpreting the clustering results of the IHRM one can answer typical questions from social network analysis concerning the relationships between members of the FOAF social system.

In general FOAF data is either uploaded by each person individually or generated automatically from user profiles of community websites like Tribe.net, LiveJournal.com or my.opera.com. The resulting network of linked FOAF-files can be gathered using a FOAF harvester, a so called "scutter". Some scutter dumps are readily available for download, e.g., in one large rdf/xml-file or stored in a relational database.

Even though this use case only covers a very basic statistical inference problem on the SW, there still are major challenges to meet. First, there are characteristics of the FOAF-data that need special consideration: For instance, the actual data is extremely sparse. With more than 100000 users, there are far more potential links as actual links between persons.

Another typical characteristic of friendship data is that the topology of the *knows*-RDF-graph consists of a few barely connected star graphs, corresponding to a few active network users with a long list of friends as the "center" of the stars and the mass of users that don't specify their friends. Second, there are prevalent challenges of SW data in general that can also be observed in a FOAF analysis. For instance, there is a variety of additional untested and potentially conflicting

---

<sup>12</sup><http://bio2rdf.org/>

ontologies specified by users. If this information is ignored by only considering data consistent with the FOAF ontology, most of the information specified by users is ignored. This also applies to the almost arbitrary use of literals by users. For instance the relation *interest* with range *Document* defined in the FOAF-schema is in reality mostly used with a literal instead. Consequently, this results in a loss of semantic information. To still make use of this information one would, e.g., need to use automated semantic annotation as described in Section 2.5. Another preprocessing step that needs to be considered in practice is the materialization of triples, which can be inferred deductively. For example there might be an instance of the relation *holdsAccount* with domain *Person* in the data, which is not given in the schema. However, from the ontology it can be inferred that *Person* is a *subClassOf Agent* which in turn has a property *holdsAccount*. As stated before, total materialization is only feasible in less expressive ontologies.

Considering these issues, it becomes clear that there are not only theoretical but also a large number of interesting practical challenges for learning on the SW.

## 2.7 Conclusions

Data in Semantic Web formats will bring many new opportunities and challenges to machine learning. Machine learning complements ontological background knowledge by exploiting regularities in the data while being robust against some of the inherent problems with Semantic Web data such as contradicting information and non-stationarity. A general issue with machine learning is that the problem of missing information needs to be carefully addressed in learning, in particular if either the selection of statistical units or the probability that a feature is missing depend on the features of interest, which is common in many-to-many relations.

We began with a section on feature-based statistical learning on the Semantic Web. This procedure is widely applicable, scales well with the size of the Semantic Web and provides a promising general purpose learning approach. The greatest challenge here is that feature-based statistical learning has no inherent way of dealing with missing data requiring additional missing data models. A common situation in social network data is that features in linked objects are mutually dependent and need to be modeled jointly. One can expect that SW learning will benefit from ongoing research in social network modeling.

We then presented the main approaches in inductive logic programming. Inductive logic programming has the potential to learn deterministic constraints that can be integrated into the employed ontology. We presented a discussion on learning with relational matrices, which is quite attractive if multiple many-to-many relations are of interest, as in recommendation systems. We then studied relational graphical models. Although these approaches were originally defined in various frameworks, e.g., frame-based logical representation, relational data models, plate models, entity-relationship models and first-order logic, they can easily be modified to be applicable in context of the Semantic Web. Relational graphical models are capable of learning a global probabilistic Semantic Web model and inherently can deal with data missing at random. Scalability to the size of the Semantic Web might be a problem for RGMs and we discussed subgraph sampling as a possible solution. All approaches have means to include ontological background knowledge by complete or partial materialization. In addition, the ontological RDF-graph

can be incorporated in learning and ontological features can be derived and exploited. Ontologically supported machine learning is an active area of research. It is conceivable that in future ontological standards, the developed statistical models could become an integral part of the ontology. Also, we have discussed that most presented approaches can be used to produce statements that are weighted by their probability value derived from machine learning, complementing statements that are derived from logical reasoning. An interesting opportunity is to include weighted triples in Semantic Web queries.

We reported about initial work on learning ontologies from textual data and on the semantic annotation of unstructured data. So far, this concerns the most advanced work in Semantic Web learning covering ontology construction and management, ontology evaluation, ontology refinement, ontology evolution, as well as the mapping, merging and alignment of ontologies. In addition there is growing work on Semantic Web mining extending the capabilities of standard web mining, although most of this work needs to wait for the Semantic Web to be realized on a large scale.

In summary, machine learning has the potential to realize a number of exciting applications on the Semantic Web and can complement axiomatic inference by exploiting regularities in the data.

## 2.8 Conclusions in the Context of LarKC

This concerns the LarKC tasks

- 3.1 Training set retrieval
- 3.2 Active learning and incomplete data
- 3.3 Abstraction/predicate invention/feature generation
- 3.4 Optimize learner and ontology integration

### 2.8.1 Feature-based Statistical Learning on the SW

The first focus is on the implementation of the feature-based learning approach described in Section 2.1. The main steps are described in Figure 2.1 (bottom). This approach is most suitable in context of LarKC. The approach is very general and highly scalable since the determining factor is the number of statistical units in the sample, which basically is independent of the size of the SW. Sampling is an issue for learning but also for the inference algorithms developed in other work packages, so we can exploit synergies.

The main steps are:

- Definition of the statistical unit
- Definition of the population
- Definition of the training set; ideally the training set should fulfill two requirements: first, it should be a random sample of the population and, second, one needs to be able to compute all features of interest for the statistical unit in the training data set (see also the following subsection)

- Alternatively we will implement an active (i.e. non random) data selection algorithm
- Definition of the queries to be executed for the views
- Definition of the features that are derived from the views
- Selection of the machine learning algorithm to be used; it is planned that WEKA models can be used for training
- Definition of the test set to which the trained model is applied. The test set is a subset of the population disjunct from the training set.
- Specification, which machine learning results will be written into the data base as a weighted triple; the triple can then be used as part of queries.

All these steps need to be defined by the user in appropriate form (XML-file). Alternatively, the user simply specifies the triple, whose truth value is of interest and all steps are performed automatically based on some default settings.

### 2.8.2 Missing Data

The main issue that needs to be addressed concerns missing data in training and testing.

The following approaches will be implemented:

- The first solution is to make a closed-world assumption and, by definition, all information is certain.
- The second solution is to apply a standard statistical approach for handling missing data (default imputation, mean imputation).
- The third case is that the missing values are coupled. A common situation is that statistical units (e.g., persons) are linked by triples (e.g., friendship) and that the statements of one statistical unit (e.g., a club membership) depend on the statements of linked statistical units (e.g., the club memberships of friends). The prediction of statements for one statistical unit will depend on the prediction of statements for linked statistical units which means that samples are not independent. In fact it is more realistic to consider that training and test set are part of a common RDF-graph with some statements known to be true and some statements unknown. We will develop solutions for this common situation.

### 2.8.3 Search for Best Features

This will initially not be the focus but might be addresses at a later stage in the program (compare, Subsection 2.1.3).

### 2.8.4 Integration of Ontological Background Knowledge

As indicated in the paper, ontological information is part of the RDF/RDFS graph and can be integrated as features in learning. We will provide means for doing that (compare, Subsection 2.1.4).

## 2.8.5 RGM: IHRM

We will provide an implementation of the IHRM model, which is suitable for predictive modeling and for latent space modeling (i.e., clustering) (compare, Subsection 2.4.4).

## 3 DATA STREAMING TECHNOLOGIES FOR DATA ABSTRACTION

### 3.1 Introduction

In many real-world applications data take the form of continuous streams instead of the form of finite data sets stored in a traditional repository; this is the case for monitoring of network traffic, for telecommunications management, for click-streams, for manufacturing, for sensor networks, and many many others. In such applications, instead of classical “one-shot” queries, clients need to register continuously running queries, which return new results as new data arrive on the streams.

A data stream is a sequence of items received continuously and in real-time, ordered either implicitly, by arrival time, or explicitly, by means of timestamps. Not only it is typically impossible to control the order in which items arrive, but, even more importantly, it is not feasible to locally store a stream in its entirety [57].

Due to all these peculiar characteristics, traditional database systems and data processing algorithms are not suitable for handling numerous and complex continuous queries over data streams; ad-hoc data management systems have been studied and developed since the late nineties.

Processing of data streams has been largely investigated in the last decade [54], specialized Data Stream Management Systems (DSMSs) have been developed, and features of DSMSs are becoming supported by major database products, such as Oracle and DB2.

Stream Database Management Systems (DSMS) represent a paradigm change in the database world because they move from persistent relations to transient streams, with the innovative assumption that streams can be consumed on the flight (rather than stored forever) and from user-invoked queries to continuous queries, i.e., queries which are persistently monitoring streams and are able to produce their answers even in the absence of invocation. DSMSs can support parallel query answering over data originating in real time and can cope with burst of data by adapting their behavior and gracefully degrading answer accuracy by introducing higher approximations.

In the paper accepted at FIS 2008 (see Appendix 1), we present a conceptual architecture for bringing Stream Database results within the pluggable algorithmic framework of LarkC. While investigating such conceptual architecture we have realized that that the first research issue to be addressed was the specification of data streams for the Semantic Web (namely RDF data streams) and a continuous query language for the Semantic Web (namely C-SPARQL).

C-SPARQL is an extension of SPARQL for querying data streams in the context of RDF repositories. A RDF data stream is a timed sequence of RDF triples produced by a data source. The distinguishing feature of C-SPARQL is the support for continuous queries, i.e. SPARQL-like queries registered over RDF data streams in the context of a C-SPARQL execution environment and then continuously executed; C-SPARQL queries consider *windows* of such streams, i.e. the most recent triples in the streams. Extending SPARQL into C-SPARQL requires several additions for defining streams, their windows, their time stamps, and lan-

guage constructs for summarizing stream information, which are described in this report.

Data streams are modeled as streams of RDF triples; in a given stream, any of the subject, predicate and object can either be fixed or time-varying. Data streams are associated with each C-SPARQL query by specifying their IRI, the window of items to be considered by the query, and the interval between two consecutive executions of the same query.

C-SPARQL queries produce as output the same output types as SPARQL: yes/no, selection of values of the variables matching the query pattern, construction of new RDF triples or molecules from these values, or description of resources. If however the query includes data streams as sources, then the first three outputs are also continuously renewed with each query execution. In addition, C-SPARQL allows the construction of new RDF data streams.

One of the most important operations upon data streams is aggregation, e.g. for counting the number of elements in the stream satisfying a given condition. Therefore, C-SPARQL supports aggregation as a qualifying extension to SPARQL. We have opted to keep aggregation and stream management orthogonal; therefore it is possible to define and implement the support for aggregates even without introducing data streams.

While the above elements constitute powerful extensions of SPARQL, we have decide to introduce them in the context of a well-defined and theoretically sound language fragment. We have therefore focused on *well designed queries* as defined in [112]; this assumption guarantees the equivalence of the operational and algebraic semantics of C-SPARQL, at least for those queries with no data streams or aggregates. Under the above assumptions, graph patterns of a SPARQL query can be reduced to a normal form which presents the UNION of union-free subpatterns, and that these can be further reduced to a form exhibiting AND conditions following by OPTIONAL conditions. At this time, we conjecture that the optimization of C-SPARQL queries in a distributed context (such as the larKC platform) will be possible through algebraic operations that add aggregation and streams to the operators presented in [112] and then use classical results of stream and distributed database management to deal with data streaming, fragmentation, and partitioning.

The rest of the section is organized as follows. Section 3.2 summarizes the restrictions on SPARQL that are required in order to obtain "well designed" queries and illustrates the structure of the normal form. Section 3.3 introduces aggregates, by defining their syntax and informal semantics and by providing some running examples. Section 3.4 introduces data streams, again by introducing their syntax, informal semantics, and running examples. Section 3.5 shows C-SPARQL at work in the urban computing use case. Finally, Section 3.6 sketches some initial hypothesis and conjectures about C-SPARQL optimization; Section 3.7 illustrates previous work in data streams and SPARQL, and specifically marks the differences between our approach and the one proposed in [18], by stressing the limitations of that approach that make it hardly usable in the larKC context. Finally, Section 3.8 illustrates how we intend to progress with this research.

