



The 3rd International Workshop on  
Inductive Reasoning and Machine  
Learning for the Semantic Web

**Proceedings**

|                  |                              |
|------------------|------------------------------|
| <i>edited by</i> | Claudia <b>d'Amato</b>       |
|                  | Nicola <b>Fanizzi</b>        |
|                  | Blaž <b>Fortuna</b>          |
|                  | Agnieszka <b>Ławrynowicz</b> |
|                  | Vojtěch <b>Svátek</b>        |

Heraklion, May 30, 2011

(this page is intentionally left blank)

## Foreword

Large amounts of data increasingly becoming available and described using real-life ontologies represented in Semantic Web languages, recently opened up the possibility for interesting real-world data mining applications on the Semantic Web. However, exploiting this global resource of data requires new kinds of approaches for data mining and data analysis that would be able to deal at the same time with its scale and with the complexity, expressiveness, and heterogeneity of the representation languages, leverage on availability of ontologies and explicit semantics of the resources, and account for novel assumptions (e.g., "open world") that underlie reasoning services within the Semantic Web.

The workshop tried to address the above issues, in particular focusing on the problems of how machine learning techniques, such as statistical learning methods and inductive forms of reasoning, can work directly on the richly structured Semantic Web data and exploit the Semantic Web technologies, what is the value added of machine learning methods for the Semantic Web, and what are the challenges for developers of machine learning techniques for the Semantic Web data, for example in the area of ontology mining.

The workshop was meant to bring together researchers and practitioners interested in the interdisciplinary research on the intersection of the Semantic Web with Knowledge Discovery and Machine Learning, and provide a meeting point for the related communities to stimulate collaboration and enable cross-fertilization of ideas.

The program of IRMLeS 2011 was further enriched by an invited talk from a prominent researcher. In her talk, Prof. Bettina Berendt presented Learning for a Social Web of Semantics.

The topics covered by IRMLeS 2011 included: ontology learning, and semantic tagging to expose the semantics of unstructured or semi-structured data as text, or Web 2.0 tags; management, and retrieval of Semantic Web resources, e.g. RDF data; probabilistic approaches; similarity measures for ontological data; inductive reasoning with ontologies; finally using ontologies, and other formal representations as background knowledge to steer whole knowledge discovery process.

**Acknowledgments** The workshop chairs are grateful to all the people who contributed to the event from the program committee members to the additional reviewers, the presenters and the participants. A special thank is due to the invited speaker who shared her vision on the topics of the workshop. We are grateful to the e-LICO EU FP7 Project for the sponsorship. Finally, we are grateful also to the ESWC 2011 Workshop Chairs, Program Chairs and General Chair for their constant support.

*Heraklion, May 30, 2011*

Claudia d'Amato  
Nicola Fanizzi  
Blaž Fortuna  
Agnieszka Ławrynowicz  
Vojtěch Svátek

## Program Committee Members

- Bettina **Berendt** – Katholieke Universiteit Leuven
- Stephan **Bloehdorn** - Karlsruhe Institute of Technology
- Ross D. **King** – University of Aberystwyth
- Jens **Lehmann** – University of Leipzig
- Thomas **Lukasiewicz** – Oxford University
- Matthias **Nickles** – University of Bath
- Achim **Rettinger** - Karlsruhe Institute of Technology
- Thomas **Scharrenbach** - University of Zurich
- Baris **Sertkaya** - SAP Research Dresden
- Steffen **Staab** – University of Koblenz-Landau
- Volker **Tresp** - Siemens Munich
- Joaquin **Vanschoren** - Leiden University

## Workshop Homepage

<http://irmles.di.uniba.it/2011/>

## Sponsor



e-LICO EU FP7 Project

*e-Laboratory for Interdisciplinary Collaborative Data Mining*

<http://www.e-lico.eu/>

(this page is intentionally left blank)

---

# Invited Talk Abstract

---

(this page is intentionally left blank)

# Learning for a Social Web of Semantics

Bettina Berendt

Department of Computer Science  
Katholieke Universiteit Leuven  
Celestijnenlaan 200A, B-3001 Heverlee, Belgium  
`bettina.berendt@cs.kuleuven.be`

## Abstract

It is often expected of The Semantic Web and Linked Data that the more data one combines, the closer one comes to "the truth". However, more often than not, there are "different truths", and acknowledging and dealing with them is a core challenge for interoperability. This diversity becomes highly visible in the Social Web. However, there are no generally accepted ways of representing and processing it in Linked Data or other Semantic Web forms.

A second challenge is how to use (semi-)automatic means to identify and characterize diversity, rather than relying on the hope for omniscient and/or impartial humans pointing it out. In this talk, I will describe several approaches to representing diversity of semantics, with a focus on our work on learning, displaying and uses of diversity.

Last but not least, I look forward to discussing the - technical and user-related - pros and cons of a more diversity-aware Social Semantic.

(this page is intentionally left blank)

---

# Full and Demo Papers

---

# Integrating Machine Learning in a Semantic Web Platform for Traffic Forecasting and Routing

Irene Celino<sup>1</sup>, Daniele Dell'Aglio<sup>1</sup>, Emanuele Della Valle<sup>1,2</sup>, Ralph Grothmann<sup>3</sup>, Florian Steinke<sup>3</sup>, and Volker Tresp<sup>3</sup>

<sup>1</sup> CEFRIEL - ICT Institute, Politecnico di Milano. Milano, Italy

<sup>2</sup> Dip. di Elettronica e Informazione, Politecnico di Milano, Milano, Italy

<sup>3</sup> Siemens AG, Corporate Technology, Munich, Germany

**Abstract.** Semantic Web technologies proved to be successful in integrating and interlinking data. However, traditional processing operations offered by Semantic Web technologies are limited to conceptual query answering. Growing the size of datasets, typical operations from Machine Learning or other data-intensive techniques would become important to be jointly used with Semantic Web processing.

In this paper, we explain how we exploited the capabilities of a pluggable Semantic Web application framework – the LarKC platform – to incorporate techniques from Machine Learning (recurring neural networks) and from Operational Research (routing algorithms) in a scenario of Traffic Forecasting and Routing in the Milano area. The result is a Web-based application that let a user calculate the most suitable path in the urban environment by taking into consideration the city streets characteristics and the traffic predictions.

## 1 Introduction

The geo-spatial Semantic Web is gaining attention because of the popularity of location-based services. A large amount of information is becoming available as open/linked data and applications are emerging to exploit such datasets and to apply formal reasoning.

Still some elaborations that are natural operations for other disciplines – like routing in operational research or trend forecasting in machine learning – are hardly integrated with Semantic Web techniques. Thus, it is difficult to find answers to questions like “which is the quickest way to this modern art exhibition?” or “is it possible to reach this concert tonight in less than 30 minutes if I can get into my car this afternoon at 8pm?”. Our goal is therefore to find a combination of pure semantic information (the city points of interest), geo-spatial processing (the path to the desired destination) and statistical learning (traffic forecasting), facing in the meantime challenges like data size, time-dependency, data heterogeneity or quality of data sources.

In this paper we present our solution, based on the LarKC semantic processing platform [5], to overcome the existing problems and to find useful answers to the aforementioned users' requests. While using RDF as interchange format to

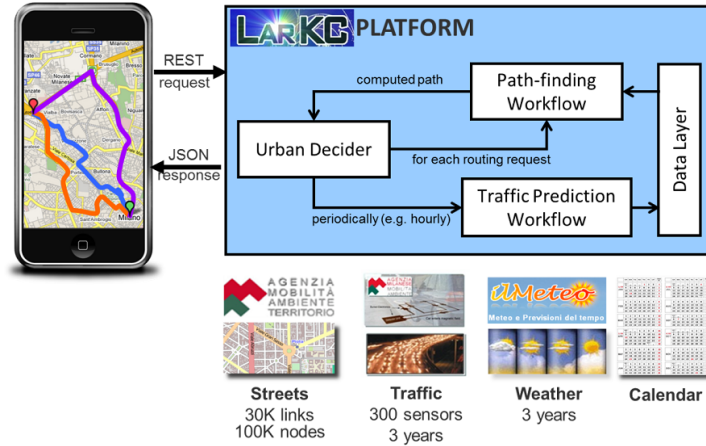
support the integration, we combined state-of-the-art statistical learning, probabilistic reasoning and operational research. The result is an application able to route users from their current position to a target point of interest within the city of Milano, taking into account both the future traffic conditions and the projected time of travel.

The description of our solution starts with an overview of the used system architecture (Section 2) and a description of the data sources (Section 3). We then describe how the traffic predictions are computed (Section 4) and how specific traffic queries can be posed and answered efficiently (Section 5). Next, we evaluate the performance and effectiveness of the framework in answering user queries (Section 6), and we finish with some concluding remarks (Section 7).

## 2 System Architecture

As mentioned in the introduction, our objective was to seamlessly combine different techniques to answer user's questions. In particular we needed to integrate routing, traffic predictions and Semantic Web querying.

For routing computation, we adopted existing operational research algorithms, in particular the Dijkstra algorithm; for traffic forecasting, we used a statistical approach based on time-delay recurrent neural network [9]. To combine those kinds of data processing together with typical Semantic Web SPARQL querying, we successfully built our solution with LarKC.



**Fig. 1.** Our LarKC workflows

LarKC [5] is a platform for massive distributed reasoning that aims to remove the scalability barriers of currently existing reasoning systems for the Semantic Web. LarKC is based on a pluggable architecture in which it is possible to exploit techniques and heuristics from diverse areas such as databases, machine learning,

cognitive science, Semantic Web, and others. LarKC also provides with built-in capabilities of parallelization and distribution of data processing, thus paving the way to easier scalability of the resulting applications.

Basing our work on the LarKC platform simplified a lot the integration of neural network algorithms (used to compute traffic predictions) with operational research methods (used in path finding algorithms) and with Semantic Web approaches (used to select the results to fulfil users' requests). Practically, we wrapped the different techniques and algorithms as "plug-ins" for the LarKC platform, then we composed those plug-ins in a set of execution workflows to combine their capabilities in the appropriate manner. Specifically we designed two execution paths (depicted in Figure 1): a first "scheduled" batch-time workflow periodically re-computes the traffic predictions for the next two hours for all streets in Milano (cf. Section 4); a second "on-demand" runtime workflow calculates the most suitable path between the starting point and the destination in the user's request (cf. Section 5); the two workflows read their inputs and write their computation results via a shared Data Layer within the platform. Finally a "Decider" component orchestrates the behaviour of the two workflows and manages the request/response interaction with the user interface.