## 3.2 Well-Designed Queries for SPARQL

In this paper we consider the following assumptions concerning graph patterns, built-in conditions, and variables, which were developed by Perez, Arenas and Gutierrez in [112]:

- Let  $P$  be a graph pattern and  $R$  a built-in condition such that  $(P \text{ FILTER } R)$  appears in a query  $Q$ ; let  $\text{var}(R)$  and  $\text{var}(P)$  denote the set of variables occurring in  $R$  and  $P$ , respectively; then, we assume  $\text{var}(R) \subseteq \text{var}(P)$ .
- Let  $P$  be the graph pattern of a query  $Q$  and  $P' = (P_1 \text{ OPT } P_2)$  be a sub pattern of  $P$ ; then for every variable  $?X$  occurring in  $P$ , if  $?X$  occurs both inside  $P_2$  and outside  $P'$ , then it also occurs in  $P_1$ .

These assumptions exclude cases which are not computationally desirable and are hardly applicable to real-life situations; under these two assumptions, [112] proves that the depth first operational semantics and the bottom up algebraic semantics defined in [39, 114] coincide, and shows the transformations which are needed in order to turn arbitrary graph patterns into a normal form consisting of unions of the following pattern:

$$(\dots(t_1 \text{ AND } \dots \text{ AND } t_k) \text{ OPT } O_1) \text{ OPT } O_2) \dots) \text{ OPT } O_n$$

Finally, the queries satisfying this pattern have an algebraic evaluation strategy, also defined in [112], on the basis of the algebraic operations upon mappings, including join, union, difference, and outer join.

## 3.3 Aggregation in (C-)SPARQL

SPARQL specification lacks aggregation functions. Hereafter, we provide a way to introduce aggregates in SPARQL as the first element characterizing our C-SPARQL proposal. Aggregation functions are specified in C-SPARQL by adding a new AGGREGATES clause. In the clause each aggregation function is expressed in terms of three elements:

- The first element is a new variable;
- The second element is an aggregation function (one of: COUNT, MAX, MIN, SUM, AVG); COUNT has no argument, while the other functions take one of the variables occurring in the WHERE clause as arguments;
- The third place is a set of one or more variables occurring in the WHERE clause.

The syntax for the AGGREGATES clause is:

```
AggregateClause ::= 'AGGREGATES' '{' Aggregation
                  ( '.' Aggregation )* [ '.' Filter ] '}'
Aggregation ::= '(' var ',' Function '(' var ')' ',' Group ')'
Function ::= 'COUNT' | 'SUM' | 'AVG' | 'MIN' | 'MAX'
Group ::= var | '{' var ( ',' var )* '}'
```

A classic example of aggregate for C-SPARQL computes the sum of amounts of all transactions of the accounts of all customers of a bank, and then selects for each customer such aggregate value:

```

PREFIX ns: <http://example.org/ns#>

SELECT DISTINCT ?owner ?balance
WHERE { ?owner ns:owns ?account .
         ?account ns:has ?transaction .
         ?transaction ns:with ?amount . }
AGGREGATES {(?balance, SUM(?amount), ?owner) }
  
```

The informal semantics of the above query consists of adding to the matching values computed by the WHERE clause (for variables ?owner, ?account, ?transaction, and ?amount) new matching values for the new variable ?balance, and then projecting over ?owner and ?balance. The value for ?balance of a given tuple is obtained by computing the aggregate function that sums all the amounts of a given owner; thus, variable ?owner plays the same role as the attribute appearing in the GROUP-BY clause of SQL.

Notice that, if  $t$  is a tuple extracted by matching the WHERE clause upon a given dataset  $D$  and  $t'$  is a tuple obtained after evaluating the AGGREGATES clause, then  $t$  is equal to the restriction of  $t'$  over all variables except the new ones, as the effect of computing AGGREGATES is to add, for every function appearing in the clause, a new variable which has a matching value for every tuple  $t$ ; if the grouping variable is not bound in  $t$ , the result of computing the aggregate is also not bound.

Of course, if grouping variables are two or more, then the grouping function is applied to every distinct n-ple of the grouping variables. The next example shows how balances are produced for each distinct account of each user:

```

PREFIX ns: <http://example.org/ns#>

SELECT DISTINCT ?owner ?account ?balance
WHERE { ?owner ns:owns ?account .
         ?account ns:has ?transaction .
         ?transaction ns:with ?amount . }
AGGREGATES {(?balance, SUM(?amount), {?owner, ?account}) }
  
```

The tuples extracted after evaluating aggregate functions can be filtered (as in standard FILTER clause in SPARQL), by using built-in conditions, which may include all the variables existing after the evaluation of the AGGREGATES clause. For instance, the following query extracts for every owner his trust, filters the transactions to be dated in the current year, and then filters those tuples in which the balance is less than the individual trust, selecting in the result also the count of transactions performed during the year:

```

PREFIX ns: <http://example.org/ns#>

SELECT DISTINCT ?owner ?balance ?transnum
WHERE { ?owner ns:is_given ?trust .
        ?owner ns:owns ?account .
        ?account ns:has ?transaction .
        ?transaction ns:with ?amount .
        ?transaction ns:on ?date .
        FILTER ( ?date > "1-1-2008"^^xsd:date )
      }
AGGREGATES { (?transnum, COUNT, ?owner).
              (?balance, SUM(?amount), ?owner).
              FILTER (?trust < ?balance) }
  
```

The AGGREGATES extension with FILTER covers the expressive power of SQL GROUP BY and HAVING clauses. As in SQL, it is a good practice to select grouping variables and aggregate functions in the query result; the DISTINCT clause eliminates duplicates from the query result, so as to produce one tuple for each distinct group.

Note that the proposed C-SPARQL syntax is more liberal than the corresponding SQL syntax, because each aggregate function can use a different set of grouping variables, and because filters after aggregation are arbitrary boolean expressions combining the variables produced in the AGGREGATES clause with the variables produced in the WHERE clause. This enables a higher expressive power which appears in line with the spirit of SPARQL, e.g. in defining OPTIONAL bindings.

### 3.4 Stream Management in C-SPARQL

In this section, we define C-SPARQL, a language for continuous SPARQL queries, both in terms of an informal semantics and a concrete query language that implements the semantics. The C-SPARQL language is based on three data types: instantaneous RDF graphs (shortly named tgraph), instantaneous relations (shortly named trelation), and RDF streams (shortly named stream). Tgraphs and trelations exist also in conventional SPARQL; stream is a new data type<sup>1</sup>. The semantics of SPARQL can be described by five classes of transformer operators:

1. Tgraph-to-trelation operators take one or more tgraphs as input and produce a trelation as output, yielding the same effect as SELECT and ASK queries in SPARQL.
2. Tgraph-to-tgraph operators take one or more tgraphs as input and produce a tgraph as output, yielding the same effect as CONSTRUCT and DESCRIBE queries in SPARQL.
3. Stream-to-trelation operators take one or more streams and zero or more tgraphs as input and produce a trelation as output, by extending SELECT queries.

<sup>1</sup>in a similar way, the stream type extends the relation type in CQL, an SQL-like query language for dealing with data streams, see Section ??

4. Stream-to-tgraph operators take one or more streams and zero or more tgraphs as input and produce a tgraph as output, by extending CONSTRUCT queries.
5. Stream-to-stream operators take one or more stream and zero or more tgraphs as input and produce a stream as output, by extending CONSTRUCT queries.

Classes (1, 2) correspond to standard SPARQL, while classes (3, 4, 5) are extensions introduced by C-SPARQL. We next define the informal semantics of these three classes of queries.

We assume that each data stream is associated with a distinct IRI, which is a locator of the actual data source of the stream. We also assume that every C-SPARQL query must be registered in the context of a C-SPARQL execution platform. Therefore, the description of how a query interacts with a data stream is split in two distinct clauses, the registration statement of the query and the FROM clauses introducing the data streams which are used by the query. We first consider the latter case.

```

StreamClause ::= FROM STREAM StreamIRI
               '[ ' RANGE WindowClause ' ]'
               [ 'AS' StreamName ]
WindowClause ::= LogicalWindow | PhysicalWindow
LogicalWindow ::= Number TimeUnit WindowOverlap
TimeUnit ::= 'MSEC' | 'SEC' | 'MIN' | 'HOUR' | 'DAY'
WindowOverlap ::= 'STEP' Number TimeUnit | 'TUMBLING'
PhysicalWindow ::= 'TRIPLES' Number
  
```

For each stream appearing in a query, the FROM STREAM clause defines its IRI, an optional name (scoped within the query) and the window of observation of the stream by the query. The notion of window hereby proposed for C-SPARQL is compatible with the notion of window supported by most continuous query languages.

A window extracts from the stream the *last* data stream elements, which are considered by the query. Such extraction can be *physical* (a given number of triples) or *logical* (triples which appear during a given time interval). Thus, a window can include the last 5 items in the stream, or a variable number of incoming items in the last 5 minutes.

Windows are denoted as TUMBLING (or non-overlapping) when they are kept unchanged during the considered unit of time and then recomputed at expiration (hence, every element of the data stream appears exactly in one of the windows considered by the query); otherwise, they are denoted as SLIDING [59], when windows are progressively advanced according to a second time interval, which should be smaller than the window interval. In such case, every element of the data stream may appear in more than one window considered by the query.

A classic example of stream management is the association of a given computer (represented by its MAC address) to a given profile (e.g. teenager interested in rock CDs) that can be done on the basis of the computer's recent hits. Assuming that hits are already associated with profiles, the next query counts how many hits of a given profile come from each computer in the last 10 minutes; the sliding

window is modified every minute. Note that each computer can hit the same profile multiple times, each one with a different stream triple.

```

PREFIX ns: <http://example.org/ns#>

SELECT DISTINCT ?mac ?profile ?hits-by-mac
FROM STREAM <http://www.example.org/hits.trdf> [RANGE 10 MIN STEP
  1 MIN]
WHERE { ?mac ns:hits ?profile }
AGGREGATES { (?hits-by-mac, COUNT, {?mac, ?profile})}
  
```

The window considers all the triples that are in the input in the last 10 minutes, and is advanced every minute. This means that at every new minute new triples enter into the window and old tuples exit from the window. Note that the result of the aggregation does not change during the slide interval, therefore also the query result does not change during the slide interval; it changes instead at every slide change (i.e., at every minute).

In this stream, as in all the streams that we will use in the examples of this report, the predicate of the triple is fixed (hits) while the subject and object part of the triple are variable (?mac, ?profile). Thus, a physical source for this stream will have items consisting of pairs of values. This arrangement is coherent with RDF repositories whose predicates are taken from a small vocabulary constituting a sort of schema, but C-SPARQL makes no assumption on variable bindings of its stream triples.

The query hereafter combines the sensors data in the stream with the static knowledge about (a) where the sensors are placed and (b) the capacity of the streets.

```

PREFIX c: <http://linkedurbandata.org/city#>
PREFIX t: <http://linkedurbandata.org/traffic#>

REGISTER QUERY CongestionsInMilanoArea-4
  COMPUTED EVERY 1 MIN AS

SELECT DISTINCT ?street
FROM STREAM
  <http://linkedurbandata.org/IT/milan/area-4/sensor.trdf> [RANGE
  10 MIN STEP 1 MIN]

WHERE { ?sensor c:placedIn ?street .
  ?street c:hasCapacity ?capacity .
  ?sensor t:counts ?passages .
  ?passages t:numberOfCars ?amount . }
AGGREGATES {
  ( ?total-passages, SUM(?amount), ?street ) .
FILTER ( ?total-passages > (0.8 * ?capacity))}
  
```

```

PREFIX ns: <http://example.org/ns#>

SELECT DISTINCT ?mac ?new_prof
FROM STREAM <http://www.example.org/hits.trdf> [RANGE 10000 TUPLES]
WHERE { ?mac ns:hits ?new_prof .
        ?mac ns:matches ?old_prof .
        FILTER ( ?new_prof != ?old_prof )
      }
AGGREGATES { (?x, COUNT, {?mac, ?new_prof}).
        FILTER ( ?x > 5 ) }
  
```

Note that in this case the WHERE clause combines triples from a static RDF repository and from a stream. In this example, the tuples produced by the query may change at each new data item in the stream. This syntax introduced so far, however, does not tell how the output should be produced by the query; this part is specified in the registration clause, discussed next.

Each C-SPARQL query is registered by means of the following statement:

```

Registration ::= 'REGISTER QUERY' QueryName
               [ 'COMPUTED EVERY' Number TimeUnit ]
               'AS' C-SPARQLQuery
  
```

This statement gives to each continuous query a name and indicates when the query must be computed; specifically, the COMPUTED EVERY clause indicates at which time the query should be computed. The clause can be omitted, in which case, as default, the query is computed at every change of the window of the first stream listed in the query itself; an explicit indication can override the default.