Obtaining the same result without LarKC would have required custom integration coding and ad hoc methods to ensure the scalability or the distribution of data processing. Alternative approaches to "mash-up" of different techniques exist – like DERI Pipes [6], SPARQLScript<sup>1</sup> or SPARQLMotion<sup>2</sup> – but they mostly consist in lightweight approaches to fast prototyping with no additional support for production environments, such as LarKC support to parallelization.

### 3 Experimental Data

Our traffic routing scenario requires integrating information from several data sources, which are diverse and heterogeneous in content and also in format and availability conditions.

To address information integration, we adopted RDF as an interchange format and we linked data to existing, shared and popular ontologies. We also adopted solutions to generate virtual RDF graphs on demand, instead of performing bulk translation of data between different formats.

Similarly, we adopted SPARQL as query language, in line with the selection of RDF format. SPARQL also makes it possible to query distributed sources and to specify FROM and FROM GRAPH clauses to restrict data selection. Those choices are well founded in a Urban Computing setting like ours [3].

In the following we give details on the available datasets and we explain the use of the aforementioned technologies and languages in our scenario.

<sup>1</sup> Cf. <http://arc.semsol.org/docs/v2/sparqlscript>.

<sup>2</sup> Cf. <http://www.topquadrant.com/products/SPARQLMotion.html>.

### 3.1 Available Data and their Characteristics

The data about street topology and traffic sensors were obtained from the Municipality of Milano, Agenzia Mobilità Ambiente e Territorio (AMAT). They consist in a very detailed topology map with more than 30,000 streets (i.e. portions of roads with a specific flow direction) with 15,000 nodes (i.e. road junctions); each street portion is described with a set of both geometrical attributes (e.g. coordinates, length, number of ways, etc.) and flow-related characteristics (e.g. indicators of flow and congestion, turning prohibitions, etc.). The traffic sensors data give information about 300 sensors with their positioning and sensing capabilities; the 3 years time-series of those sensors' data records the traffic as sensed every 5 minutes intervals. As such, the sensors records sum up to more than  $10^9$  records in a 250GB database.

Additionally, for the same time-span, we complemented that information with historical weather data from the Italian website *ilMeteo.it* (CSV data with  $10^8$  records) and with calendar information (week days and week-end days, holidays, etc.) to take in consideration seasonal effects.

Those data characteristics pose the following challenges. Our historic traffic database contains more than 1 billion triples and predictions amount to 9 million new ones each day. Thus, data size is an issue when real-time predictions are required. The data is also very noisy, e.g. due to broken sensors, and does not obey a closed world assumption due to many unobserved effects, e.g. parking cars or small accidents. Moreover, traffic data is time-dependent and a prediction framework requires heterogeneous data sources, such as a street graph, historic time series of speed and flow at traffic sensors, weather data, or different calendar events like special holidays. On the query side, the routing should take into account the desired path (the shortest path vs. the fastest one, the best path on an average day vs. the best one at a specified time-date).

### 3.2 Data Representation in RDF

Reusing and linking to pre-existing ontologies and vocabularies for geo-spatial modelling, we designed and specified a dedicated ontology to represent the path-finding scenario and the outcomes of routing computation. The result is a generic Urban Path-finding Ontology<sup>3</sup>, which includes the basic concepts and relations used to model the street topology and its features, the traffic sensors, their time records and the traffic predictions.

Streets are represented by **Links** (portion of roads) and **Nodes** (junctions); **TrafficSensors** are linked to this road topology. Streets are categorized by **LinkCategory** (e.g. main road vs. secondary street) and have properties indicating *length* and reference values for *normal flow* and *congested speed*. Those properties are used to calculate *nominal* and *estimated travel time* for all streets; in case of traffic forecasting, predictions change over time, thus we employed Named Graphs [1]: for each time interval of validity, we build a *time-stamped graph* containing all the predicted travel times attached to **Links**; the graph

<sup>3</sup> Cf. <http://larkc.cefriel.it/ontologies/urbanpathfinding>.

timestamp is then used to easily identify and select the relevant graphs. The advantages of this approach are explained in [1].

## 4 Traffic Predictions

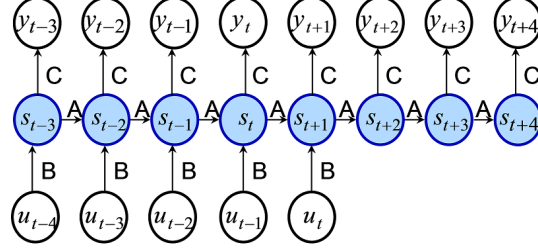
To derive a valid traffic forecast for the next four hours, we developed a LarKC plug-in that follows a statistical regression approach to traffic prediction [7]. State-of-the-art neural networks [9] are coupled with semi-supervised learning methods [11]. In contrast to simulation based methods [8] which describe each traffic participant individually, this approach does not require detailed, explicit assumptions about the behaviour of traffic participants. Instead predictions are directly determined by actually observed data. Moreover, the model complexity of a statistical approach can be adapted much easier to specific performance needs.

Our traffic prediction LarKC workflow consists of a pipeline with three steps. First, we forecast traffic speed and flow at sensor locations for the next four hours in 5 min intervals; for this task, we use the sensors traffic observations from the last 24 hours that are available through the shared platform Data Layer. We then categorize the predictions into two robust traffic conditions: normal or congested. Last, we generalize the traffic conditions from sensor locations to all streets of the road network and assign estimated travel times based on predicted traffic condition, road length and category. The results are then written back to the Data Layer for further query processing.

### 4.1 Sensor Predictions

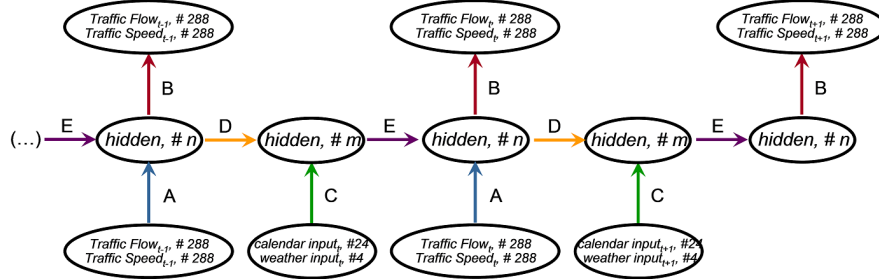
Our prediction approach is focused on the identification of underlying traffic dynamics. We assume that traffic dynamics are partially driven by an autonomous development (e.g. traffic characteristics of different day types, holidays and special calendar events) and a variety of external influences (e.g. events, construction sites or weather conditions).

We use an open, discrete-time state space model [10] with state transition equation  $s_{t+1} = f(s_t, u_t)$  and output equation  $y_t = g(s_t)$ . Identifying the dynamic system then means finding functions  $f$  and  $g$  such that the averaged quadratic distance between the observed data  $y_t^d$  and the computed data  $y_t$  of the model is minimized. We model  $f()$  and  $g()$  as parametrized functions in form of a time-delay recurrent neural network (RNN). Using weight matrices  $A$ ,  $B$  and  $C$  the model equations are  $s_{t+1} = \tanh(As_t + Bu_t)$  and  $y_t = Cs_t$ . The corresponding system identification task to determine  $A$ ,  $B$  and  $C$  is solved by “finite unfolding” in time [9], i.e. we truncate the unfolding after some time steps. The autonomous part of the RNN is extended into the future by overshooting, i.e. we iterate matrices  $A$  and  $C$  in future direction. Overshooting regularizes the learning and thus may improve the model performance. Moreover, we get as an output a whole sequence of forecasts. Figure 2 depicts the resulting spatial neural network architecture.



**Fig. 2.** Time delay recurrent neural network using unfolding in time

The RNNs to predict whole time series of speed and flow at a sensor incorporate an additional mechanism to separate time variant from time invariant structures within the traffic dynamics. We use a coordinate transformation in form of so-called *bottleneck neural networks*. The bottlenecks are embedded into the RNN and are focused on the prediction of the non-linear principal components of the traffic dynamics. In the broadest sense, the time variants of the traffic dynamics are comparable with the (non-linear) principle components of the traffic dynamics. Since the traffic dynamics can be reconstructed from the variants and invariants, we only have to forecast the variants in order to predict the development of the traffic flow and speed. The final neural network architecture is depicted in Figure 3.



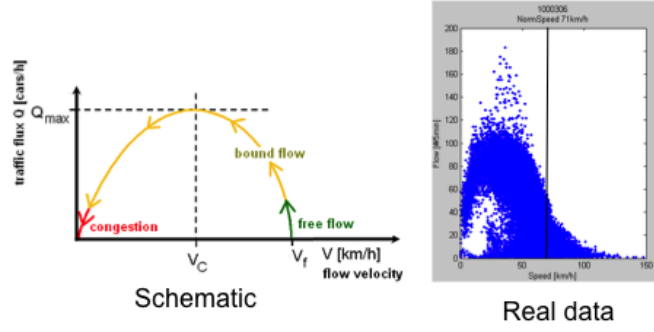
**Fig. 3.** RNN incorporating a bottleneck for the separation of time variant from invariant structures

We trained an individual neural network for each of the 317 traffic sensors having a sufficiently long, hole-free training time-series. Parameter training was performed off-line with the Siemens proprietary software SENN. The parameters were then used inside the LarKC plug-in to evaluate the RNNs on-line for each query with novel observations obtained through the Data Layer.

#### 4.2 Prediction Categorization

Speed and flow are strongly dependent on specifics of a particular link or on the sensor's field of view. We thus summarize the speed/flow predictions of the RNNs

into two more robust traffic conditions: normal or congested traffic. Motivated by the fundamental diagram of traffic flow, we threshold the predicted traffic speed at the 70%-quantile of the fastest speeds (cf. Figure 4).



**Fig. 4.** Fundamental diagram of traffic flow (left figure from Wikipedia)

#### 4.3 Network-Wide Generalization

To obtain traffic conditions for all streets and not only for sensor locations, we employ a Bayesian formulation of semi-supervised learning [11]. This is because the generalization should not only be based on Euclidean distance alone, but should take the street graph topology into account.

The assumption that neighbouring links have similar traffic conditions  $f_i$  (normal=0, congested=1) is encoded in the a priori distribution

$$p(\mathbf{f}) \propto \exp(-\sum_{ij \in E} w_{ij} (f_i - f_j)^2)$$

and traffic predictions  $y_i$  are included via the likelihood

$$p(\mathbf{y}|\mathbf{f}) \propto \prod_i \exp(-\frac{1}{\sigma^2} (f_i - y_i)^2)$$

The maximum-a-posteriori estimate can then be computed by solving a linear system and thresholding. Since all involved matrices are sparse, the linear system can be solved efficiently even for large street graphs. Nevertheless, we only apply this method for the 11.000 major streets of Milan. For minor residential streets, congestion will not normally occur.