The meaning of continuously computing query Q - producing as output either yes/no, or the selection of values of the variables matching the query pattern, or the construction of new RDF triples from these values - is to continuously renew the output at each query execution. Such output could be used to plot the traffic on a Web site as a function of time, or be periodically transferred to a reasoner for further analysis. In addition, C-SPARQL allows the construction of new RDF data streams, by supporting the possibility to register CONSTRUCT queries as STREAM, according to the following syntax:

```

Registration ::= 'REGISTER STREAM' StreamName
               [ 'COMPUTED EVERY' Number TimeUnit ]
               'AS' ConstructQuery
  
```

Note that the construct query may produce from a minimum of one triple to a maximum of an entire graph. In the former case, a different timestamp is assigned to every triple produced by the query; in the latter case, the same timestamp is assigned to all the triples which are produced by the same evaluation of the query. In either cases, timestamps are system-generated in monotonic order.

For instance, the following query produces a stream of triples, each one representing the new profile matched to a given computer; note that this example uses the same logical conditions as the query above, but it constructs the output in the format of a stream of RDF triples.

```

PREFIX ns: <http://example.org/ns#>

REGISTER STREAM ProfileStream AS
CONSTRUCT { ?mac ns:matches ?new_prof }
FROM STREAM <http://www.example.org/hits.trdf> [RANGE 10000 TUPLES]
WHERE { ?mac ns:hits ?new_prof .
        ?mac ns:matches ?old_prof .
        FILTER ( ?new_prof != ?old_prof )
        }
AGGREGATES { (?x, COUNT, {?mac, ?new_prof}) .
        FILTER ( ?x > 5 ) }
  
```

On the contrary, the following CONSTRUCT query builds an RDF stream in which each element consists of a simple graph that includes two triples: a the new (empty) variable is bound to the profile and mac extracted by the evaluation of the WHERE and AGGREGATES clauses. Triples produced by the same construction carry the same timestamp. Note also that the query uses a physical window, therefore outputs at each new input at most one graph.

```

PREFIX ns: <http://example.org/ns#>

REGISTER STREAM ProfileStream AS
CONSTRUCT { _:a ns:matches ?new_prof .
        _:a ns:user ?mac }
FROM STREAM <http://www.example.org/hits.trdf> [RANGE 10000
        TUPLES] AS HitStream
WHERE { ?mac ns:hits ?new_prof .
        ?mac ns:matches ?old_prof .
        FILTER ( ?new_prof != ?old_prof )
        }
AGGREGATES { (?x, COUNT, {?mac, ?new_prof}) .
        FILTER ( ?x > 5 ) }
  
```

So far we discuss timestamps as system-generated values that cannot be exposed as result of the query, however the timestamp of a stream element can also be retrieved and bound to a variable by means of a Timestamp function. Such Timestamp function has two arguments:

- The first is the name of a stream (names should be assigned within the optional AS subclause of the FROM STREAM clause);
- The second is the name of a variable that is bound by pattern matching with an RDF triple of that stream.

If the variable is not bound, the timestamp function yields a *null* value; if the variable gets bound multiple times, the function produces the most recent timestamp value.

For instance, the following CONSTRUCT query extends the RDF stream such that a the new (empty) variable is bound, in addition to the new profile and to the mac, also to the time at which the new profile has been observed. Timestamp values can also be used within filters (e.g. for comparing timestamps of items from different streams).

Note that the timestamp function can compute different values for the same variable in the context of a given query execution; when the timestamp function is retrieved, as in this example, each such value is added to the result pattern table as an annotation, so as to enable the selection of the correct value for each variable binding.

```

PREFIX ns: <http://example.org/ns#>

REGISTER STREAM ProfileStream AS
CONSTRUCT { _:a ns:matches ?new_prof .
              _:a ns:user ?mac .
              _:a ns:atTime timestamp(HitStream, ?new_prof ) }
FROM STREAM <http://www.example.org/hits.trdf> [RANGE 10000
          TUPLES] AS HitStream
WHERE { ?mac ns:hits ?new_prof .
          ?mac ns:matches ?old_prof .
          FILTER ( ?new_prof != ?old_prof )
          }
AGGREGATES { (?x, COUNT, {?mac, ?new_prof}) .
              FILTER ( ?x > 5 ) }
  
```

### 3.5 Urban Computing Scenario

In this section, we show C-SPARQL at work on Urban Computing, one of the use case scenarios of larKC. The following is sample data expressed in the Urban-Computing ontology; it is the minimal information needed to understand the examples in this document.

```

PREFIX uc: <http://www.larkc.eu/ontologies/urban-computing#>

uc:IT-MI-VialeCorsica a uc:Street .
uc:IT-MI-VialeCorsica uc:hasLength "3400"xsd:integer .
uc:IT-MI-VialeCorsica uc:hasMaxSpeed "50"xsd:integer .
uc:IT-MI-VialeCorsica uc:hasThroughput "2500"xsd:float .

uc:IT-MI-District-4 a uc:District .
uc:IT-MI-District-4 uc:contains uc:IT-MI-VialeCorsica .
uc:IT-MI-District-4 uc:traffic "752"xsd:integer .

uc:IT-MI-S-42 a uc:Sensor .
uc:IT-MI-S-42 uc:placedIn uc:IT-MI-VialeCorsica .
uc:IT-MI-S-42 uc:measures _:m
_:m uc:numberOfCars "42"xsd:integer .
_:m uc:numberOfTrucks "11"xsd:integer .
_:m uc:numberOfOther "7"xsd:integer .
_:m uc:numberOfVehicles "60"xsd:integer .

uc:IT-AZ420CR a uc:Vehicle .
uc:IT-AZ420CR uc:travelsThrough uc:IT-MI-VialeCorsica .
uc:IT-AZ420CR uc:hasSpeed "110"^^xsd:integer .
uc:IT-AZ420CR uc:hasHighSpeedOn uc:IT-MI-VialeCorsica .
uc:IT-AZ420CR a uc:Pirate .
  
```

```

uc:TrafficLevelLow   a uc:TrafficLevel .
uc:TrafficLevelMid   a uc:TrafficLevel .
uc:TrafficLevelHigh  a uc:TrafficLevel .
uc:TrafficLevelLow   uc:hasMinValue "0"^^xsd:integer .
uc:TrafficLevelLow   uc:hasMaxValue "33"^^xsd:integer .
uc:TrafficLevelMid   uc:hasMinValue "34"^^xsd:integer .
uc:TrafficLevelMid   uc:hasMaxValue "66"^^xsd:integer .
uc:TrafficLevelHigh  uc:hasMinValue "67"^^xsd:integer .
uc:TrafficLevelHigh  uc:hasMaxValue "100"^^xsd:integer .
uc:IT-MI-VialeCorsica uc:trafficLevel uc:TrafficLevelMid .
  
```

Queries exhibit progressive complexity, from selection, to aggregation, to stream composition, to classification.

### 3.5.1 Aggregation Queries

Query 1 introduces aggregation, by counting how many vehicles are present at all sensors which are located in the same street.

```

PREFIX uc: <http://www.larkc.eu/ontologies/urban-computing#>

SELECT DISTINCT ?street ?sensor ?total
WHERE {
  ?sensor uc:placedIn ?street .
  ?sensor uc:measures ?x .
  ?x uc:numberOfVehicles ?vehicleCount .
}
AGGREGATES {(?total, SUM(?vehicleCount), {?street, ?sensor})}
  
```

Query 2 performs multiple aggregations (for cars, truck, and other vehicles respectively) for all the sensors which are located in the same street.

```

PREFIX uc: <http://www.larkc.eu/ontologies/urban-computing#>

SELECT DISTINCT ?street ?sensor ?CarBalance ?TruckBalance
              ?OtherVehicleBalance
WHERE {
  ?sensor uc:placedIn ?street .
  ?sensor uc:measures ?x .
  ?x uc:numberOfcars ?CarAmount .
  ?x uc:numberOfTrucks ?TruckAmount .
  ?x uc:numberOfOtherVehicles ?OtherVehicleAmount .
}
AGGREGATES {
  (?CarBalance, SUM(?CarAmount), {?street, ?CarAmount}) .
  (?TruckBalance, SUM(?TruckAmount), {?street, ?TruckAmount}) .
  (?OtherVehicleBalance, SUM(?OtherVehicleAmount),
    {?street, ?OtherVehicleAmount}) .
}
  
```

### 3.5.2 Stream Selection Query

Query 3 is a simple query against a physical window over a stream, which returns all vehicles with speed over the threshold. The query is registered to create a triple

every time a new triple in sensors.trdf occurs.

```

PREFIX uc: <http://www.larkc.eu/ontologies/urban-computing#>

REGISTER STREAM high_speed AS
CONSTRUCT {?vehicle uc:hasHighSpeedOn ?street}
FROM STREAM <http://uc.larkc.eu/sensors.trdf> [TRIPLES 1]
WHERE {
  ?vehicle uc:travelsThrough ?street .
  ?vehicle uc:hasSpeed ?speed .
  ?street uc:hasMaxSpeed ?maxspeed .
  FILTER (?speed > ?maxspeed)
}
  
```

Results of Query 3 are shown in Table 3.1. Note that every triple in the constructed stream is timestamped; timestamps can be considered as annotations of RDF triples which are not part of the triple itself.

Table 3.1: Query 3 Results

| Timestamp | Vehicle | has high speed on | Street                  |
|-----------|---------|-------------------|-------------------------|
| $t_1$     | uc:V001 | uc:hasHighSpeedOn | uc:IT-MI-ViaDigione     |
| $t_2$     | uc:V001 | uc:hasHighSpeedOn | uc:IT-MI-ViaWashington  |
| $t_3$     | uc:V002 | uc:hasHighSpeedOn | uc:IT-MI-ViaSardegna    |
| $t_4$     | uc:V002 | uc:hasHighSpeedOn | uc:IT-MI-PiazzaPiemonte |
| $t_5$     | uc:V002 | uc:hasHighSpeedOn | uc:IT-MI-ViaElba        |

Query 4 searches for vehicles whose speed has been over the threshold for more than 5 times in 1 hour. The query is computed every second, while the window slide is changed every minute. If the vehicles are not yet included in the pirates list, they are marked as pirates and retrieved together with the number of violations that triggered the condition.

```

PREFIX uc: <http://www.larkc.eu/ontologies/urban-computing#>

REGISTER STREAM new_pirates COMPUTED EVERY 1 SEC AS
CONSTRUCT {
  ?vehicle a uc:Pirate .
  ?vehicle uc:numberOfRecentViolations ?countViolations }
FROM STREAM <http://uc.larkc.eu/highspeed.trdf> [RANGE 1 HOURS
  STEP 1 MIN]
WHERE {
  ?vehicle uc:hasHighSpeedOn ?street .
  ?v a uc:Pirate .
  FILTER (?v != ?vehicle)
}
AGGREGATES {(?countViolations, COUNT, ?vehicle) .
  FILTER (?count > 5) }
  
```

Results of Query 4 are shown in Table 2; note that pairs of triples carry the same timestamp as they are produced by the same construction.

Table 3.2: Query 4 Results

| Timestamp | Subject | Predicate                   | Object    |
|-----------|---------|-----------------------------|-----------|
| $t_{143}$ | V001    | a                           | uc:Pirate |
| $t_{143}$ | V001    | uc:numberOfRecentViolations | 3         |
| $t_{144}$ | V007    | a                           | uc:Pirate |
| $t_{144}$ | V007    | uc:numberOfRecentViolations | 7         |

### 3.5.3 Stream Composition Queries

Query 5 is an example of composition of three streams (with the same structure) located in the same district. The query counts vehicles passing through sensors every 5 minutes, generating a stream of aggregate traffic status in a district. The TUMBLING window property means that two executions of the query can't count the same vehicle passage more than once.

```

PREFIX uc: <http://www.larkc.eu/ontologies/urban-computing#>

REGISTER STREAM district_traffic COMPUTED EVERY 1 MIN AS
CONSTRUCT {?district uc:traffic ?count}
FROM STREAM <http://uc.larkc.eu/streetA.trdf> [RANGE 5 MIN
TUMBLING]
FROM STREAM <http://uc.larkc.eu/streetB.trdf> [RANGE 5 MIN
TUMBLING]
FROM STREAM <http://uc.larkc.eu/streetC.trdf> [RANGE 5 MIN
TUMBLING]
WHERE { ?vehicle uc:travelsThrough ?street .
        ?district uc:contains ?street .
        }
AGGREGATES {(?count, COUNT, ?district)}
    
```

Results of Query 5 are shown in Table 3.3.

Table 3.3: Query 5 Results

| Timestamp | district          | traffic    | count |
|-----------|-------------------|------------|-------|
| $t_{400}$ | uc:IT-MI-Baggio   | uc:traffic | 150   |
| $t_{401}$ | uc:IT-MI-Barona   | uc:traffic | 220   |
| $t_{402}$ | uc:IT-MI-Fiera    | uc:traffic | 380   |
| $t_{403}$ | uc:IT-MI-Centrale | uc:traffic | 260   |

The following Query 6 is a variant of Query 5 composing three streams with different structures (i.e., different predicate names). The GRAPH clause is used to indicate which stream to look for a given graph pattern; the property “travelsThrough” is used in stream A, while “passesBy” is used in stream B and “isIn” is used in stream C.

```

PREFIX uc: <http://www.larkc.eu/ontologies/urban-computing#>
PREFIX x1: <http://www.x1.org/ontologies/urban-computing#>
PREFIX x2: <http://www.x2.org/ontologies/urban-computing#>

REGISTER STREAM district_traffic COMPUTED EVERY 1 MIN AS
CONSTRUCT {?district uc:traffic ?count}
FROM STREAM <http://uc.larkc.eu/streetA.trdf> [RANGE 5 MIN
  TUMBLING]
FROM STREAM <http://uc.x1.org/streetB.trdf> [RANGE 5 MIN TUMBLING]
FROM STREAM <http://uc.x2.org/streetC.trdf> [RANGE 5 MIN TUMBLING]
WHERE {
  GRAPH <http://uc.larkc.eu/streetA.trdf> {
    ?vehicle uc:travelsThrough ?streetA .
    ?district uc:contains ?streetA
  } UNION
  GRAPH <http://uc.x1.org/streetB.trdf> {
    ?vehicle x1:passedBy ?streetB .
    ?district uc:contains ?streetB
  } UNION
  GRAPH <http://uc.x2.org/streetC.trdf> {
    ?vehicle x2:isIn ?streetC .
    ?district uc:contains ?streetC
  }
}
AGGREGATES {(?count, COUNT, ?district)}

```

### 3.5.4 Classification queries

Query 7 maps traffic values into a 3-values scale (TrafficLevelLow, TrafficLevelMid, TrafficLevelHigh).

```

PREFIX uc: <http://www.larkc.eu/ontologies/urban-computing#>

REGISTER STREAM district_traffic COMPUTED EVERY 1 MIN AS
CONSTRUCT {?street uc:trafficLevel ?trafficLevel}
FROM STREAM <http://uc.larkc.eu/street/A.trdf> [RANGE 5 MIN
  TUMBLING]
FROM STREAM <http://uc.larkc.eu/street/B.trdf> [RANGE 5 MIN
  TUMBLING]
FROM STREAM <http://uc.larkc.eu/street/C.trdf> [RANGE 5 MIN
  TUMBLING]
WHERE { ?vehicle uc:travelsThrough ?street .
  ?district uc:contains ?street .
}
AGGREGATES {(?count, COUNT, ?street) .
  FILTER (?count >= ?min AND ?count <= ?max) }

```

Results of Query 7 are shown in Table 4.

Query 8 retrieves all the streets which have been full for 80% or more of their capacity in the last minute.

Table 3.4: Query 7 Results

| Timestamp | Street                  | traffic    | Traffic Level       |
|-----------|-------------------------|------------|---------------------|
| $t_{400}$ | uc:IT-MI-ViaDigione     | uc:traffic | uc:TrafficLevelLow  |
| $t_{401}$ | uc:IT-MI-ViaWashington  | uc:traffic | uc:TrafficLevelMid  |
| $t_{402}$ | uc:IT-MI-PiazzaPiemonte | uc:traffic | uc:TrafficLevelHigh |
| $t_{403}$ | uc:IT-MI-ViaSardegna    | uc:traffic | uc:TrafficLevelLow  |

```

PREFIX uc: <http://www.larkc.eu/ontologies/urban-computing#>

REGISTER QUERY full_streets AS
SELECT { ?street }
FROM STREAM <http://uc.larkc.eu/street/A.trdf> [RANGE 1 MIN
TUMBLING]
FROM STREAM <http://uc.larkc.eu/street/B.trdf> [RANGE 1 MIN
TUMBLING]
FROM STREAM <http://uc.larkc.eu/street/C.trdf> [RANGE 1 MIN
TUMBLING]
...
WHERE { ?vehicle uc:travelsThrough ?street .
?street uc:hasThroughput ?capacity . }
AGGREGATES {(?count, COUNT, ?street)}
FILTER ( ?capacity < (0.8 * ?count) )
    
```

Results of Query 8 are shown in Table 5.

Table 3.5: Query 8 Results

| Street                 |
|------------------------|
| uc:IT-MI-ViaWashington |
| uc:IT-MI-VialeUmbria   |

### 3.6 Optimization Scenarios for C-SPARQL

We envision execution scenarios for C-SPARQL consisting of a collection of several interconnected nodes, each one supporting the execution of C-SPARQL queries. Nodes could have limitations/specializations, i.e. the ability of supporting pure SPARQL (with no data stream or aggregation) or specific stream-focused queries (with no capability of integration with a global RDF repository). Optimization in this context yields to several interesting problems.

- The first issue concerns the development of analysis techniques for multiple queries, in order to recognize their common parts and reformulate the queries so as to compute common parts only once. This is a classical problem of continuous queries, known as multi-query optimization problem [122].
- Another issue concerns the development of incremental evaluation strategies that take into account the particular features of windows used in the query

in order to compute the result at the next iteration as a function of the deltas that are produced due to incoming and outgoing triples in the window. This is as well a classic problem of continuous queries [135].

- Another interesting optimization problem concerns the incremental materialization of knowledge that could be available to higher-order RDF-based reasoners (e.g., reasoners supporting RDF-S or OWL) when derived knowledge depends upon streaming data. Again, we can envision scenarios where knowledge at the next iteration could be computed as a function of the deltas that are produced due to incoming and outgoing triples in the window. This is an application of continuous queries to scenarios which support derived data, studied in [133].
- Another issue, specific to larKC, concerns the ability to dynamically adapt query computations under heavy query loads, e.g. by sampling input streams or by relaxing the window refreshment. Queries could be registered together with confidence limits so as to help the system's tuning. Sampling functions could be made available as part of the query language so as to enable their explicit declaration within queries.
- Another issue, also specific to larKC, concerns the distribution of query execution nodes close to data stream sources. The idea is that each source could provide input stream of arbitrary nature and then specific nodes could perform the transformation of such streams into summarized RDF triples and then transmit them to higher-level nodes for more complex computations. Nodes local to streams could support only stream-specific operations. Optimization in this case could reuse classic results of distributed database systems [28].

Thus, the study of C-SPARQL execution environment is suggestive of several important research directions, which may reuse and adapt classic query optimization results. In the course of the larKC project, we will choose the most promising and interesting ones and concentrate on them.

## 3.7 Previous Related Work

This section illustrates previous work on data streams and then on the SPARQL language.

### 3.7.1 Data Streams

In many real-world applications data take the form of continuous streams instead of the form of finite data sets stored in a traditional repository; this is the case for monitoring of network traffic, for telecommunications management, for click-streams, for manufacturing, for sensor networks, and many many others. In such applications, instead of classical “one-shot” queries, clients need to register continuously running queries, which return new results as new data arrive on the streams. A data stream is a sequence of items received continuously and in real-time, ordered either implicitly, by arrival time, or explicitly, by means of timestamps. Not

only it is typically impossible to control the order in which items arrive, but, even more importantly, it is not feasible to locally store a stream in its entirety [57]. Due to all these peculiar characteristics, traditional database systems and data processing algorithms are not suitable for handling numerous and complex continuous queries over data streams; ad-hoc data management systems have been studied and developed since the late nineties.

One of the first proposed models for data streams was the Chronicle data model [73]. It introduced the concept of chronicles as append-only ordered sequences of tuples, together with a restricted view definition language and an algebra that operates over chronicles as well as over traditional relations. OpenCQ [95] and NiagaraCQ [31] addressed continuous queries for monitoring persistent data sets spread over wide-area networks. Another data stream management system is Aurora [7], which in turn evolved into the Borealis project [1], which addresses the distribution issues.

In [5], Babu et al. tackle the problem of continuous queries over data streams addressing semantic issues as well as efficiency concerns. They specify a general and flexible architecture for query processing in the presence of data streams. This research evolved into the specification and development of a query language tailored for data streams, named CQL [3, 4]. Further optimizations are discussed in [105].

Another stream of research was developed by Law et al. [86], putting particular emphasis on the problem of mining data streams [87]. Another project that addresses data mining issues is the Stream Mill project [6], which extensively considered the problem of data aggregation. Its query language (ESL) efficiently supports physical and logical windows (with optional slides and tumbles) on both built-in aggregates and user-defined aggregates. The constructs introduced in ESL extend the power and generality of DSMSs.

The problem of processing delays is one of the most critical issues and at the same time a strong quality requirement for many data stream applications, since the value of query results decreases dramatically over time as the delays sum up. In [32], the authors address the problem of keeping delays below a desired threshold in situations of overload, which are common in data stream systems. The framework described in the paper is built on top of the Borealis platform.

As for the join over data streams, rewriting techniques are proposed in [58] for streaming aggregation queries, studying the conditions under which joins can be optimized and providing error bounds for results of the rewritten queries. The basis of the optimization is a theory in which constraints over data streams can be formulated and the result error bounds are specified as functions of the boundary effects incurred during query rewriting.

### 3.7.2 SPARQL

The most authoritative source about the syntax and semantics of the SPARQL language is the W3C recommendation [114].

Cyganiak [39] presents a relational model of SPARQL. The author uses relational algebra operators (join, left outer join, projection, selection, etc.) to model the SPARQL SELECT clauses. A translation system between SPARQL and SQL is outlined. The system extensively resorts to the use of COALESCE and IS NULL

in order to express some SPARQL features. Harris [66] presents an implementation of SPARQL queries over a relational database engine. The use of relational algebra operators is similar to that of [39].

In [23], a logical reconstruction of RDF family of languages is given. The authors prove the equivalence of their framework to W3C definition of RDF, getting complexity results and a model theoretic semantics.

In [64], Gutierrez et al. discuss the semantics and the computational complexity of a conjunctive query language for RDF with basic patterns, which is a formal and unambiguous basis for defining the semantics of SPARQL queries evaluation. In [112], Perez et al. consider simple RDF graphs (without special semantics for literals) and a simplified version of filters; these assumptions allow them to provide a compositional semantics, to prove that there is a normal form in which, under certain constraints over variable bindings, a wide range of queries can be expressed, to fix some complexity bounds, and to discuss optimization opportunities. Haase et al. [65] present a comparison of functionalities of pre-SPARQL query languages, many of which gave inspirations to the definition of SPARQL.

As for streams of RDF data, a previous attempt to extend SPARQL to support data streams is [18]. This work introduces a syntax for the specification of logical and physical windows in SPARQL queries by means of local grammar extensions and grounds their semantics in terms of new algebraic operators. One of the main differences between this work and our proposal is the support for aggregates. However, without aggregates it is hard to specify many significant computations over streams, that are very useful in real-life scenarios.

### 3.8 Conclusions

This report has sketched the syntax and semantics of Continuous SPARQL, a new language for querying RDF-based repositories supporting aggregation and augmented with data streams. At this stage, we have selected the ingredients of the language and produced a first proposal for syntax, which has been tested on the Urban Computing use case. We have defined streams through orthogonal concepts (such as registration parameters, from stream clause, window parameters, timestamp assignment and query); Although the semantics has only been sketched so far, we believe that the concepts are well-selected and that, therefore, it will be possible to progressively clarify the semantics by adding aggregation and streams as orthogonal extensions. In the future we plan to:

- Consolidate the syntax, based upon other on-paper experiments, and define a C-SPARQL language dialect hierarchy, where the language or dialect elements will be identified by grammar productions.
- Specify the semantics, most likely with a bottom-up, algebraic approach which is most amenable to optimization. The initial reduction to well-designed queries guarantees the equivalence of top-down and bottom-up semantics at least on the SPARQL-compatible core of C-SPARQL.
- Define the ingredients of a C-SPARQL execution environment, by taking into accounts the specificities of the larKC project.

- Define query optimization problems for C-SPARQL, along the lines that we have sketched in Section 6, and then propose and evaluate solutions for some of them.
- Develop and experiment with simple prototypes that may show the use of C-SPARQL applied to selection, aggregation, and reasoning tasks.