### 5 Traffic Aware Routing

Every hour the traffic prediction are re-computed and updated in the LarKC Data Layer. Thus, whenever a user makes a request to compute a path between a starting point and a destination, the second LarKC workflow comes into play. Hereafter, we explain how such routing is expressed in a SPARQL query and how the query results are computed.

### 5.1 Querying in SPARQL

Our LarKC path-finding workflow encapsulates the operational research algorithm to compute the best path between two points on the map of Milano. The basic SPARQL query to ask for a path between two nodes is the following<sup>4</sup>:

```
SELECT DISTINCT ?path
WHERE {
  ?path a upf:Path;
  upf:hasStart <START-NODE>;
  upf:hasGoal <END-NODE>;
  upf:isComposedBy ?link;
  upf:hasPathLength ?length;
  upf:hasPathNominalTravelTime ?nominaltt;
  upf:hasPathEstimatedTravelTime ?estimatedtt;
  upf:hasPolicy ?policy.
  ?policy upf:hasMinimizedDimension <DIMENSION>.
}
```

Each policy tries to minimize a specific dimension when computing the “shortest” path; in our scenario, this dimension can assume three values: the length, the nominal travel time (traversal without traffic) or the estimated travel time (using traffic predictions). Following this modelling, we express the path computation in RDF – our interchange format – while keeping the actual processing inside the LarKC plug-in that encapsulates the Dijkstra algorithm.

### 5.2 Efficient Query Evaluation

The query illustrated above is executed at each user’s request in an “on-demand” LarKC workflow; in case of travel time estimation, the traffic predictions, in the form of estimated travel time for each link, are stored in the LarKC data layer within time-stamped Named Graphs described by their time validity; those Named Graphs are deleted and substituted by the new predictions every hour as per the workflow described in Section 4. Thus, at runtime, the relevant graphs can be selected on the basis of the “time validity” specified in the SPARQL query, thus allowing for an efficient query evaluation.

The SPARQL path-finding query is minimal in that it just contains the basic input information for the routing algorithm. Similarly to D2R [4], we are treating the path computation as a virtual RDF graph, in that the Dijkstra-based plug-in within the path-finding Workflow “hides” its algorithm behind a SPARQL-compliant interface.

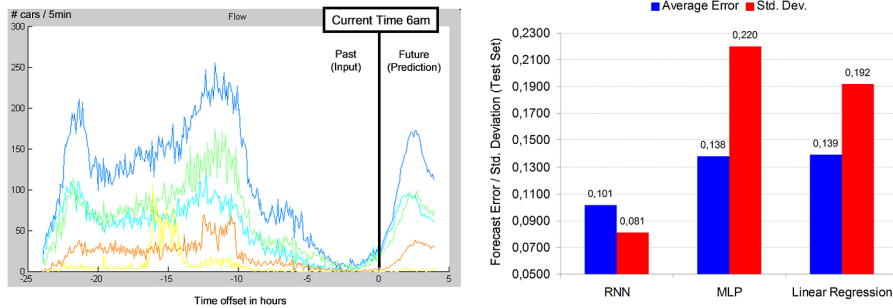
The value of our approach lies in the capability of the LarKC platform to seamlessly integrate different technologies to fulfil a specific purpose. LarKC is “semantic” in that it uses RDF as data format and lightweight data integration means, but it goes well beyond usual Semantic Web platforms in that it demonstrates its flexibility in encapsulating neural network systems for the traffic prediction and operational research routing algorithms for path finding. Without LarKC, we would have been forced to build an ad hoc system to put

<sup>4</sup> The prefix upf: refers to the cited Urban Path-finding Ontology.

together the different pieces and to make them “talk” to each other. On the contrary, we do not constrain the Dijkstra algorithm to deal with ontologies and we do not employ traditional reasoning to compute the shortest path.

## 6 Evaluation

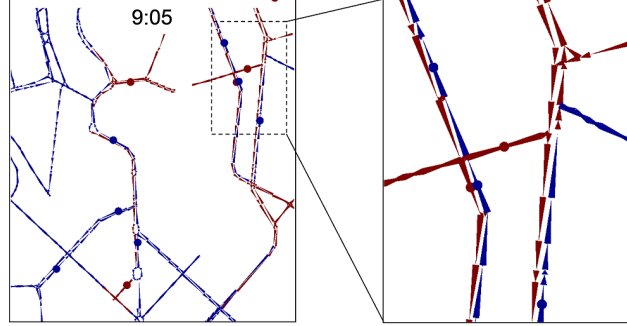
The quality of the RNN traffic forecasts is examined in Figure 5. On the left traffic flow time series for some examples sensors are shown. The past 24h of known measurements are used to predict the next four hours. A numerical evaluation against other standard regression techniques, namely a feed forward neural network and linear regression, is presented on the right. The average relative error of the time delay RNNs is significantly lower than for the competing methods, and also shows a much smaller variance.



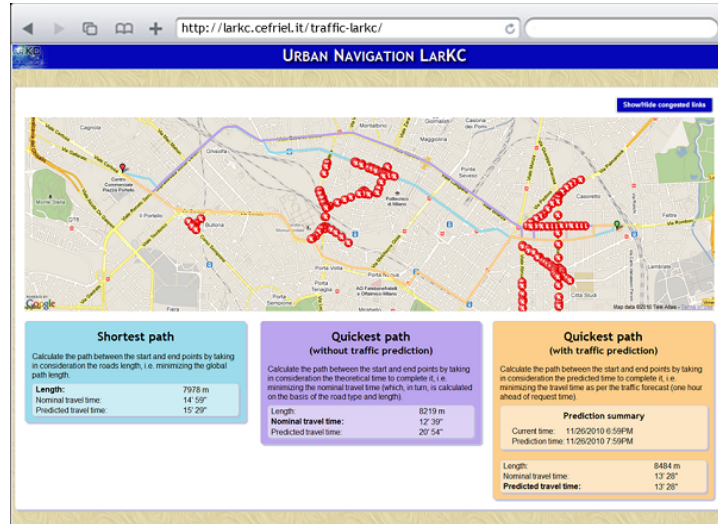
**Fig. 5.** RNN Traffic predictions: time-series with some example sensors (left), comparison of the time delay RNNs vs. feed-forward neural networks and linear regression (right)

An example of the network-wide generalization is shown in Figure 6. Numerical validation is problematic here, as no in-between-the-sensors information was available. However, the results are qualitatively plausible. They show connected areas of congestion around sensor locations with traffic distortions. Different road directions – modelled with separate links – may show different traffic situations, as common in real situations.

The Traffic LarKC application for the final user is made available as a Web application at <http://larkc.cefriel.it/traffic-larkc/>. As illustrated in Figure 7, it consists of a simple user interface where users can set start and destination points, calculate their path according to the three different policies and visualize the results together with the predicted congestion on a familiar map.



**Fig. 6.** Results of generalizing traffic condition predictions from sensor locations (dots) to all links of the road network via Bayesian Semi-Supervised Learning. Blue means normal condition, red congested.



**Fig. 7.** Screenshot of the Traffic LarKC user application

## 6.1 Performance and Scalability

Regarding computational performance, we split the discussion along the two proposed workflows.

The traffic prediction workflow is invoked once per hour and predicts the traffic for the next 4 hours on 30.000 links in 5 min intervals. This amounts to 600.000 triples that have to be updated in each run based on the newly available knowledge. While the whole runs needs 90 seconds, it should be noted that the interaction with the LarKC Data Layer – deleting old traffic predictions, reading the necessary data for the RNNs and rewriting new predictions – takes a share of

81 sec, leaving only 9 sec of the actual prediction algorithms. This may seem like a large overhead, however, the RDF representation allows for flexibly working with the traffic predictions in other plug-ins in the platform that do not have to know about the internals of the traffic prediction computation.

We evaluated the performances and scalability of the traffic aware routing by stress-testing the path-finding workflow. The test conditions were: LarKC was deployed on a six-core AMD Opteron Processor 2431 (2.4 GHz) machine<sup>5</sup>, with 8 GB RAM and Ubuntu 10.04 64 bit; the concurrent requests were issued from an Intel Core 2 (2.16 GHz) machine, with 2 GB RAM and Microsoft Windows XP Professional (SP3) in a different European location.

The results, illustrated in Figure 8 (left side), show that the response time is independent from the minimized dimension; thanks to the periodic “batch-time” re-computation of traffic predictions, the “run-time” routing is unaffected by the forecasting processing. Moreover, thanks to a smart result caching approach, the LarKC platform shows a sub-linear response-time increment with the number of concurrent requests (cf. Fig 8, right side).

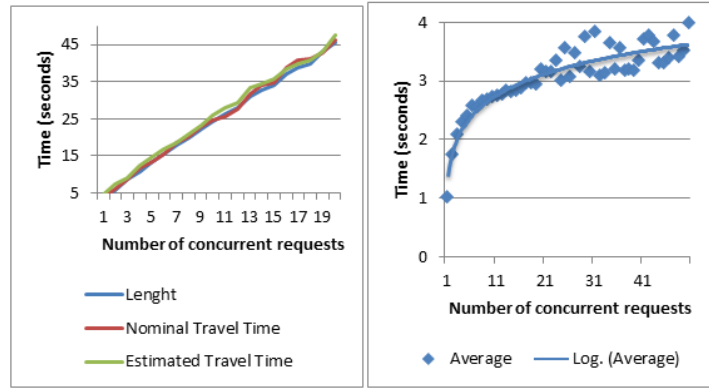


Fig. 8. Stress tests on the Traffic LarKC without caching (left) and with caching (right)

## 7 Conclusions

In this paper, we provided experimental proof that state-of-the-art statistical learning and operational research techniques can be integrated with Semantic Web query answering techniques within the LarKC platform.

The result of such integration is an efficient system that can be used to offer traffic-aware routing services. Its efficacy was only partially proved, because the information system that collects Milano traffic data was not designed to deliver

<sup>5</sup> Only one core was assigned to the LarKC virtual machine.

information in real time<sup>6</sup>, and, thus, we could not try the presented solution on real-time data. However, the experiments we conducted on historical data, the low query latency of the system and its scalability make us believe high added-value services can be delivered to the final users.

## Acknowledgments

This work was partially supported by the EU project LarKC (FP7-215535).

## References

1. D. Barbieri, E. Della Valle: “A Proposal for Publishing Data Streams as Linked Data - A Position Paper”, in Proceedings of LDOW2010, co-located with WWW2010 (2010).
2. J. Carroll, C. Bizer, P. Hayes, P. Stickler “Named Graphs”, In Journal of Web Semantics, Vol. 3, Nr. 4, p. 247-267, 2005.
3. E. Della Valle, I. Celino, D. Dell’Aglia “The Experience of Realizing a Semantic Web Urban Computing Application”, In Transactions in GIS (T. GIS) 14(2):163-181 (2010).
4. C. Bizer, R. Cyganiak “D2R-Server - Publishing Relational Databases on the Web as SPARQL-Endpoints” in Proceedings of WWW2006, Edinburgh, Scotland, May 2006.
5. D. Fensel, F. van Harmelen, B. Andersson, P. Brennan, H. Cunningham, E. Della Valle, F. Fischer, Z. Huang, A. Kiryakov, T. Kyung il Lee, L. School, V. Tresp, S. Wesner, M. Witbrock, N. Zhong “Towards LarKC: a Platform for Web-scale Reasoning”, in Proceedings of the IEEE International Conference on Semantic Computing 2008, p. 524-529, 2008.
6. D. Le Phuoc, A. Polleres, C. Morbidoni, M. Hauswirth, G. Tummarello. “Rapid semantic web mashup development through semantic web pipes”, In Proceedings of WWW2009, Madrid, Spain, April 2009.
7. S. Clark “Traffic prediction using multivariate nonparametric regression”, Journal of Transportation Engineering 129 (2), p. 161-168, 2003.
8. Q. Yang, H.N. Koutsopoulos, M.E. Ben-Akiva “Simulation laboratory for evaluating dynamic traffic management systems”, Journal of the Transportation Research Board 1710, p. 122-130, 2000.
9. H. G. Zimmermann, R. Neuneier “Neural Network Architectures for the Modeling of Dynamical Systems”, in: A Field Guide to Dynamical Recurrent Networks, Kolen, J.F.; Kremer, St. (Eds.); IEEE Press, p. 311-350, 2001.
10. S. Haykin “Neural Networks. A Comprehensive Foundation”, Macmillan College Publishing, New York, 2nd Edition, 1998.
11. O. Chapelle, B. Schölkopf and A. Zien, ed. , “Semi-Supervised Learning”, MIT Press, Cambridge, MA, 2006.

<sup>6</sup> The system that collects traffic data was designed for traffic-light tuning; data from the sensors become accessible in the information system several hours after the actual registration at the sensor.

# A Novel Metric for Information Retrieval in Semantic Networks

Joshua L. Moore<sup>1,2</sup>, Florian Steinke<sup>1</sup>, and Volker Tresp<sup>1</sup>

<sup>1</sup> Siemens AG, Corporate Technology, München, Germany

<sup>2</sup> Cornell University, Ithaca, NY

jlm@cs.cornell.edu, {Florian.Steinke, Volker.Tresp}@siemens.com

**Abstract.** We propose a novel graph metric for semantic entity-relationship networks for solving two tasks. First, given a semantic entity-relationship graph such as for example DBpedia we find relevant neighbors for a given query node. Second, we search for paths between two given nodes in order to discover interesting links between the nodes. Compared to using the default step metric our approach yields more specific and informative results, as we demonstrate for two semantic web datasets. Moreover, we show that our proposed metric can intuitively be interpreted in terms of random walks. The distances are defined via paths that maximize the log-likelihood of a restricted round trip in such a random process. This also yields a link to the commute distance, which is highly plausible for the described tasks but prohibitively expensive to compute. In comparison, our metric can be calculated efficiently using standard graph algorithms, rendering the approach feasible also for the very large graphs of the semantic web.

**Keywords:** Entity-relationship graph, information retrieval, random walk, commute distance, graph metric, path finding

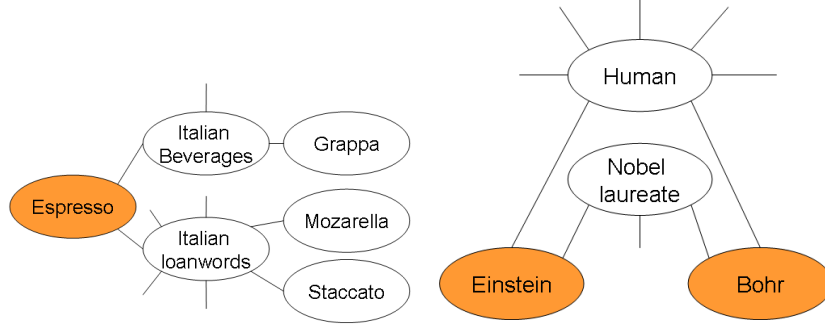
## 1 Introduction

Large entity-relationship (ER) graphs have recently become available on the semantic web. Sources like DBpedia (Auer et al., 2008), YAGO (Suchanek et al., 2007), OpenCyc<sup>3</sup> or Linked Life Data<sup>4</sup> (Momtchev et al., 2009) now encode useful information on large scale, and simple and efficient information retrieval methods for these data sources are a pressing need.

Given an ER graph, there are many interesting questions to be answered. We focus on the following two. First, given an entity node in the graph, e.g. a person or a category in DBpedia, which other nodes in the graph represent entities that are most likely to be useful in the context of the given query node? Answers might be other concepts that could be used to refine or extend an interactive search session. As another problem, consider selecting two nodes from the graph and asking how they are related. For example, via which people or concepts are Albert Einstein and Niels Bohr related in DBpedia? In the field of bioinformatics such

<sup>3</sup> <http://www.cyc.com/opencyc>

<sup>4</sup> <http://linkedlifedata.com/>



**Fig. 1.** Stylized examples of the two proposed tasks (data taken from DBpedia) and the associated challenges. In both examples there are links of equal length which are of very different informative value. Paths through nodes with many links are often too broad to be informative. Query nodes are marked orange, black lines denote links to nodes which are not depicted.

a link query between genes and diseases might well be used to discover unknown pathways from the existing literature (Antezana et al., 2009), given that such knowledge is extracted into semantic graph form as is done, for example, in Bundschuh et al. (2008).

The two described tasks can both be solved with shortest-path search on the ER graph. Most simply, one assumes a step metric, i.e. one assigns every edge in the graph a unit cost, and computes the shortest-path distance from the query node to all other nodes for the first task, e.g. using Dijkstra’s algorithm (Dijkstra, 1959), or the  $k$ -shortest paths connecting the two given nodes for the second task, e.g. by using the  $k$ -shortest path algorithm of (Yen, 1971). While this is straightforward and efficient to implement, it often returns highly irrelevant results. Consider for example a graph which contains a vertex which is connected to nearly all other vertices, such as a broad category that encompasses most entities in the graph. In the first problem we defined, this method would return the high-degree vertex with high rankings for almost any vertex queried. This lack of discrimination hinders context search, since a broad topic would lack specific relevance to any one node in the graph. Also, most of the vertices in the graph would be returned with distance two, although many of them are unrelated to the query. In the second problem, consider a database in which every person is connected to a “human” category node. In this case, the fact that Einstein and Bohr are both humans is less informative than the fact that they are both Nobel laureates, but the “human” and “Nobel laureates” nodes would have equal ranking. These problems are schematically depicted in Figure 1.

One way to allow for differently weighted nodes is to transfer PageRank-like concepts (Brin and Page, 1998) to ER graphs. However, a simple implementation of this might have an adverse effect in our setting: nodes are deemed popular and thus important if many links point to them. Thus high-level, highly connected nodes would become more important, although they are often uninformative as

argued above. A different popularity-based ranking concept assumes that facts that have many witnesses in a corpus are highly informative (Kasneci et al., 2008). This, however, requires additional input apart from the graph, and the assumption that important facts are expressed more often than others may not hold in curated data stores like wikipedia or company networks.

Another approach based on the graph structure alone is to use properties of random walks on the graph. It has been argued that the commute time between nodes in an ER graph is a useful distance measure to find relevant neighboring nodes (Baluja et al., 2008). In each step, the random walk jumps from one node to any neighboring node. The commute time is the expected number of steps for a random walk starting from one node to reach another before returning to the first. Using this metric, the problems with the step distance are alleviated in that the commute distance decreases not only if there is a short path between two nodes, but also if there are several short paths between them. Thus, a single link over a high-degree hub is not likely to yield a small distance. Moreover, if two nodes are joined by a path containing a high-degree node, a random walk is likely to “get lost” at the high-degree node, taking steps into unrelated regions of the graph, increasing the commute time between the two nodes.

While these are strong intuitive arguments for the commute distance, it is very difficult to compute. There exists an analytic formula for the commute distance in terms of the pseudoinverse of the graph Laplacian (Klein and Randić, 1993). This, however, is computationally prohibitive for the large graphs encountered in the semantic web. The pseudoinverse of the sparse graph Laplacian matrix is in general not sparse, and a square matrix of size of the number of nodes in the graph can typically not be stored or worked with efficiently. More efficient approximations of the commute distance have been developed for citation graphs in Sarkar et al. (2008). Their method, however, still needs 4 seconds for a graph of 600k nodes, which is only a moderate size in the context of the semantic web. Moreover, it is not clear how it would perform on structurally more complex graphs such as DBpedia.

In order to combine the simplicity and speed of shortest path finding with the properties of the commute distance, we propose our novel approach. We also perform shortest path finding, but with a problem adapted graph metric that assigns each edge a weight dependent on the degrees of its endpoints. Finding shortest paths in our novel metric can then be interpreted in terms of maximizing the log-likelihood of the path between the two vertices in a random walk on the graph. It can be seen as an optimally adapted first order approximation to the commute distance, and thus experimentally inherits many of the favorable properties of the commute distance. At the same time the computations are very efficient since they reduce to purely local shortest path searches that can be performed with standard graph algorithms.

In the next section we describe exactly the approach we propose to solve the two proposed tasks. In Section 3 we explain its justification in terms of random walks, and how it can be seen as an approximation to the commute distance. In Section 4 we show a number of examples and a numeric evaluation on several semantic datasets, demonstrating the superior behavior both in comparison to

the step distance path finding approach and to another simple approximation of the commute distance.

## 2 Proposed Approach

Let the semantic ER graph be represented as  $G = (V, E)$  where  $V$  is the set of nodes or entities and  $E$  the set of edges or relations holding between the entities. We do not distinguish between different relation types for the edges, opting to treat them all equally. Moreover, we also symmetrize the graphs, since the semantic direction of a relation statement is often not syntactically obvious; for example, “buys” or “is bought by” might both appear in a graph.

For each edge  $(u, v) \in E$ , we then define a weight

$$w_{(u,v)} = \log(\deg(u)) + \log(\deg(v)),$$

where  $\deg(u)$  is the degree of the node  $u$ . If  $G$  is connected, then the degree  $\deg(u)$  of all nodes  $u$  is greater than or equal to one and thus  $w_{(u,v)} \geq 0$  for all  $(u, v) \in E$ . The weights are therefore a valid positive semi-definite path metric on  $G$ , and the two described tasks can be solved using these novel edge weights in standard shortest paths routines.