---

## REFERENCES

- [1] Daniel J Abadi, Yanif Ahmad, Magdalena Balazinska, Ugur Cetintemel, Mitch Cherniack, Jeong-Hyon Hwang, Wolfgang Lindner, Anurag S Maskey, Alexander Rasin, Esther Ryvkina, Nesime Tatbul, Ying Xing, and Stan Zdonik. The Design of the Borealis Stream Processing Engine. In *Second Biennial Conference on Innovative Data Systems Research (CIDR 2005)*, Asilomar, CA, January 2005.
- [2] Rie Kubota Ando and Tong Zhang. A high-performance semi-supervised learning method for text chunking. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, 2005.
- [3] Arvind Arasu, Brian Babcock, Shivnath Babu, Mayur Datar, Keith Ito, Itaru Nishizawa, Justin Rosenstein, and Jennifer Widom. Stream: the stanford stream data manager (demonstration description). In *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 665–665, New York, NY, USA, 2003. ACM.
- [4] Arvind Arasu, Shivnath Babu, and Jennifer Widom. The cql continuous query language: semantic foundations and query execution. *The VLDB Journal*, 15(2):121–142, 2006.
- [5] Shivnath Babu and Jennifer Widom. Continuous queries over data streams. *SIGMOD Record*, 30(3):109–120, 2001.
- [6] Yijian Bai, Hetal Thakkar, Haixun Wang, Chang Luo, and Carlo Zaniolo. A data stream language and system designed for power and extensibility. In *CIKM*, pages 337–346, 2006.
- [7] Hari Balakrishnan, Magdalena Balazinska, Don Carney, Ugur Cetintemel, Mitch Cherniack, Christian Convey, Eddie Galvez, Jon Salz, Michael Stonebraker, Nesime Tatbul, Richard Tibbetts, and Stan Zdonik. Retrospective on aurora. *The VLDB Journal*, 13(4):370–383, 2004.
- [8] Omar Benjelloun, Hector Garcia-Molina, David Menestrina, Qi Su, Steven E. Whang, and Jennifer Widom. Swoosh: A generic approach to entity resolution. *VLDB Journal*, 2008.
- [9] Bettina Berendt, Andreas Hotho, and Gerd Stumme. Towards semantic web mining. In *ISWC '02: Proceedings of the First International Semantic Web Conference on The Semantic Web*. Springer-Verlag, 2002.
- [10] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American*, 2001.
- [11] Indrajit Bhattacharya and Lise Getoor. A latent dirichlet model for unsupervised entity resolution. In *The 6th SIAM Conference on Data Mining (SIAM SDM-06)*, 2006.
- [12] Indrajit Bhattacharya and Lise Getoor. Collective entity resolution in relational data. *ACM Transactions on Knowledge Discovery from Data*, 1(1), 2007.

- 
- [13] Chris Biemann. Ontology learning from text: A survey of methods. *LDV Forum*, 20(2), 2005.
- [14] David M. Blei, Thomas L. Griffiths, Michael I. Jordan, and Joshua B. Tenenbaum. Hierarchical topic models and the nested chinese restaurant process. In *Advances in Neural Information Processing Systems*, 2003.
- [15] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3, 2003.
- [16] Hendrik Blockeel and L. De Raedt. Top-down induction of first-order logical decision trees. *Artificial Intelligence*, 101(1-2), 1998.
- [17] Stephan Bloehdorn, Peter Haase, York Sure, and Johanna Voelker. Ontology evolution. In John Davies, Rudi Studer, and Paul Warren, editors, *Semantic Web Technologies*. John Wiley and Sons, 2006.
- [18] Andre Bolles, Marco Grawunder, and Jonas Jacobi. Streaming sparql - extending sparql to process data streams. In *ESWC*, pages 448–462, 2008.
- [19] Kalina Bontcheva, Hamish Cunningham, Atanas Kiryakov, and Valentin Tablan. Semantic annotation and human language technology. In John Davies, Rudi Studer, and Paul Warren, editors, *Semantic Web Technologies*. John Wiley and Sons, 2006.
- [20] Dan Brickley and Libby Miller. *FOAF Vocabulary Specification*. <http://xmlns.com/foaf/spec/>.
- [21] Dan Brickley and Libby Miller. *The Friend of a Friend (FOAF) project*. <http://www.foaf-project.org/>.
- [22] Jos de Bruijn, Marc Ehrig, Cristina Feier, Francisco Martin-Recuerda, Francois Scharffe, and Moritz Weiten. Ontology mediation, merging, and aligning. In John Davies, Rudi Studer, and Paul Warren, editors, *Semantic Web Technologies*. John Wiley and Sons, 2006.
- [23] Jos De Bruijn, Enrico Franconi, and Sergio Tessaris. Logical reconstruction of normative rdf. In *In OWL: Experiences and Directions Workshop (OWLED-2005)*, 2005.
- [24] Paul Buitelaar and Philipp Cimiano. *Ontology Learning and Population: Bridging the Gap between Text and Knowledge*. IOS Press, 2008.
- [25] M. Bundschuh, M. Dejori, M. Stetter, V. Tresp, and H.P. Kriegel. Extraction of semantic biomedical relations from text using conditional random fields. *BMC Bioinformatics*, 9, 2008.
- [26] Razvan C. Bunescu and Raymond J. Mooney. Subsequence kernels for relation extraction. In *Advances in Neural Information Processing Systems*, 2005.
- [27] George Casella and Roger L. Berger. *Statistical Inference*. Duxbury Press, 1990.

- 
- [28] Stefano Ceri and Giuseppe Pelagatti. *Distributed databases principles and systems*. McGraw-Hill, Inc., New York, NY, USA, 1984.
- [29] S. Chakrabarti, B. Dom, and P. Indyk. Enhanced hypertext categorization using hyperlinks. In *SIGMOD*, 1998.
- [30] Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien. *Semi-Supervised Learning*. The MIT Press, 2006.
- [31] Jianjun Chen, David J. DeWitt, Feng Tian, and Yuan Wang. NiagaraCQ: A scalable continuous query system for internet databases. In Weidong Chen, Jeffrey F. Naughton, and Philip A. Bernstein, editors, *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA*, pages 379–390. ACM, 2000.
- [32] Yi cheng Tu, Song Liu, Sunil Prabhakar, and Bin Yao. Load shedding in stream databases: a control-based approach. In *In Proc. Int. Conf. on Very Large Data Bases (VLDB)*, pages 787–798, 2006.
- [33] Ph Cimiano and St Staab. Learning concept hierarchies from text with a guided agglomerative clustering algorithm. In *Proceedings of the ICML 2005 Workshop on Learning and Extending Lexical Ontologies with Machine Learning Methods*, 2005.
- [34] Philipp Cimiano. *Ontology Learning and Population from Text: Algorithms, Evaluation and Applications*. Springer-Verlag, 2006.
- [35] Philipp Cimiano, Andreas Hotho, and Steffen Staab. Comparing conceptual, divide and agglomerative clustering for learning taxonomies from text. In *Proceedings of the 16th European Conference on Artificial Intelligence, ECAI'2004*, 2004.
- [36] Philipp Cimiano and Steffen Staab. Learning concept hierarchies from text with a guided agglomerative clustering algorithm. In Chris Biemann and Gerhard Paas, editors, *Proceedings of the ICML 2005 Workshop on Learning and Extending Lexical Ontologies with Machine Learning Methods*, 2005.
- [37] Philipp Cimiano and Johanna Völker. Towards large-scale, open-domain and ontology-based named entity classification. In G. Angelova, K. Bontcheva, R. Mitkov, and N. Nicolov, editors, *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP)*, 2005.
- [38] William W. Cohen and Haym Hirsh. Learning the CLASSIC description logic: Theoretical and experimental results. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourth International Conference (KR94)*, 1994.
- [39] R. Cyganiak. A relational algebra for sparql. Technical report, HP-Labs.
- [40] L. De Raedt. Attribute-value learning versus inductive logic programming: The missing links (extended abstract). In *ILP '98: Proceedings of the 8th International Workshop on Inductive Logic Programming*. Springer-Verlag, 1998.
-

- 
- [41] L. De Raedt and L. Dehaspe. Clausal discovery. *Machine Learning*, 26, 1997.
- [42] L. De Raedt and Wim. Van Laer. Inductive constraint logic. In *ALT '95: Proceedings of the 6th International Conference on Algorithmic Learning Theory*, 1995.
- [43] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41, 1990.
- [44] P. Domingos and M. Richardson. Markov logic: A unifying framework for statistical relational learning. In L. Getoor and B. Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press, 2007.
- [45] S. Džeroski. Inductive logic programming in a nutshell. In L. Getoor and B. Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press, 2007.
- [46] P. Edwards, G. Grimnes, and A. Preece. An empirical investigation of learning from the semantic web. In *ECML/PKDD, Semantic Web Mining Workshop*, 2002.
- [47] Werner Emde and Dietrich Wettschereck. Relational instance based learning. In Lorenza Saitta, editor, *Machine Learning - Proceedings 13th International Conference on Machine Learning*, 1996.
- [48] F. Erik, T.K. Sang, and F. De Meulder. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In Walter Daelemans and Miles Osborne, editors, *Proceedings of CoNLL-2003*, 2003.
- [49] L. Fahrmeir, R. Künstler, I. Pigeot, G. Tutz, A. Caputo, and Stefan Lang. *Arbeitsbuch Statistik*. Springer, fourth edition, 2004.
- [50] Dieter Fensel, James A. Hendler, Henry Lieberman, and Wolfgang Wahlster. *Spinning the Semantic Web: Bringing the World Wide Web to its Full Potential*. MIT Press, 2003.
- [51] Blaz Fortuna, Marko Grobelnik, and Dunja Mladenic. Ontogen: Semi-automatic ontology editor. In *HCI (9)*, 2007.
- [52] Yoshio Fukushige. Representing probabilistic relations in rdf. In *ISWC-URSW*, 2005.
- [53] Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer-Verlag, 1997.
- [54] Minos Garofalakis, Johannes Gehrke, and Rajeev Rastogi. *Data Stream Management: Processing High-Speed Data Streams (Data-Centric Systems and Applications)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.
- [55] L. Getoor, N. Friedman, D. Koller, A. Pferrer, and B. Taskar. Probabilistic relational models. In L. Getoor and B. Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press, 2007.
-

- 
- [56] L. Getoor, N. Friedman, D. Koller, and B. Taskar. Learning probabilistic models of link structure. *Journal of Machine Learning Research*. *Journal of Machine Learning Research*, 2002.
- [57] Lukasz Golab, David DeHaan, Erik D. Demaine, Alejandro López-Ortiz, and J. Ian Munro. Identifying frequent items in sliding windows over on-line packet streams. In *Internet Measurement Conference*, pages 173–178, 2003.
- [58] Lukasz Golab, Theodore Johnson, Nick Koudas, Divesh Srivastava, and David Toman. Optimizing away joins on data streams. In *SSPS*, pages 48–57, 2008.
- [59] Lukasz Golab and M. Tamer Özsu. Processing sliding window multi-joins in continuous queries over data streams. In *VLDB*, pages 500–511, 2003.
- [60] T. L. Griffiths and M. Steyvers. Finding scientific topics. *Proc. Natl. Acad. Sci. USA*, 2004.
- [61] Ralph Grishman and Beth Sundheim. Design of the MUC-6 evaluation. In *MUC6 '95: Proceedings of the 6th conference on Message understanding*, 1995.
- [62] Marko Grobelnik and Dunja Mladenic. Automated knowledge discovery in advanced knowledge management. *Library Hi Tech News incorporating Online and CD Notes*, 9(5), 2005.
- [63] Marko Grobelnik and Dunja Mladenic. Knowledge discovery for ontology construction. In John Davies, Rudi Studer, and Paul Warren, editors, *Semantic Web Technologies*. John Wiley and Sons, 2006.
- [64] Claudio Gutierrez, Carlos Hurtado, and Alberto O. Mendelzon. Foundations of semantic web databases. In *PODS '04: Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 95–106, New York, NY, USA, 2004. ACM.
- [65] Peter Haase, Jeen Broekstra, Andreas Eberhart, and Raphael Volz. A comparison of rdf query languages. In *Proceedings of the Third International Semantic Web Conference*, pages 502–517, 2004.
- [66] Steve Harris. Sparql query processing with conventional relational database systems. In *International Workshop on Scalable Semantic Web Knowledge Base Systems*, pages 235–244, 2005.
- [67] Zellig S. Harris. *Mathematical Structures of Language*. John Wiley and Sons, 1968.
- [68] Marti A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics*, 1992.
- [69] Donald Hindle. Noun classification from predicate-argument structures. In *Meeting of the Association for Computational Linguistics*, 1990.