In our first task, a node is specified as input, and we must retrieve a set of other nodes that are ranked based on how relevant and related they are to the query node. The results of this search might include, for example, topics that are contained within the query topic, topics which contain the query node, or topics that are related by common membership within a category or broader topic. In order to solve this task, we find the shortest path between the query node and all other nodes and rank the results. Note that using Dijkstra’s algorithm allows to directly retrieve the top ranked nodes without computing the shortest path to all other nodes.

In our second task, we are given two distinct nodes as input and wish to find paths between both that, ideally, provide unique insight into the relationship between the two nodes. This might include interesting or distinct ways that the two nodes are related. We solve this by finding the  $k$  shortest paths between the two nodes in the weighted graph, where  $k$  is a free parameter. We return the sequence of nodes in each of the  $k$  shortest paths.

The proposed metric can be justified intuitively: The distance to high-degree nodes which carry potentially very unspecific information, e.g. the “human” node, is large. This problem can be avoided as long as more specific, low degree nodes are within reach. This means that we are effectively searching in compactly connected, local subgraphs, assumed to carry context specific information.

As we will see in the experiments section, the proposed approach yields matches for queried nodes that are highly specific in subject matter and are very appropriate for someone who, for example, wants to explore a particular academic subject in detail. In addition, our metric facilitates the discovery of novel, distinct relations between nodes: vertices that are related to each other in some unique way (i.e. there is a path between them that is connected to relatively few other vertices outside of that path) are closer to each other than vertices that are linked by a very common relationship.

These intuitions can be further motivated by relating our approach with random walks on semantic graphs. Before we do so in the next section, note that the proposed approach only requires standard graph algorithms and is thus simple to implement. It also runs very efficiently even for large graphs. For example with Dijkstra’s algorithm, we only have to visit  $k$  nodes to find the  $k$  closest neighbors.

### 3 The Connection to Random Walks

Consider a random walk on  $G$ . In each step the walk moves from one vertex  $v$  to an arbitrary adjacent node with probability  $\deg(v)^{-1}$ , where  $\deg(v)$  is the degree of  $v$ . Denote the set of paths between two fixed vertices  $u$  and  $v$  by  $\Pi_{u,v}$ , i.e.  $\pi = (\pi_1, \pi_2, \dots, \pi_{\text{len}(\pi)}) \in \Pi_{u,v}$  iff  $\pi_1 = u$  and  $\pi_{\text{len}(\pi)} = v$ . The probability of the random walk following such a path and returning on the same route then is

$$\begin{aligned} p(\pi) &= \left( \prod_{i=1}^{n-1} \deg(\pi_i)^{-1} \right) \left( \prod_{i=2}^n \deg(\pi_i)^{-1} \right) \\ &= \deg(\pi_1)^{-1} \prod_{i=2}^{n-1} \deg(\pi_i)^{-2} \deg(\pi_n)^{-1}. \end{aligned}$$

The negative log-likelihood follows as

$$-\log p(\pi) = \log \deg(\pi_1) + 2 \sum_{i=2}^{n-1} \log \deg(\pi_i) + \log \deg(\pi_n) = \sum_{i=1}^{n-1} w_{(\pi_i, \pi_{i+1})}.$$

This result shows that the negative log-likelihood of the path is exactly equal to the path length in our proposed metric. Moreover, shortest path finding between  $u$  and  $v$  using this metric is thus equivalent to finding that path in  $\Pi_{u,v}$  that has minimal negative log-likelihood, or maximal probability, of a random walk following that path back and forth.

As argued in the introduction, random walks are very well suited to the proposed information retrieval task. At high-degree nodes, there are many possible nodes that could be visited next and the random walk might get lost in graph regions that often are relatively unrelated to the user’s information needs. In contrast, on local, compactly connected subgraphs for a given topic, the random walk is highly likely to return quickly.

#### 3.1 Approximation of the Commute Distance

Random walk probabilities also determine the commute time which has been proposed as an information metric on ER graphs before (Baluja et al., 2008). In contrast to our approach, the commute distance does not only measure whether there is a single high-probability connection between two nodes, but also takes into account how many such paths there are.

Since the commute distance uses more of the structure of the graph, it is potentially more robust. However, this comes at a huge computational cost and our approach is in comparison extremely efficient. Still, it can be seen as a first order approximation of the commute distance, as we will discuss now.

The commute time  $C(u, v)$  between nodes  $u$  and  $v$  is

$$C(u, v) = \sum_{\pi: (\pi_1=u, \dots, \pi_k=v, \dots, \pi_{len(\pi)}=v)} len(\pi) p(\pi) = \sum_{\pi} len(\pi) \prod_{i=1}^{len(\pi)-1} deg(\pi_i)^{-1}.$$

The sum goes over all paths that start and end at  $u$  and visit  $v$  in between. Since all terms are positive, a first order lower bound is to take into account only a single such path  $\hat{\pi}$ , i.e.

$$C(u, v) \geq len(\hat{\pi}) p(\hat{\pi}).$$

Whether this is a tight bound depends on how concentrated the path probabilities are on a single term. While there are certainly situations where this is not the case, we would argue that for many semantic graphs the approximation might be acceptable. The reason is that the degree of the nodes enters multiplicatively into the sum. Consider query nodes that are both members of two categories of highly different sizes. Then the path through the smaller category and back on the same way is actually quadratically preferred over the one through the larger category.

Given above lower bound we now try to find the optimal lower bound for the commute distance  $C(u, v)$ . That is we search for that path  $\hat{\pi}$  that contributes the most to the sum above. This then leads to

$$\max_{\pi} len(\pi) p(\pi) = \min_{\hat{\pi}} -\log len(\pi) + \sum_{i=1}^{n-1} \log deg(\hat{\pi}_i).$$

The second term is additive in the length of the path and quickly dominates the first term whose magnitude increases sub-linearly. At the same time, for paths of equal length only the second term has to be considered for the minimization. Without too large an error we can thus neglect the first term in most cases. Moreover, we restrict the optimization set to those paths that go from  $u$  to  $v$  and return the same way. The result will still be a lower bound on the commute distance, and it allows us rewrite the problem using our proposed metric as

$$\min_{\hat{\pi} \in \Pi_{u,v}} 2 \sum_{i=1}^{n-1} w_{(\hat{\pi}_i, \hat{\pi}_{i+1})}.$$

This is equivalent to our proposed approach up to a constant factor. We can thus interpret our approach as (approximately) finding an optimal lower bound to the commute distance, with the advantage that it can be computed very efficiently and with simple standard graph algorithms.

This derivation has involved a number of approximation steps that are not necessarily the tightest ones possible, see the review of Lovász (1993) for other

approximations. Yet, this argument still gives some intuition why minimizing our proposed objective might be sensible and it allows to derive a computationally very advantageous algorithm.

## 4 Experiments

To demonstrate our methods we use two large, real world semantic ER graphs, namely DBpedia and OpenCyc.

For the DBpedia dataset, we combine the category (skos) and the article-category data files. From this we create an unweighted, undirected graph neglecting the different relationship types and directions. We ignore literals since they do not add information to the graph structure. Furthermore, we discard the “Concept” node to which each category is connected. Although this is exactly the type of high-degree node that disadvantages the naïve step distance against our method, we leave it out in the interest of a fair comparison. For the step distance one could assume this master node to be removed, but as we will show below, there are still many other high-level nodes that cause similar problems. Removing these is not as trivial and could lead to unwanted effects on the search results.

We similarly define the graph for the OpenCyc dataset. An overview of the properties of both graphs is given in Table 1.

|                | DBPedia   | OpenCyc |
|----------------|-----------|---------|
| Vertices       | 3,660,898 | 150,088 |
| Edges          | 8,947,631 | 554,762 |
| Average degree | 4.88      | 7.39    |

**Table 1.** Basic statistics of the used datasets.

As baseline methods for our comparisons we use the following two approaches: First, we compare our method to using shortest paths with the step distance. Second, we compute a simple approximation of the commute distance.

The exact computation of the commute distance on the full graph is intractable, since it requires the graph Laplacian’s pseudoinverse, a matrix that for most graphs is too big to even be stored. Instead, we assume here that the commute distance is moderately local. For each query, we extract the 1000 closest nodes to the query node – in step distance – and only use the subgraph spanned by these nodes and the edges between them to compute the commute distance using the analytic formula of Klein and Randić (1993). If the subgraph has only few edges leaving it, the approximation is fairly reasonable. However, if a very unspecific node with many neighbors is among the closest nodes to the query vertex, then it will connect almost any node in the graph to the query by a path of, say, length 2. In this case the selection of the 1000 closest neighbor nodes is arbitrary and not much can be expected from our approximation of the commute distance. The baseline should thus not be regarded as a completely accurate representative of the true commute distance.

| Step                            | Our approach           | Approx. Commute                     |
|---------------------------------|------------------------|-------------------------------------|
| Espresso                        | 0 Espresso             | 0 Espresso                          |
| (C)Italian beverages            | 1 (C)Italian beverages | 4.5 (C)Italian loanwords            |
| (C)Italian loanwords            | 1 (C)Coffee beverages  | 5.05 (C)Coffee beverages            |
| (C)Coffee beverages             | 1 (C)Italian loanwords | 6.09 (C)Italian beverages           |
| (C)Italian cuisine              | 2 Bombardino           | 7.9 (C)Italian cuisine              |
| (C)Italian words and phrases    | 2 Caffè corretto       | 8.59 (C)Opera terminology           |
| (C)Italian language             | 2 Grappa               | 8.59 (C)Italian words and phrases   |
| (C)English words foreign origin | 2 Torani               | 8.59 (C)Pasta                       |
| (C)Romance loanwords            | 2 Lemonsoda            | 8.59 (C)Mediterranean cuisine       |
| (C)Beverages by region          | 2 Oransoda             | 8.59 (C)Cuisine by nationality      |
| (C)Italian alcoholic beverages  | 2 Pelmosoda            | 8.59 (C)Opera genres                |
| (C)Coffee preparation           | 2 Beverly (drink)      | 8.59 (C)Opera                       |
| Castrato                        | 2 Doppio               | 8.59 (C)Performing arts             |
| Da capo                         | 2 Caffè                | 9 (C)Musical notation               |
| Graffiti                        | 2 Chinotto             | 9 (C)European cuisine               |
| Glissando                       | 2 Ammazzacaffè         | 9 (C)Italian language               |
| Macaroni                        | 2 Stappj               | 9 Turkish coffee                    |
| Mozzarella                      | 2 Galvanina            | 9 (C)Beverages by region            |
| Opera                           | 2 Irish coffee         | 9 (C)Dried meat                     |
| Pasta                           | 2 Cortado              | 9 (C)Musical theatre                |
| Pizza                           | 2 Iced coffee          | 9 (C)Music                          |
| Spaghetti                       | 2 Pepsi Kona           | 9 (C)Articulations                  |
| Tempo                           | 2 Flat white           | 9 (C)English words foreign origin   |
| Cappuccino                      | 2 Mochasippi           | 9 (C)Singing                        |
| Legato                          | 2 Red eye (drink)      | 9 (C)Salumi                         |
| Staccato                        | 2 Liqueur coffee       | 9 (C)Croatian cuisine               |
| Operetta                        | 2 Lungo                | 9 (C)Entertainment                  |
| Cadenza                         | 2 Caffè Americano      | 9 (C)Theatrical genres              |
| Concerto                        | 2 Espresso con panna   | 9 (C)Italian culture                |
| Cantata                         | 2 Caffè breve          | 9 (C)Italian prod. protected origin |

**Table 2.** Top 30 results of neighborhood search for query node “Espresso” in DBpedia, along with the distances from the query node. First, Step means shortest path finding with the step distance; then follows our proposed approach; the last column shows the results of our simple approximation of the commute distance. Entities marked with (C) represent skos categories, other items are regular DBpedia resources.

#### 4.1 First Task: Neighborhood finding

In the following we discuss a number of example results from the two datasets. In Table 2 we list the results of a search for the query node “Espresso.” In this case, the step distance gets easily distracted by the high-degree neighbor “Italian loanwords.” As a result, the majority of the results listed are unrelated Italian terms which refer mostly to music and food. The commute distance approximation returns highly irrelevant words that are also related mostly to food and music, but this is probably due to the nature of the approximation we use – most of the 1000 nodes nearest to the espresso node are probably also due to the Italian loanwords node. Our method, on the other hand, returns a list of about one third Italian sodas and non-coffee beverages and about two thirds drinks made with espresso or at least coffee, as well as a few other types of terms.

In Table 3 we performed another search for the term “iPod.” The step distance mostly gives us various categories relating to hardware or software, and the commute distance mixes these results with a few more specific terms relating to the iPod’s function and to the related iPhone. Our method, on the other hand, yields mostly articles relating specifically to variations and functions of the iPod and the iTunes software that is integral to the use of the iPod.

We also provide results for the OpenCyc dataset, which is of a slightly different nature. It contains many rather unspecific nodes like “temporally stuff like thing” which are nice examples of how such high-degree nodes are avoided by our algorithm. In Table 4 we show the results of a search for “machine learning.” While the results of the other methods become wildly irrelevant after only the first few matches, nearly the first half of our results are still relevant to the topic at hand.

| Step                            | Our approach |                           | Approx. Commute |                                       |
|---------------------------------|--------------|---------------------------|-----------------|---------------------------------------|
| iPod                            | 0            | iPod                      | 0               | iPod                                  |
| (C)2001 introductions           | 1            | (C)iPod                   | 4.97            | (C)iTunes                             |
| (C)iPod                         | 1            | (C)Industrial designs     | 5.78            | (C)Portable media players             |
| (C)Portable media players       | 1            | (C)iTunes                 | 5.98            | (C)Digital audio players              |
| (C)iTunes                       | 1            | (C)2001 introductions     | 6.29            | (C)iPhone OS software                 |
| (C)iPhone OS software           | 1            | (C)Portable media players | 6.49            | (C)iPod                               |
| (C)Industrial designs           | 1            | (C)iPhone OS software     | 6.52            | (C)Industrial designs                 |
| (C)2001                         | 2            | iPod click wheel          | 8.15            | (C)Smartphones                        |
| (C)Apple Inc. software          | 2            | iPod Photo                | 8.84            | (C)2001 introductions                 |
| (C)Industrial design            | 2            | List of iPod models       | 8.84            | (C)Mac OS X software                  |
| (C)Windows software             | 2            | Dock Connector            | 8.84            | (C)Touchscreen portable media players |
| (C)Software by operating system | 2            | iPod Mini                 | 9.25            | (C)Consumer electronics brands        |
| (C)Apple Inc. hardware          | 2            | iPod advertising          | 9.25            | (C)Apple Inc. software                |
| (C)Windows media players        | 2            | iPhone Touch              | 9.25            | (C)iPhone                             |
| (C)Mac OS X software            | 2            | iPod Nano                 | 9.53            | (C)2007 introductions                 |
| (C)Digital audio players        | 2            | Neistat Brothers          | 9.53            | iPhone                                |
| (C)USA PATRIOT Act              | 2            | iPod Classic              | 9.53            | (C)iPhone OS                          |
| (C)MPEG                         | 2            | iPod+HP                   | 9.53            | (C)Web 2.0                            |
| (C)iPod accessories             | 2            | List of iPhone OS devices | 9.53            | (C)Windows software                   |
| (C)iPod software                | 2            | iPod Shuffle              | 9.76            | iTunes                                |
| (C)21st-century introductions   | 2            | Juicy Salif               | 9.77            | (C)Apple Inc. hardware                |
| (C)iTunes-exclusive releases    | 2            | DADVSI                    | 10.09           | (C)Software by operating system       |
| (C)iPhone OS games              | 2            | NextWorth Solutions       | 10.09           | (C)Online social networking           |
| (C)Mac OS X media players       | 2            | iMix                      | 10.17           | (C)Mac OS software                    |
| (C)Apple Inc. peripherals       | 2            | Genius (iTunes)           | 10.17           | (C)Personal digital assistants        |
| (C)Apple Inc. services          | 2            | AirTunes                  | 10.17           | (C)Brands                             |
| (C)Vehicles introduced in 2001  | 2            | iTunes law                | 10.17           | (C)Media players                      |
| (C)iPhone                       | 2            | iTunes Music Store        | 10.17           | (C)Creative Technology products       |
| (C)2001 comic debuts            | 2            | iTunes U                  | 10.17           | (C)iPod software                      |
| (C)iPhone OS                    | 2            | iTunes Applications       | 10.17           | Nimbuzz                               |

**Table 3.** Top 30 results of neighborhood search for query node “iPod” in DBpedia. Labels as in Table 2.

To demonstrate the dramatic computational advantage of our method against even the described approximation of the commute distance, we picked 1000 query nodes at random and performed a query with it using all three methods. The mean run-times on a standard desktop PC as well as the standard deviation for each method is given below

| Step          | Our Approach  | Approx. Commute |
|---------------|---------------|-----------------|
| 0.13s (0.07s) | 0.11s (0.04s) | 10.43s (9.51s)  |

The average run-time for our method was 0.11 seconds, compared to an average of 10.43 seconds for our approximation of the commute time – a difference of two orders of magnitude. As would be expected, our method runs approximately as fast as the step distance method.

It is worth noting that the time taken to approximate the commute time was extremely high in some cases. The longest time taken with the commute distance was over one minute, whereas the longest time taken with our method was only 0.66 seconds. Furthermore, one should also consider that the method we have used to calculate the commute distance is only an approximation using a graph of 1000 nodes. The most computationally intensive step required of the commute distance is the calculation of the pseudoinverse. Since this step requires cubic time to calculate, an attempt to improve the accuracy of the estimate by adding more nodes to the approximation would drastically increase the time required for computation, while an exact computation would be intractable for most practical problems.

## 4.2 Task 2: Path Finding

In this section, we present an example of our method as applied to the path finding task, showing the novel connections that our method is able to find

| Step                                 |   | Our approach                         |       | Approx. Commute                  |         |
|--------------------------------------|---|--------------------------------------|-------|----------------------------------|---------|
| machine learning                     | 0 | machine learning                     | 0     | machine learning                 | 0       |
| temporal stuff also a durative event | 1 | machine rule induction               | 2.48  | first-order collection           | 875.61  |
| computer activity                    | 1 | discriminative weight learning       | 2.89  | temp stuff also a durative event | 887.91  |
| discriminative weight learning       | 1 | generative weight learning           | 2.89  | computer activity                | 897.63  |
| generative weight learning           | 1 | MLN Generated Using Learning Type    | 3.18  | temporal stuff                   | 921.03  |
| machine rule induction               | 1 | computer activity                    | 6.27  | employee computer activity type  | 1061.05 |
| MLN Generated Using Learning Type    | 1 | markov logic network                 | 6.87  | computer activity type           | 1090.34 |
| alcoholism                           | 2 | temporal stuff also a durative event | 7.75  | athletic activity                | 1104.59 |
| burning                              | 2 | MLN Data File Pathname               | 9.86  | physical information transfer    | 1115.24 |
| flowing                              | 2 | MLN File Pathname                    | 9.86  | biological transportation        | 1138.37 |
| anthem                               | 2 | MLN Generated Using Cmd String       | 9.86  | body movement                    | 1152.19 |
| the union of ensemble showman        | 2 | MLN Rule File Pathname               | 9.86  | recreational activity            | 1169.12 |
| playing                              | 2 | MLN Type Const Dec File Pathname     | 9.86  | using a computer                 | 1181.74 |
| halt                                 | 2 | MLN Represented By Microtheory       | 10.27 | information-accessing event      | 1195.75 |
| rock climbing                        | 2 | Content Of MLN Fn                    | 10.56 | physical event                   | 1196.47 |
| snow-skiing                          | 2 | computer activity that computer did  | 11.85 | structured information source    | 1213.39 |
| Iter. Event Scene Fn id veg. 1-3 km  | 2 | computer activity that person did    | 11.85 | type of accomplishment           | 1236.74 |
| rafting                              | 2 | hack                                 | 11.85 | individual                       | 1239.97 |
| candy making                         | 2 | computer thread                      | 11.85 | computer editing                 | 1256.46 |
| composting                           | 2 | help desk session                    | 11.85 | internet activity                | 1266.56 |
| woodworking                          | 2 | network packet filtering             | 11.85 | running computer process         | 1280.32 |
| diagnosis of Wegeners granulomatosis | 2 | network packet routing               | 11.85 | locomotion event                 | 1280.92 |
| breast cancer treatment              | 2 | opening presents                     | 11.85 | ride                             | 1303.08 |
| AIDS treatment                       | 2 | packet sniffing                      | 11.85 | CW instantiating                 | 1313.32 |
| acne care                            | 2 | partitioning a disk                  | 11.85 | unnatural thing                  | 1315.36 |
| affliction procedure                 | 2 | placing a residual malicious program | 11.85 | biological process               | 1321.43 |
| allergic reaction treatment          | 2 | browser requests a secure connection | 12.13 | QA clarifying collection type    | 1338.81 |
| atrial septal aneurysm med treatment | 2 | locking computer display             | 12.13 | internet communication           | 1355.33 |
| most autistic procedure              | 2 | website maintenance                  | 12.13 | network propagation              | 1357.71 |
| vision impairment treatment          | 2 | network prop. malicious program      | 12.13 | candidate KB completeness node   | 1360.95 |

**Table 4.** Top 30 results of neighborhood search for query node “machine learning” in OpenCyc. Labels as in Table 2.

between two nodes in the graph – i.e., between two concepts in our semantic network. We also compare our method to path finding using the naïve step distance to show the advantage that our method has in discovering truly distinct and specific connections between concepts.