- 
- [70] Thomas Hofmann. Probabilistic latent semantic analysis. In *Uncertainty in Artificial Intelligence (UAI)*, 1999.
- [71] Incubator Group. *Uncertainty Reasoning for the World Wide Web*. W3C, <http://www.w3.org/2005/Incubator/urw3/>, 2005.
- [72] Manfred Jaeger. Relational bayesian networks. In *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence (UAI)*, 1997.
- [73] H. V. Jagadish, Inderpal Singh Mumick, and Abraham Silberschatz. View maintenance issues for the chronicle data model. pages 241–252, 1999.
- [74] Aram Karalič and Ivan Bratko. First order regression. *Machine Learning*, 26(2-3), 1997.
- [75] Charles Kemp, Joshua B. Tenenbaum, Thomas L. Griffiths, Takeshi Yamada, and Naonori Ueda. Learning systems of concepts with an infinite relational model. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2006.
- [76] Kristian Kersting and L. De Raedt. Bayesian logic programs. Technical report, Albert-Ludwigs University at Freiburg, 2001.
- [77] Atanas Kiryakov. Measurable targets for scalable reasoning. *Ontotext Technology White Paper*, 2007.
- [78] Daphne Koller and Avi Pfeffer. Probabilistic frame-based systems. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 1998.
- [79] S. Kramer, N. Lavrac, and P. Flach. From propositional to relational data mining. In S. Džeroski and L. Lavrac, editors, *Relational Data Mining*. Springer-Verlag, 2001.
- [80] Trausti T. Kristjansson, Aron Culotta, Paul A. Viola, and Andrew McCallum. Interactive information extraction with constrained conditional random fields. In *Nineteenth National Conference on Artificial Intelligence (AAAI)*, 2004.
- [81] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of 18th International Conference on Machine Learning*, 2001.
- [82] N. Landwehr, A. Passerini, L. De Raedt, and Frasconi. kFOIL: Learning simple relational kernels. In *National Conference on Artificial Intelligence (AAAI)*, 2006.
- [83] Niels Landwehr, Kristian Kersting, and L. De Raedt. nFOIL: Integrating naïve bayes and FOIL. In M. Veloso and S. Kambhampati, editors, *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05)*, 2005.
- [84] LarKC. *The large Knowledge Collider*. EU FP 7 Large-Scale Integrating Project, <http://www.larkc.eu/>, 2008.
-

- 
- [85] Nada Lavrač, Sašo Džeroski, and Marko Grobelnik. Learning nonrecursive definitions of relations with LINUS. In *EWSL-91: Proceedings of the European working session on learning on Machine learning*, 1991.
- [86] Yan-Nei Law, Haixun Wang, and Carlo Zaniolo. Query languages and data models for database sequences and data streams. In *VLDB*, pages 492–503, 2004.
- [87] Yan-Nei Law and Carlo Zaniolo. An adaptive nearest neighbor classification algorithm for data streams. In *PKDD*, pages 108–120, 2005.
- [88] Yi-Feng Lin, Tzong-Han Tsai, Wen-Chi Chou, Kuen-Pin Wu, Ting-Yi Sung, and Wen-Lian Hsu. A maximum entropy approach to biomedical named entity recognition. In *Proceedings of 4th ACM SIGKDD Workshop on Data Mining in Bioinformatics (BioKDD)*, 2004.
- [89] C. Lippert, Y. Huang, S. H. Weber, V. Tresp, M. Schubert, and H.-P. Kriegel. Relation prediction in multi-relational domains using matrix factorization. Technical report, Siemens, 2008.
- [90] Francesca A. Lisi. Principles of inductive reasoning on the semantic web: A framework for learning in AL-Log. In F. Fages and S. Soliman, editors, *Principles and Practice of Semantic Web Reasoning, Series: Lecture Notes in Computer Science*, 2005.
- [91] Francesca A. Lisi. The challenges of the semantic web to machine learning and data mining. In *Tutorial at ECML 2006*, 2006.
- [92] Francesca A. Lisi. A methodology for building semantic web mining systems. In *The 16th International Symposium on Methodologies for Intelligent Systems*, 2006.
- [93] Francesca A. Lisi. Practice of inductive reasoning on the semantic web: A system for semantic web mining. In *Principles and Practice of Semantic Web Reasoning, 4th International Workshop, PPSWR 2006*, 2006.
- [94] Roderick J. A. Little and Donald B. Rubin. *Statistical Analysis with Missing Data*. John Wiley and Sons, second edition, 2002.
- [95] Ling Liu, Calton Pu, and Wei Tang. Continual queries for internet scale event-driven information delivery. *IEEE Trans. Knowl. Data Eng.*, 11(4):610–628, 1999.
- [96] Q. Lu and L. Getoor. Link-based classification. In *ICML*, 2003.
- [97] S. Macskassy and F. Provost. Classification in networked data: a toolkit and a univariate case study. *Machine Learning*, 2007.
- [98] Alexander Maedche and Steffen Staab. Semi-automatic engineering of ontologies from text. In *Proceedings of the 12th International Conference on Software Engineering and Knowledge Engineering*, 2000.

- [99] James Mayfield, Paul McNamee, and Christine Piatko. Named entity recognition using hundreds of thousands of features. In *Proceedings of the seventh conference on natural language learning*, 2003.
- [100] Andrew McCallum. Information extraction: distilling structured data from unstructured text. *Queue*, 3(9), 2005.
- [101] Qiaozhu Mei, Xuehua Shen, and ChengXiang Zhai. Automatic labeling of multinomial topic models. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2007.
- [102] D. Mladenic, M. Grobelnik, B. Foruna, and M. Grcar. Knowledge discovery for the semantic web. *Submitted*, 2008.
- [103] S. Muggleton. Inverse entailment and Progol. *New Generation Computing, Special issue on Inductive Logic Programming*, 13(3-4), 1995.
- [104] S. Muggleton and C. Feng. Efficient induction of logic programs. In *Proceedings of the 1st Conference on Algorithmic Learning Theory*. Ohmsma, Tokyo, Japan, 1990.
- [105] Kamesh Munagala, Utkarsh Srivastava, and Jennifer Widom. Optimization of continuous queries with shared expensive filters. In *PODS*, pages 215–224, 2007.
- [106] J. Neville and D. Jensen. Iterative classification in relational data. In *AAAI*, 2000.
- [107] J. Neville and D. Jensen. Relational dependency networks. *Journal of Machine Learning Research*, 2007.
- [108] Jennifer Neville and David Jensen. Dependency networks for relational data. In *ICDM '04: Proceedings of the Fourth IEEE International Conference on Data Mining*, 2004.
- [109] T. Ono, H. Hishigaki, A. Tanigami, and T. Takagi. Automated extraction of information on protein-protein interactions from the biological literature. *Bioinformatics*, 17(2), 2001.
- [110] Ontoprise. Neue Version des von Ontoprise entwickelten Ratgebersystems beschleunigt die Roboterwartung. *Ontoprise Pressemitteilung*, 2007.
- [111] Gerhard Paaß, Jörg Kindermann, and Edda Leopold. Learning prototype ontologies by hierarchical latent semantic analysis. In *Knowledge Discovery and Ontologies*, 2004.
- [112] Jorge Pérez, Marcelo Arenas, and Claudio Gutierrez. Semantics and complexity of sparql. In *International Semantic Web Conference*, pages 30–43, 2006.
- [113] A. Popescul and L. H Ungar. Feature generation and selection in multi-relational statistical learning. In L. Getoor and B. Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press, 2007.

- 
- [114] Eric Prud'hommeaux and Andy Seaborne. SPARQL Query Language for RDF. <http://www.w3.org/TR/rdf-sparql-query/>.
- [115] J. R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5(3), 1990.
- [116] A. K. Ramani, R. C. Bunescu, R. J. Mooney, and E. M. Marcotte. Consolidating the set of known human protein-protein interactions in preparation for large-scale mapping of the human interactome. *Genome Biol*, 6(5), 2005.
- [117] Soumya Ray and Mark Craven. Representing sentence structure in hidden markov models for information extraction. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, 2001.
- [118] S. Reckow and V. Tresp. Integrating ontological prior knowledge into relational learning. Technical report, Siemens, 2007.
- [119] A. Rettinger, M. Nickles, and V. Tresp. A statistical relational model for trust learning. In *Proceeding of 7th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, 2008.
- [120] Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine Learning*, 62(1-2), 2006.
- [121] Celine Rouveirol and Veronique Ventos. Towards learning in CARIN-ALN. In *International Workshop on Inductive Logic Programming*, 2000.
- [122] Prasan Roy, S. Seshadri, S. Sudarshan, and Siddhesh Bhole. Efficient and extensible algorithms for multi query optimization. *SIGMOD Rec.*, 29(2):249–260, 2000.
- [123] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad. Collective classification in network data. *AI Magazine (Special Issue on AI and Networks)*, forthcoming, 2008.
- [124] John F. Sowa. Ontology, metadata, and semiotics. In *International Conference on Computational Science*, 2000.
- [125] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. In *KDD Workshop on Text Mining*, 2000.
- [126] Gabor Takacs, Istvan Pitaszy, Bottyan Nemeth, and Domonkos Tikk. On the gravity recommendation system. In *Proceedings of KDD Cup and Workshop 2007*, 2007.
- [127] Benjamin Taskar, Pieter Abbeel, and Daphne Koller. Discriminative probabilistic models for relational data. In *Uncertainty in Artificial Intelligence (UAI)*, 2002.
- [128] Volker Tresp. Committee machines. In Yu Hen Hu and Jenq-Nen Hwang, editors, *Handbook for Neural Network Signal Processing*. CRC Press, 2001.
- [129] William Trochim. *The Research Methods Knowledge Base*. Atomic Dog Publishing, second edition, 2000.
-

- [130] Frank van Harmelen. Semantische Techniken stehen kurz vor dem Durchbruch. *C'T Magazine*, 2007.
- [131] W. Van Laer and L. De Raedt. How to upgrade propositional learners to first order logic: A case study. In *Machine Learning and Its Applications, Advanced Lectures*, 2001.
- [132] Johanna Völker, Peter Haase, and Pascal Hitzler. Learning expressive ontologies. In Paul Buitelaar and Philipp Cimiano, editors, *Ontology Learning and Population: Bridging the Gap between Text and Knowledge*. IOS Press, 2008.
- [133] Raphael Volz, Steffen Staab, and Boris Motik. Incremental maintenance of materialized ontologies. In *In 2nd Int. Conf. on Ontologies and Databases (ODBASE)*, 2003.
- [134] Zhao Xu, Volker Tresp, Kai Yu, and Hans-Peter Kriegel. Infinite hidden relational models. In *Uncertainty in Artificial Intelligence (UAI)*, 2006.
- [135] Jun Yang and Jennifer Widom. Incremental computation and maintenance of temporal aggregates. *The VLDB Journal*, 12(3):262–283, 2003.
- [136] A. Yeh, A. Morgan, M. Colosimo, and L. Hirschman. Biocreative task 1a: gene mention finding evaluation. *BMC Bioinformatics*, 6, 2005.
- [137] Kai Yu, Wei Chu, Shipeng Yu, Volker Tresp, and Zhao Xu. Stochastic relational models for discriminative link prediction. In *Advances in Neural Information Processing Systems 19*, 2006.
- [138] Shipeng Yu, Kai Yu, and Volker Tresp. Soft clustering on graphs. In *Advances in Neural Information Processing Systems 18*, 2005.
- [139] Guodong Zhou, Jian Su, Jie Zhang, and Min Zhang. Exploring various knowledge in relation extraction. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, 2005.
- [140] X. Zhu. Semi-supervised learning literature survey. Technical report, Technical Report 1530, Department of Computer Sciences, University of Wisconsin, Madison, 2005.

## 4 APPENDIX

In the following pages we add as appendixes the paper “*A First Step Towards Stream Reasoning*” accepted at FIS 2008.

# A First Step Towards Stream Reasoning

Emanuele Della Valle, Stefano Ceri, Davide F. Barbieri, Daniele Braga and  
Alessandro Campi

Dip. di Elettronica e Informazione, Politecnico di Milano, Milano, Italy  
email: {name.surname}@polimi.it

**Abstract.** While reasoners are year after year scaling up in the classical, time invariant domain of ontological knowledge, reasoning upon rapidly changing information has been neglected or forgotten. On the contrary, processing of data streams has been largely investigated and specialized Stream Database Management Systems exist. In this paper, by coupling reasoners with powerful, reactive, throughput-efficient stream management systems, we introduce the concept of Stream Reasoning. We expect future realization of such concept to have high impact on the future Internet because it enables reasoning in real time, at a throughput and with a reactivity not obtained in previous works.

**Keywords:** Data Streams, Reasoning, Real-time, Throughput-efficiency, Urban Computing, Pervasive Computing

## 1 Introduction and Motivation

Semantics is more and more evoked as a powerful tool to facilitate interoperability, flexibility and adaptability. The growing scalability of reasoning techniques [1] is key to the relevant role that semantics will play in the future Internet. While reasoners scale up in the classical, static domain of ontological knowledge, reasoning upon rapidly changing information has been neglected or forgotten.