We compare the results of our path-finding method to those of the step distance path finding method in the case of finding connections between “computer vision” and “machine learning”, again with data from DBpedia. The resulting paths are listed in Table 5. Many of the results of our method provide insight into exactly how machine learning is used to solve specific tasks in the computer vision domain. Although insightful, some of the paths returned here by our method have significant intersection with each other. This could, however, be remedied by, for example, modifying the k-shortest paths algorithm to add extra weight to the edges equivalent to the ones traversed in previously discovered paths. Such a modification would lead to increased diversity in the results.

The step distance, on the other hand, gives us only very vague, general connections between the two subjects. The most that we learn from these results is that computer vision and machine learning are both within the subject of artificial intelligence.

Note that our method is actually able to find informative paths of significant length. While for the step distance the exponential number of possibilities for such paths quickly renders the retrieval infeasible, our method is still able to discriminate between the many choices. This might be an important advantage when applying this framework to biomedical databases, such as for example Linked Life Data. Here, one often tries to find non-obvious rather long distance interactions between different genes and diseases to discover novel pathways. Focusing on the most discriminative ones might save significant research time in this domain.

**Our Approach:**

- Path 1 (length 15.2407): Computer vision - (C)Computer vision - (C)Learning in computer vision - Machine learning
- Path 2 (length 22.1722): Computer vision - (C)Computer vision - (C)Object recognition and categorization - Boosting methods for object categorization - (C)Learning in computer vision - Machine learning
- Path 3 (length 22.4706): Computer vision - (C)Artificial intelligence - (C)Cybernetics - Machine learning
- Path 4 (length 23.5585): Computer vision - (C)Computer vision - Segmentation based object categorization - (C)Object recognition and categorization - Boosting methods for object categorization - (C)Learning in computer vision - Machine learning
- Path 5 (length 23.5585): Computer vision - (C)Computer vision - Object recognition (computer vision) - (C)Object recognition and categorization - Boosting methods for object categorization - (C)Learning in computer vision - Machine learning

**Step Distance:**

- Path 1 (length 3): Computer vision - (C)Artificial intelligence - (C)Machine learning - Machine learning
- Path 2 (length 3): Computer vision - (C)Computer vision - (C)Learning in computer vision - Machine learning
- Path 3 (length 3): Computer vision - (C)Artificial intelligence - (C)Cybernetics - Machine learning
- Path 4 (length 4): Computer vision - (C)Artificial intelligence - (C)Machine learning - (C)Learning - Machine learning
- Path 5 (length 4): Computer vision - (C)Computer vision - (C)Artificial intelligence - (C)Machine learning - Machine learning

**Table 5.** Path finding between the terms “Computer vision” and “machine learning” in DBpedia.

## 5 Conclusion

We have presented a novel metric that allows us to solve two important information retrieval task in semantic networks efficiently. The metric just depends on the degrees of adjacent nodes, and shortest path search with such a metric will thus avoid unspecific, high-degree nodes. This allows us to find interesting neighbors of a query node and novel, specific links between entities, while only using standard graph algorithms.

Often the authors themselves discovered novel, interesting information when querying the test datasets DBpedia and OpenCyc with different entities. This makes us strongly believe that the proposed approach could also be helpful to others.

A detailed user study is currently under way. From a technical point one could imagine mixing the step metric and the proposed one to obtain a tunable trade-off between the length and the distinctiveness of a path. It would also be interesting to explore ways to learn additional parameters in the metric, e.g. assigning different weights to specific edge types. Such parametric learning approaches, however, would require a benchmarking dataset which is currently not available to us. In contrast, the proposed approach is parameter free and solely dependent on intuitive arguments.

## Bibliography

- Antezana, E., Kuiper, M., and Mironov, V. (2009). Biological knowledge management: the emerging role of the Semantic Web technologies. *Briefings in bioinformatics*.
- Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., and Ives, Z. (2008). Dbpedia: A nucleus for a web of open data. In *Proceedings of 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference (ISWC+ASWC 2007)*, pages 722–735.
- Baluja, S., Seth, R., Sivakumar, D., Jing, Y., Yagnik, J., Kumar, S., Ravichandran, D., and Aly, M. (2008). Video suggestion and discovery for youtube: taking random walks through the view graph. In *Proceeding of the 17th international conference on World Wide Web*, pages 895–904. ACM.
- Brin, S. and Page, L. (1998). The anatomy of a large-scale hypertextual Web search engine. *Computer networks and ISDN systems*, 30(1-7):107–117.
- Bundschuh, M., Dejori, M., Stetter, M., Tresp, V., and Kriegel, H. (2008). Extraction of semantic biomedical relations from text using conditional random fields. *BMC bioinformatics*, 9(1):207.
- Dijkstra, E. (1959). A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271.
- Kasneci, G., Suchanek, F., Ifrim, G., Ramanath, M., and Weikum, G. (2008). Naga: Searching and ranking knowledge. In *Proc. of ICDE*, pages 1285–1288.
- Klein, D. and Randić, M. (1993). Resistance distance. *Journal of Mathematical Chemistry*, 12(1):81–95.
- Lovász, L. (1993). Random walks on graphs: A survey. *Combinatorics, Paul Erdos is Eighty*, 2(1):1–46.
- Momtchev, V., Peychev, D., Primov, T., and Georgiev, G. (2009). Expanding the pathway and interaction knowledge in linked life data. In *Proc. of International Semantic Web Challenge*.
- Sarkar, P., Moore, A., and Prakash, A. (2008). Fast incremental proximity search in large graphs. In *Proceedings of the 25th international conference on Machine learning*, pages 896–903. ACM.
- Suchanek, F. M., Kasneci, G., and Weikum, G. (2007). Yago: A Core of Semantic Knowledge. In *16th international World Wide Web conference (WWW 2007)*, New York, NY, USA. ACM Press.
- Yen, J. (1971). Finding the k shortest loopless paths in a network. *Management Science*, 17(11):712–716.

# A prototype system for semantic aggregation of SPARQL query results

Agnieszka Ławrynowicz<sup>1</sup>, Jędrzej Potoniec<sup>1</sup>, Łukasz Konieczny<sup>1</sup>, Michał Madziar<sup>1</sup>, Aleksandra Nowak<sup>1</sup>, and Krzysztof T. Pawlak<sup>1</sup>

Institute of Computing Science, Poznan University of Technology, ul. Piotrowo 2,  
60-965 Poznan, Poland

**Abstract.** We report of a prototype system, named ASPARAGUS, we develop for the purpose of performing aggregation of SPARQL query results on a semantic baseline, that is by an exploitation of the background ontology expressing the semantics of the returned results. The system implements the recent research results on semantic aggregation. This includes 'semantic clustering', where the groups are inductively formed, based on the semantic similarity of the individual resources.

## 1 Introduction

Query answering on the Web can return a large number of results that can be hardly manageable by users. Moreover, typically, only a small part of the result set is relevant to the user thus making necessary the analysis of the retrieved results to identify those relevant ones. This phenomenon is known as *information overload*. To liberate users of a tedious manual analysis job, various services have been set up, such as predefined category structures, and fully automatic search engines. A promising solution, adopted in this work, is to marry the benefits of the two mentioned approaches and generate dynamic groupings/clustering over retrieved query results.

This paper presents a prototype of the system based on the recent research results on the topic of *semantic aggregation* of Web query results. 'Semantic' stands for an approach where background knowledge is used (in the form of ontologies) during the process of aggregation. In [1, 2] a new method for grouping query results was introduced, coined *semantic grouping*. Given on input a conjunctive query, and a background OWL<sup>1</sup> ontology, the method returns a dynamic, *runtime* categorization over ranked query results. During generation of the categorization, the method exploits the *semantics* of knowledge bases of reference (in the form of ontologies) by an application of deductive reasoning. In particular, given a grouping criterion expressed as a (complex) concept from a knowledge base of reference, results are grouped according to (part of) the subsumption hierarchy deductively obtained by considering the specified concept and the given ontology. In this way, the results can be shown and navigated similarly to a faceted search.

---

<sup>1</sup> <http://www.w3.org/TR/owl-features/>

In [3], a concept and a method were proposed for *semantic clustering* of query results, that consists in an application of data mining methods to inductively build clusters of similar results. In [4] a possible extensions for SPARQL to support such type of the results aggregation was discussed. Importantly, in the 'semantic clustering' approach, the semantics of clustered objects (expressed in the background ontologies) is exploited to compute similarities or distances between the individual resources.

Let us illustrate the concepts of semantic grouping and semantic clustering of query results by means of a simple scenario, where the user searches for weekend break offers, and would like to have them grouped by a destination criterion. Classically, aggregation abilities are provided by SQL-like GROUP BY clause, which is to be supported in a new version of a standard Semantic Web query language – SPARQL<sup>2</sup>. However, the classical GROUP BY semantics, which is to partition the results by identical values, is not always suitable. In the case, when destination is represented by a town name, one group for each town name would be created by GROUP BY, which could result in too many groups to be easily managed by the user. In turn, if the destination criterion would be represented by an ontology concept *Destination*, being a root of a concept hierarchy which we would like to exploit for grouping the results, the towns would be annotated by its subconcepts such as *City*, *Resort*, *SeasideResort*, *SkiResort*, then it would be possible to aggregate the results into a smaller number of 'semantic' groups e.g. into seaside or ski resort destinations.

Let us assume further, that the user would like to have the results grouped by towns' mutual proximity. Would be then grouping by longitude and latitude what the user actually expected? Since no two towns share the same co-ordinates, one group for each town would be created, which obviously would not meet the needs of the user. Thus, by relaxing strict, boolean criteria of the GROUP BY by means of similarity measures, we may extend the framework of 'semantic aggregation' to encompass inductive, clustering abilities.

We have implemented the mechanisms of semantic aggregation of the query results, both deductive ('semantic grouping'), and inductive ('semantic clustering'), in the system named ASPARAGUS (*Automatic SPARQL query results AGgregation Using Semantics*), which is described in the next section.

## 2 Overview of the system

The architecture of ASPARAGUS is presented in Fig. 1. ASPARAGUS exploits additional syntax elements that extend SPARQL for semantic aggregation: i) **CATEGORIZE BY** clause that corresponds to the SQL **GROUP BY**, but instead of grouping the results by their exact values, they are grouped according to their membership to the ontology classes, deduced by a reasoner, and ii) **CLUSTER BY** clause whose functionality is achieved in our system by means of advanced inductive reasoning, clustering methods, that exploit semantic similarity measures to inductively build clusters of similar individuals.