Data streams are unbounded sequences of time-varying data elements; they occur in a variety of modern applications, such as network monitoring, traffic engineering, sensor networks, RFID tags applications, telecom call records, financial applications, Web logs, click-streams, etc. Processing of data streams has been largely investigated in the last decade [2], specialized Data Stream Management Systems (DSMSs) have been developed, and features of DSMSs are becoming supported by major database products, such as Oracle and DB2.

The combination of reasoning techniques with data streams gives rise to **Stream Reasoning**, an unexplored, yet high impact, research area. To understand the potential impact of Stream Reasoning, we can consider the emblematic case of Urban Computing [3–6] (i.e., the application of pervasive computing to urban environments). The very nature of Urban Computing can be explained by means of data streams, representing real objects that are monitored at given locations: cars, trains, crowds, ambulances, parking spaces, and so on. Reasoning

---

2 E. Della Valle, S. Ceri, D. Barbieri, D. Braga and A. Campi

about such streams can be very effective in reducing costs: for instance, looking for parking lots in large cities may cost up to 40% of the daily fuel consumption. Problems dramatically increase when big events, involving lots of people, take place; a typical Urban Computing problem is to help citizens willing to participate to such events in finding a parking lot and reaching the event locations in time, while globally limiting the occurrences of traffic congestions.

Some years ago, due to the lack of data, solving Urban Computing problems looked like a Sci-Fi idea. Nowadays, a large amount of the required information can be made available on the Internet at almost no cost: computerized systems contain maps with the commercial activities and meeting places (e.g., Google Earth), events scheduled in the city and their locations, positions and speed information of public transportation vehicles and of mobile phone users [5], parking availabilities in specific parking areas, and so on. However, current technologies are not up to the challenge of solving Urban Computing problems: this requires combining a huge amount of static knowledge about the city (i.e., urbanistic, social and cultural knowledge) with an even larger set of data streams (originating in real time from heterogeneous and noisy data sources) and reasoning above the resulting time-varying knowledge.

A new generation of reasoners is clearly needed in order to simultaneously instruct the car GPS of numerous citizens with the fastest route to the most convenient parking lot during exceptional events. Time constraints for such a reasoner are very demanding (i.e., few ms per query) because citizens are continuously making driving decisions and the traffic keeps evolving; therefore, continuous inference is required. In this work, we define Stream Reasoning as a new paradigm, based upon the state of the art in DSMS and reasoning, in order to enable such applications. By coupling reasoners with powerful, reactive, throughput-efficient stream management systems, we expect to enable reasoning in real time, at a throughput and with a reactivity not obtained in previous works.

In the rest of the paper, we identify the problem we want to untangle (Section 2). We introduce a Conceptual Architecture for Stream Reasoning (Section 3) that instantiates the pluggable algorithmic framework proposed in the LarKC project [7, 8]. We present two stream reasoning frameworks based on such architecture, one representing an evolutionary approach that combines existing solutions (Section 4), the other representing a revolutionary approach that proposes a new reasoning paradigm (Section 5). We conclude the paper discussing the challenges we are facing while planning our future work (Section 6).

## 2 Problem Definition

Our attempt to combine data stream and reasoning technologies starts from terminology. Database (DB) and Knowledge Engineering (KE) communities often use different terms to indicate the same concepts. DB community distinguishes among schema and data, whereas KE community distinguishes among factual, terminological, and nomological knowledge. The notion of data is close to the notion of factual knowledge, and similarly the notion of schema is close to the

notion of terminological knowledge. Nomological knowledge is information about rules defining actions and action-types and governing means-ends relationships in a given culture or society (e.g., when it rains, traffic gets slower); this notion is somehow captured by constraint languages for DBs, but it is mainly peculiar of KE. For the purpose of this paper, we name “*knowledge*” both terminological and nomological knowledge (thus we include in term knowledge the DB notion of schema) and we use “*data*” as a synonymous of factual knowledge.

Knowledge and data can change over time. For instance, in Urban Computing, names of streets, landmarks, kinds of events, etc. change very slowly, whereas the number of cars that go through a traffic detector in five minutes changes very fast. In order to classify knowledge and data according to the frequency of their changes we first need to introduce the notion of “*observation period*”, defined as the period when we the system is subject to querying. In the context of this paper, we consider knowledge as **invariable during the observation period**; only data can change. Of course, knowledge is subject to change, but then the mutating part of the system is not object of observation. This is not surprising: in the DB context, change of schemas occur by means of create or alter table command; while, for instance, the alter table command is executed all query processing relative to that table is suspended.

Examples of invariable knowledge, in the case of Urban Computing, include obvious terminological knowledge (such as an address is made up by a street name, a civic number, a city name, and a ZIP code), which defines the *conceptual schema* of the application, and less obvious nomological knowledge that describes how the world is expected to be (e.g., given traffic lights are switched off or certain streets are closed during the night) or to evolve (e.g., traffic jams appears more often when it rains or when important sport events take place).

Data can be further classified according with the frequency they are expected to change.

1. **Invariable** data: data that do not change in the observation period, e.g. the names and lengths of the roads.
2. **Periodically changing** data, for which a temporal law describing their evolution is present in the invariable knowledge. We can distinguish:
  - (a) *Probabilistic* data, e.g. the fact that a traffic jam is present in the west side of Milan due to bad weather or due to a soccer match is taking place in San Siro stadium;
  - (b) *Pure periodic* data, e.g. the fact that every night at 10pm Milan west-side overpass road closes.
3. **Event driven changing** data that got updated as a consequence of some external event not described in the knowledge, which are further characterized by the mean time between changes:
  - (a) *Fast*, as an example consider the intensity of traffic (as monitored by sensors) for each street in a city;
  - (b) *Medium*, as an example consider roads closed for accidents or congestion due to traffic;
  - (c) *Slow*, as an example consider roads closed for scheduled works.

4 E. Della Valle, S. Ceri, D. Barbieri, D. Braga and A. Campi

Traditional databases are suitable for capturing relatively small quantity of knowledge in their schema and huge dataset of both invariable data and event driven changing data whose mean time between changes is slow or medium. Periodically changing data can be modeled by means of triggers that perform updates; for example a trigger may update the state of a traffic light when it gets switch off for night-time.

Current reasoners are suitable for capturing large and complex knowledge, but at the cost of small datasets. Complex form of periodically changing data can be modeled by means of rules. However, reasoners cannot capture event-driven changing data whose mean time between changes is fast.

If we consider dynamic query generation, we observe that reasoners are best equipped to execute in reaction to the user's invocation, while many modern applications such as urban computing (but also network monitoring, financial analysis, sensor networks, etc.) require long-running, or continuous, queries or reasoning tasks.

Stream Database Management Systems (DSMS) represent a paradigm change in the database world because they move from persistent relations to transient streams, with the innovative assumption that streams can be *consumed* on the flight (rather than stored forever) and from user-invoked queries to *continuous queries*, i.e., queries which are persistently monitoring streams and are able to produce their answers even in the absence of invocation. DSMSs can support parallel query answering over data originating in real time and can cope with burst of data by adapting their behavior and gracefully degrading answer accuracy by introducing higher approximations.

Is combining data stream and reasoning possible? Can the innovation so far confined within the DB community be leveraged in realizing a new generation of reasoners able to cope with continuous reasoning tasks?

### 3 A Conceptual Architecture for Stream Reasoning

We are developing the Stream Reasoning vision with the LarKC European Research Project<sup>1</sup>. LarKC proposes [7, 8] a pluggable algorithmic framework which will be implemented on a distributed computational platform. The pluggable algorithmic framework ideally includes five steps to be iterated until a good enough answer [9] is found:

1. *retrieve* relevant resource/content/context;
2. *abstract* by extracting information, calculating statistics and transforming to logic,
3. *select* relevant problems/methods/data,
4. *reason* upon the aggregated knowledge, and
5. *decide* if a new iteration is needed.

In Figure 1 we present our vision in plugging data stream technologies in the LarKC framework. The top part of the figure represents the problem space

<sup>1</sup> <http://www.larkc.eu>

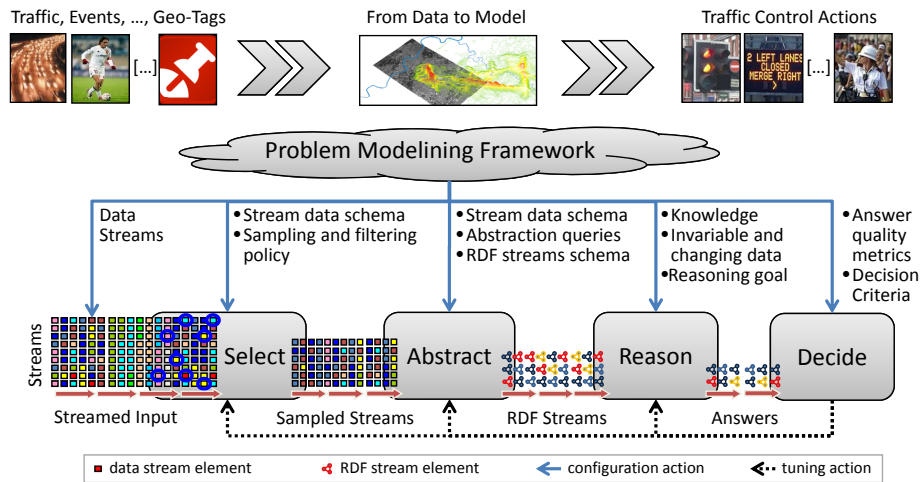


Fig. 1. Conceptual System Architecture

grounded in the Urban Computing field: data from the urban environment (e.g., traffic info, events, geo-tags, etc.) are translated in models and by reasoning on those models traffic control actions can be taken (e.g., controlling traffic lights, showing messages on traffic information panels, asking police intervention, etc.). The bottom part of the figure represents the four pluggable steps of the LarKC approach that we consider for Stream Reasoning<sup>2</sup> interconnected by the data that flow from left to right.

Data arrives to the Stream Reasoner as streamed input. A first step **selects** the relevant data in the input stream by exploiting *load-shedding* techniques [10]. Such techniques were developed to deal with bursty streams that may have unpredictable peaks during which the load may exceed available system resources. The key idea behind load-shedding is to introduce sampling policies that probabilistically drop stream elements as they are input to the selection step. Sampling and filtering policies can be either a) specified explicitly at stream-registration time, or b) inferred by gathering statistics over time, or c) by explicitly including punctuation in streams [11].

An example of sampling and filtering policy could be: if in a city a data stream originates from each traffic control camera, images should be sampled at given times rather than be continuously analyzed; in normal traffic condition, each stream could be sampled every 5 minutes, with options for increasing or decreasing the sampling rates (in congestion condition sample every 2 minutes, at night sample every 10 minutes).

The sampled streams resulting from the selection step are fed into a second step that **abstracts** from fine grain data streams into aggregated events. Such abstraction step can be done either by exploiting data compression techniques or

<sup>2</sup> We are explicitly omitting the retrieval step, because data stream retrieval should not be different from any other resource retrieval, therefore we will rely on pluggable components conceived by others.

6 E. Della Valle, S. Ceri, D. Barbieri, D. Braga and A. Campi

by aggregation queries. Data compression techniques includes the usage of histograms [12] or wave-lets [13], when the abstraction is meant to be an aggregation (e.g., counting the number of cars running through traffic detectors), and using Bloom filters [14] for duplicate elimination, set difference, or set intersection.

By **abstraction query** we mean, a continuous query that, given a large set of (possibly) unrelated low-level data in the input streams, produces an aggregated event. For instance, the abstraction step may rise a traffic congestion alert for a given street if the number of cars counted by the traffic control camera exceed 100 cars and it has been continuously increasing in the last 15 minutes.

The main proposition brought up in this paper is that, either for doing the abstraction step, or immediately after the abstraction, data streams are consolidated as **RDF streams**. RDF streams are new data formats set at the confluence of conventional data streams and of conventional atoms usually injected into reasoners. At this stage of our research, we envision two alternative formats for RDF streams:

- A **RDF molecules stream** is an unbounded bag of pairs  $\langle \rho, \tau \rangle$ , where  $\rho$  is a RDF molecule [15] and  $\tau$  is the timestamp that denotes the logical arrival time of RDF molecule  $\rho$  on the stream;
- A **RDF statements stream** is a special case of RDF molecules stream in which  $\rho$  is an RDF statement instead of an RDF molecule.