<sup>2</sup> <http://www.w3.org/TR/rdf-sparql-query/>

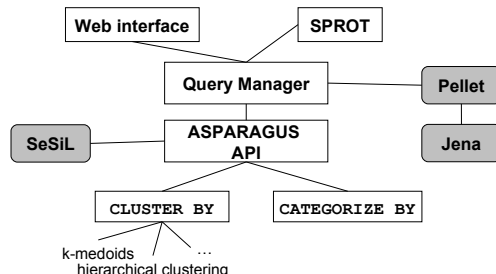


Fig. 1: The architecture of ASPARAGUS.

Central to the system is *Query Manager*, that is responsible for task scheduling, and invoking of subsequent operations. To manipulate, and reason with knowledge bases, and to answer SPARQL queries, Pellet<sup>3</sup> [5] and Jena<sup>4</sup> libraries are used. *Query Manager* schedules execution of queries submitted either through WWW interface or SPARQL endpoint. A *Query* is an object that is delivered to *Query Manager* in the form containing: a) the query full text (consisting of standard SPARQL query part, the mode of aggregation that is **CATEGORIZE BY** or **CLUSTER BY**, the list of variables to be used during aggregation, and an algorithm to be used with its parameters), b) an indication of a SPARQL endpoint to retrieve data from, and c) list of URLs or access paths of files to be read in to a default graph. Query processing consist in performing the following steps: 1) Creation of Pellet's helper objects, that contain SPARQL query, and a list of URLs to retrieve data from, 2) Generation of the query execution plan, 3) Query execution and retrieving the results, 4) Aggregation of the results with use of a previously indicated algorithm, 5) Marking the query as finished. In case of an error during execution of any of the above steps, it is recorded as an answer, and the query is marked as finished. Since ASPARAGUS works within a J2EE application server, *Query Manager* is run as a stand-alone thread that uses a multi-threaded queue.

The implemented algorithms are linked by a common programming interface of ASPARAGUS. The interface mediates also between the system and the SeSiL library, that implements a set of similarity measures. SeSiL (*Semantic Similarity Measures Library*) has been developed by the Poznan University of Technology since 2010 within EU FP7 project e-LICO, and to this end, it implements a set of (semantic) kernel functions (e.g. the ones proposed in [6, 7]) to compute the similarity between individual resources described by ontologies represented in description logics.

<sup>3</sup> <http://clarkparsia.com/pellet/>

<sup>4</sup> <http://jena.sourceforge.net/>

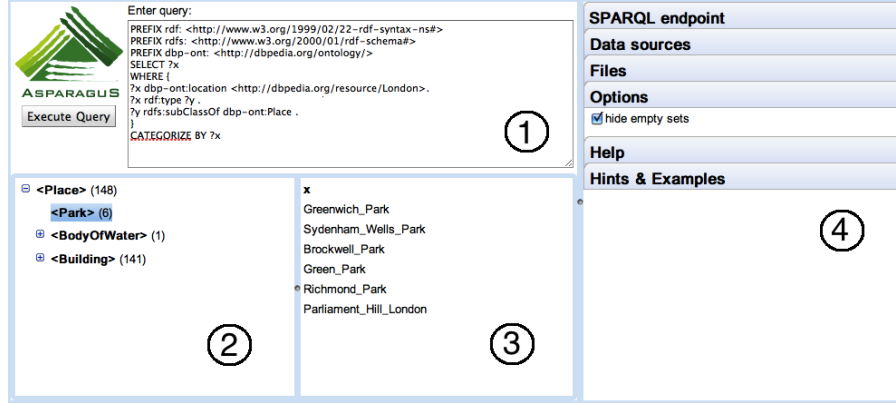


Fig. 2: A screenshot of the Web interface of ASPARAGUS.

Currently ASPARAGUS implements two clustering algorithms, a classical Agglomerative Hierarchical Clustering type algorithm, extended by an ability to produce semantic descriptions of clusters, and a variant of k-medoids algorithm proposed in [8], extended with the possibility to apply similarity measures for datatypes.

An interaction with the user may be performed in two ways: i) by a Web interface, and ii) (semi-)automatically by use of the standard SPROT interface for submitting SPARQL queries. A sample screenshot of the Web interface is shown in Fig. 2. The interface 1) enables entering a text of SPARQL query, 2) displays a dynamically generated, navigable hierarchy on top of the retrieved results, 3) the content of a group the user navigated to, and 4) enables the specification of a set of options such as: indication of a SPARQL endpoint or HTTP addresses to retrieve data from, indication of a local file to upload temporarily to the system, displaying or hiding empty groups, help, hints and examples.

### 3 Conclusions and future work

In this paper we described a prototype of the system called ASPARAGUS implementing the algorithms that involve deductive and inductive reasoning, to semantically aggregate the results of SPARQL queries. The system has a component that employs inductive reasoning techniques, such as clustering, and similarity computation. Besides that it is released with its own Web user interface, ASPARAGUS may be also integrated as a part of a more complex system, e.g. as a middleware between SPARQL query engines/endpoints and a customized user interface, specific to some knowledge base, what we plan in the future work.

Despite that the first prototype of the system is ready, we have identified a set of open challenges we currently work on towards a subsequent version of ASPARAGUS: a) suitable solutions for retrieving relevant parts of RDF datasets

from SPARQL endpoints for improving the data acquisition capabilities of the system, b) more efficient similarity measures, suitable for an on-line application (we are developing our own measures as well as plan to investigate some further approaches proposed in the literature like [9]), c) extensions of methods for generating navigable groupings of the results, and their labels, e.g. refinement operator based ones proposed in [2], or the ones assuming an individual (nominal class) as the label of a group, in this context we can for example try to adapt further different clustering algorithms proposed in the context of databases coupled with concept learning techniques, d) improvements of the coverage of datatypes of the clustering module, e) migration of parts of source code to an internal API for handling knowledge bases, and making it compatible with further APIs of popular available reasoners and semantic stores (in particular Jena API, OWL API, and Sesame API to use OWLim<sup>5</sup> as an underlying semantic store), and further work on ASPARAGUS API.

We also plan to investigate a possibility of using proposals of extensions of SPARQL towards OWL (rather than RDF), like SPARQL-DL [10]. In this respect, we will also consider convenient ways of seamlessly switching between a working mode that uses pure SPARQL endpoint (e.g. for the queries like "SELECT ?place ?longitude ?latitude ", which can be clustered by a proximity measure), and a mode that exploits reasoners such as Pellet and possibly SPARQL extensions towards OWL.

Moreover, there have been some proposals for extending SPARQL potentially relevant to be incorporated into our system like [11] and [12]. Especially the former one, proposing SPARQL-ML, is related to our work and worth investigation for our CLUSTER BY mode. However, in the current shape, it is mostly developed towards performing predictive, and not descriptive data mining tasks. In particular, [11] describes an approach where first a special type of query is executed to build a model of the data (by using a training set of data), and at the second stage, another kind of query is applied for prediction or classification tasks using the already built model. Hence it would need an investigation how to re-use the proposed framework for clustering.

**Acknowledgements.** This work is partially supported by Polish Ministry of Science and Higher Education (grant number N N516 186437) and by European Community 7th framework program ICT-2007.4.4 (grant number 231519 "e-LICO: An e-Laboratory for Interdisciplinary Collaborative Research in Data Mining and Data-Intensive Science").

## References

1. d'Amato, C., Fanizzi, N., Lawrynowicz, A.: Categorize by: Deductive aggregation of semantic web query results. In Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T., eds.: ESWC (1). Volume 6088 of Lecture Notes in Computer Science., Springer (2010) 91–105

<sup>5</sup> <http://www.ontotext.com/owlim>

2. Lawrynowicz, A., d'Amato, C., Fanizzi, N.: A refinement operator based method for semantic grouping of conjunctive query results. In Setchi, R., Jordanov, I., Howlett, R.J., Jain, L.C., eds.: KES (3). Volume 6278 of Lecture Notes in Computer Science., Springer (2010) 359–368
3. Lawrynowicz, A.: Grouping results of queries to ontological knowledge bases by conceptual clustering. In Nguyen, N.T., Kowalczyk, R., Chen, S.M., eds.: ICCCI. Volume 5796 of Lecture Notes in Computer Science., Springer (2009) 504–515
4. Lawrynowicz, A.: Query results clustering by extending SPARQL with CLUSTER BY. In Meersman, R., Herrero, P., Dillon, T.S., eds.: OTM Workshops. Volume 5872 of Lecture Notes in Computer Science., Springer (2009) 826–835
5. Sirin, E., Parsia, B., Grau, B., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. *Journal of Web Semantics* **5**(2) (2007) 51–53
6. Fanizzi, N., d'Amato, C.: A declarative kernel for  $\mathcal{ALC}$  concept descriptions. In Esposito, F., Ras, Z.W., Malerba, D., Semeraro, G., eds.: ISMIS. Volume 4203 of Lecture Notes in Computer Science., Springer (2006) 322–331
7. Bloehdorn, S., Sure, Y.: Kernel methods for mining instance data in ontologies. In Aberer, K., et al., eds.: Proceedings of the 6th International Semantic Web Conference, ISWC2007. Volume 4825 of LNCS., Springer (2007) 58–71
8. Fanizzi, N., d'Amato, C., Esposito, F.: Conceptual clustering for concept drift and novelty detection. In Bechhofer, S., et al., eds.: Proceedings of the 5th European Semantic Web Conference, ESWC2008. Volume 5021 of LNCS. Springer (2008) 318–332
9. Janowicz, K., Wilkes, M.: Sim-dla: A novel semantic similarity measure for description logics reducing inter-concept to inter-instance similarity. In Aroyo, L., Traverso, P., Ciravegna, F., Cimiano, P., Heath, T., Hyvönen, E., Mizoguchi, R., Oren, E., Sabou, M., Simperl, E.P.B., eds.: ESWC. Volume 5554 of Lecture Notes in Computer Science., Springer (2009) 353–367
10. Sirin, E., Parsia, B.: Sparql-dl: Sparql query for owl-dl. In Golbreich, C., Kalyanpur, A., Parsia, B., eds.: OWLED. Volume 258 of CEUR Workshop Proceedings., CEUR-WS.org (2007)
11. Kiefer, C., Bernstein, A., Locher, A.: Adding data mining support to sparql via statistical relational learning methods. In Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M., eds.: ESWC. Volume 5021 of Lecture Notes in Computer Science., Springer (2008) 478–492
12. Huang, Y., Tresp, V., Bundschuh, M., Rettinger, A.: Scalable relational learning for sparse and incomplete domains. In: Proceedings of the International Workshop on Statistical Relational Learning (SRL-2009). (2009)