Descending from the two formats, we conceive two different stream reasoning frameworks. RDF molecule streams introduce stream reasoning as a progressive process, allowing for reuse of existing DSMS and reasoners. RDF statements stream introduce stream reasoning as a revolutionary process, requiring upon reasoners the same paradigm shift as the introduction of data streams upon databases. Section 4 and 5 describe respectively the two frameworks.

As last step, before producing the solution of the application problem of our concern (e.g., a congestion situation is monitored and traffic is rerouted according to planning activities), the answering process reaches the **decision** step. In such step quality metrics and decision criteria, defined by the application developer, are used to check if the quality of the answer is good enough and to adapt the behavior of each step (e.g., changing the sampling frequency).

## 4 RDF Molecules Stream Reasoning Framework

As we have just stated, RDF molecule streams introduce stream reasoning as a progressive process. They allow for reuse of existing DSMS and reasoners by coupling them using a transcoder and a pre-reasoner (see Figure 2).

In particular, the abstraction step can be realized using a DSMS and a transcoder. The DSMS receives the sampled data streams and generates an abstracted data stream by continuously answering the abstraction queries designed by the application developer, which typically perform an aggregation of events. The **transcoder** generates a stream element  $\langle \rho, \tau \rangle$ , where  $\rho$  is a RDF molecule

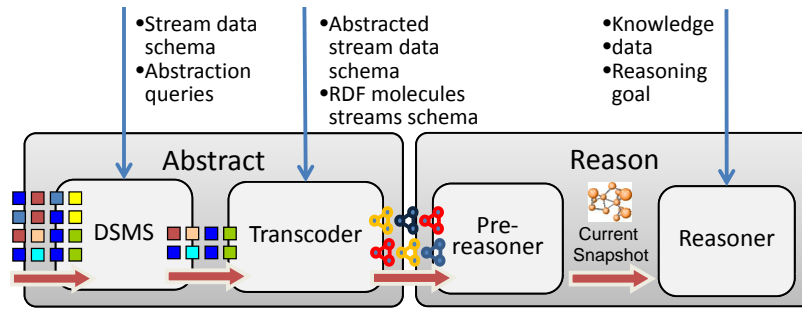


Fig. 2. RDF Molecules Stream Reasoning Framework.

and the timestamp  $\tau$  typically corresponds to the end of the aggregation interval, and puts it in the outgoing RDF stream.

We choose RDF molecules [15] as the minimum amount of information, because RDF molecules are the finest component into which an RDF graph can be decomposed without loss of information. Given that a data stream is composed by tuples and each tuple carries a minimum amount of processable information, a direct transcoding of each tuple into an RDF molecule is always possible.

For instance, in our Urban Computing example we may have a system of traffic sensors that feed a data stream by recording every sensed car across a given road. An aggregator associated with each sensor counts the number of vehicles, distinguishing them according to their type; then, the transcoder encodes this information into an RDF molecule stream element. For instance, an RDF molecule for this example is composed of four triples connected by a blank node  $\_ : x$ .

$$\left\langle \begin{array}{l} \text{http://uc.ex/tcc\#123} \quad \text{uc : measure} \quad \_ : x. \\ \_ : x \quad \text{uc : numberOfCars} \quad 120. \\ \_ : x \quad \text{uc : numberOfTrucks} \quad 70. \\ \_ : x \quad \text{uc : numberOfOtherVehicles} \quad 37. \end{array} \right\rangle, \text{Jun.17, 09 : 06 : 16AM}$$

RDF molecule streams are fed into **pre-reasoners** that perform the incremental maintenance of materialized *RDF snapshots*, i.e. RDF views describing the state of the system at a given time, which are given as input to reasoners according to application-specific strategies. Reasoners are not aware of time and produce a set of answers that remain valid until pre-reasoners produce the next snapshot. The efficient incremental materialization of RDF snapshots performed by pre-reasoners is a research challenge under investigation; background studies concern the incremental maintenance of materialized ontologies [16] and indexing of temporal XML documents [17].

## 5 RDF Statements Stream Reasoning Framework

As we anticipated in Section 3, we are also considering a more revolutionary approach, where streams are directly represented in RDF, and therefore continuous

8 E. Della Valle, S. Ceri, D. Barbieri, D. Braga and A. Campi

and/or aggregation queries can be directly expressed in RDF languages. Compared to RDF molecule streams, RDF statement streams are fine grain streams of triples. We envision the possibility to define up to eight different types of RDF statements streams depending upon the kind of information that changes at each stream input, ranging from a completely unspecified to a completely specified RDF triple. In the former case, every new element in the stream is an arbitrary RDF triple; in the latter case, every new element in the stream corresponds to the occurrence of an instance of RDF stream which is totally fixed (e.g., another unidentified vehicle seen at a given sensor). The following table summarizes the eight cases:

| Name            | Subject | Predicate | Object | Denotation |
|-----------------|---------|-----------|--------|------------|
| free            | -       | -         | -      | free       |
| bound subject   | s       | -         | -      | s          |
| bound predicate | -       | p         | -      | p          |
| bound object    | -       | -         | o      | o          |
| free subject    | -       | p         | o      | po         |
| free predicate  | s       | -         | o      | so         |
| free object     | s       | p         | -      | sp         |
| bound           | s       | p         | o      | spo        |

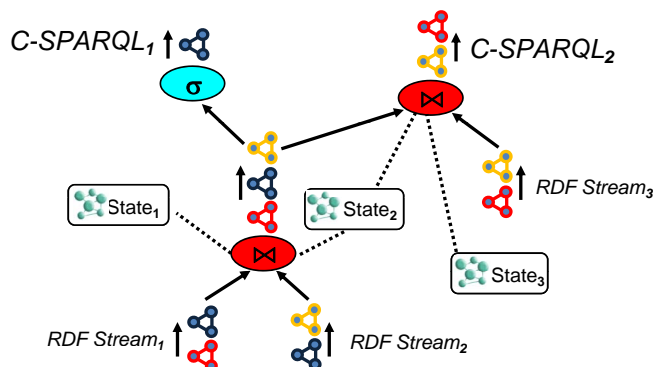
For RDF statement streams it is possible to define continuous queries both in terms of a formal abstract semantics and a concrete query language that implements the abstract semantics (namely *C-SPARQL*), following the path already explored in designing CQL [18] for DSMS.

As for CQL, the abstract semantics of such a C-SPARQL language is based on two data types, RDF statements streams (later on shortly named RDFstream) and **instantaneous RDF graphs** (later on shortly named **tgraph**). The two data types are a direct mapping of stream and relation data types in CQL. C-SPARQL queries are executed as trees of fine-grain operators performing selection and abstraction over streams; their optimization and parallelization can be approached by using techniques which are translated from DSMS systems. In Figure 3 we depict how we expect our C-SPARQL engine to share query plans among different registered queries in order to continuously answer in a throughput-efficient manner.

As for CQL, the abstract semantics of C-SPARQL includes operators of three classes:

- A *tgraph-to-tgraph* operator takes one or more tgraph as input and produces a tgraph as output.
- A *RDFstream-to-tgraph* operator takes a RDF statements stream as input and produces a tgraph as output.
- A *tgraph-to-RDFstream* operator takes a tgraph as input and produces a RDFstream as output.

RDFstream-to-tgraph operators use *sliding windows* [19] over RDF statements streams; their efficient evaluation can use the fact that stream elements



**Fig. 3.** RDF Statements Stream Reasoning Framework.

enter into windows and then exit from windows sequentially, according to the total order associated with time. RDF data is typically used in the context of ontological languages (e.g., RDF/S and OWL) enabling to describe resources and their properties. The efficient evaluation of multiple queries with several overlapping sliding windows upon both RDF data and language-specific ontological properties is a research challenge currently under investigation; methods presented in [16] can be adapted.

In this framework, scalability will be achieved by distribution and parallelism. Indeed, while each stream should be allocated to a given processor, all other operator-based computations can be distributed according to an explicit, well-defined data flow; hence, distributed database methods fully apply.

## 6 Conclusions and Future Works

In this paper we have presented some preliminary steps toward Stream Reasoning. Our main contribution is an integration architecture, taking advantage of the benefits of both data streams and reasoners, from which two stream reasoning frameworks can be derived.

The one based on RDF molecules is an evolution of the currently available solutions that relies on the possibility to couple DSMSs and state-of-the-art reasoners. This approach requires investigating an appropriate solution for incremental maintenance of time-varying RDF views and engineering throughput-efficient transcoder technology for bridging data streams to RDF Molecules Streams.

The one based on RDF statements is a revolutionary approach to reasoning that requires defining C-SPARQL semantics, studying its computational complexity, defining the concrete C-SPARQL language, and implementing a query processor that heavily exploits the intrinsic characteristic of streams.

### Acknowledgements

The work described in this paper has been partially supported by the European project LarkC (FP7-215535).

10 E. Della Valle, S. Ceri, D. Barbieri, D. Braga and A. Campi

## References

1. Kiryakov, A.: Measurable targets for scalable reasoning (2007)
2. Garofalakis, M., Gehrke, J., Rastogi, R.: *Data Stream Management: Processing High-Speed Data Streams (Data-Centric Systems and Applications)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA (2007)
3. Kindberg, T., Chalmers, M., Paulos, E.: Guest editors' introduction: Urban computing. *IEEE Pervasive Computing* **6**(3) (2007) 18–20
4. Arikawa, M., Konomi, S., Ohnishi, K.: Navitime: Supporting pedestrian navigation in the real world. *IEEE Pervasive Computing* **6**(3) (2007) 21–29
5. Reades, J., Calabrese, F., Sevtsuk, A., Ratti, C.: Cellular census: Explorations in urban data collection. *IEEE Pervasive Computing* **6**(3) (2007) 30–38
6. Bassoli, A., Brewer, J., Martin, K., Dourish, P., Mainwaring, S.: Underground aesthetics: Rethinking urban computing. *IEEE Pervasive Computing* **6**(3) (2007) 39–45
7. Fensel, D., van Harmelen, F., Andersson, B., Brennan, P., Cunningham, H., Della Valle, E., Fischer, F., Huang, Z., Kiryakov, A., il Lee, T.K., School, L., Tresp, V., Wesner, S., Witbrock, M., Zhong, N.: Towards larkc: a platform for web-scale reasoning, *IEEE International Conference on Semantic Computing (ICSC 2008)* (Aug. 2008)
8. Fensel, D., van Harmelen, F.: Unifying reasoning and search to web scale. *IEEE Internet Computing* **11**(2) (2007) 9695
9. Shvaiko, P., Giunchiglia, F., Bundy, A., Besana, P., Sierra, C., Van Harmelen, F., Zaihrayeu, I.: Benchmarking methodology for good enough answers. Technical report, DISI-08-003, Informatica e Telecomunicazioni, University of Trento (2008)
10. Tatbul, N., Çetintemel, U., Zdonik, S., Cherniack, M., Stonebraker, M.: Load shedding in a data stream manager. In: *vldb'2003: Proceedings of the 29th international conference on Very large data bases, VLDB Endowment* (2003) 309–320
11. Tatbul, N., Cetintemel, U., Zdonik, S., Cherniack, M., Stonebraker, M.: Exploiting punctuation semantics in continuous data streams. *IEEE Trans. on Knowledge and Data Eng.* **15**(3) (2003) 555–568
12. Thaper, N., Guha, S., Indyk, P., Koudas, N.: Dynamic multidimensional histograms. In: *SIGMOD '02: Proceedings of the 2002 ACM SIGMOD international conference on Management of data, New York, NY, USA, ACM* (2002) 428–439
13. Chakrabarti, K., Garofalakis, M., Rastogi, R., Shim, K.: Approximate query processing using wavelets. *The VLDB Journal* **10**(2-3) (2001) 199–223
14. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. *Commun. ACM* **13**(7) (1970) 422–426
15. Ding, L., Finin, T., Peng, Y., da Silva, P.P., McGuinness, D.L.: Tracking RDF Graph Provenance using RDF Molecules. Technical report, UMBC (April 2005)
16. Volz, R., Staab, S., Motik, B.: Incrementally maintaining materializations of ontologies stored in logic databases. *J. Data Semantics* **2** (2005) 1–34
17. Mendelzon, A.O., Rizzolo, F., Vaisman, A.: Indexing temporal xml documents. In: *VLDB '04: Proceedings of the Thirtieth international conference on Very large data bases, VLDB Endowment* (2004) 216–227
18. Babu, S., Widom, J.: Continuous queries over data streams. *SIGMOD Rec.* **30**(3) (2001) 109–120
19. Babcock, B., Babu, S., Datar, M., Motwani, R., Widom, J.: Models and issues in data stream systems. In: *PODS '02: Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, New York, NY, USA, ACM* (2002) 1–16