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

Irene Celino[1], Daniele Dell'Aglio[1], Emanuele Della Valle[1,2], Ralph Grothmann[3], Florian Steinke[3], and Volker Tresp[3]

[1] CEFRIEL - ICT Institute, Politecnico di Milano. Milano, Italy
[2] Dip. di Elettronica e Informazione, Politecnico di Milano, Milano, Italy
[3] 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 answers 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.

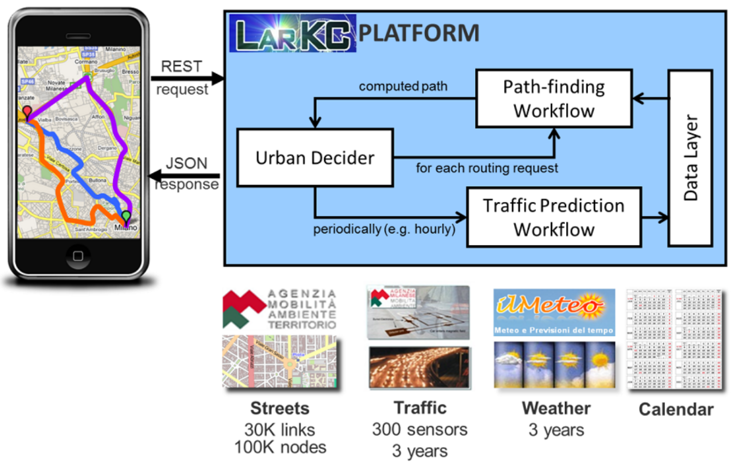


Fig. 1. Our LarKC workflows

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[1] or SPARQLMotion[2] – 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.

[1] Cf. `http://arc.semsol.org/docs/v2/sparqlscript`.

[2] 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[3], 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

[3] 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.

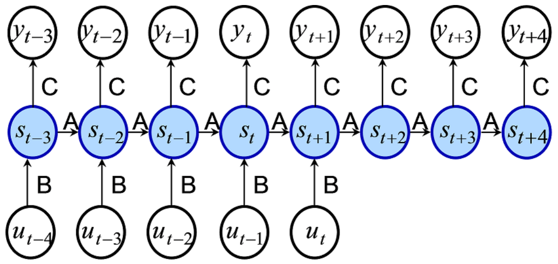


Fig. 2. Time delay recurrent neural network using unfolding in time

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.

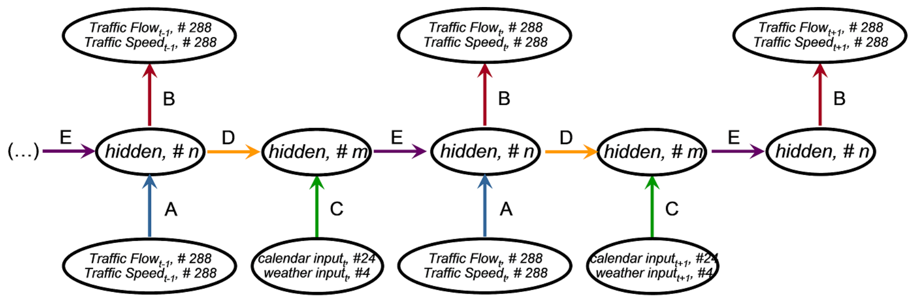

Traffic Flow$_{t-1}$, # 288
Traffic Speed$_{t-1}$, # 288
Traffic Flow$_t$, # 288
Traffic Speed$_t$, # 288
Traffic Flow$_{t+1}$, # 288
Traffic Speed$_{t+1}$, # 288
B
(…)
E
D
hidden, # n
hidden, # m
A
C
calendar input$_t$, #24
weather input$_t$, #4
calendar input$_{t+1}$, #24
weather input$_{t+1}$, #4

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).

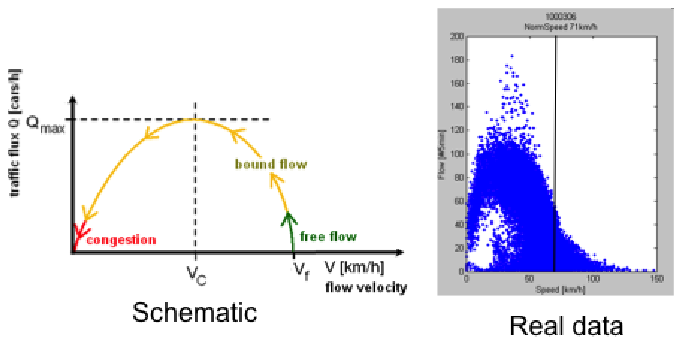


Fig. 4. Fundamental diagram of traffic flow (left figure from Wikipedia)

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

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

and traffic predictions y_i are included via the likelihood

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

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[4]:

```
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

[4] 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.

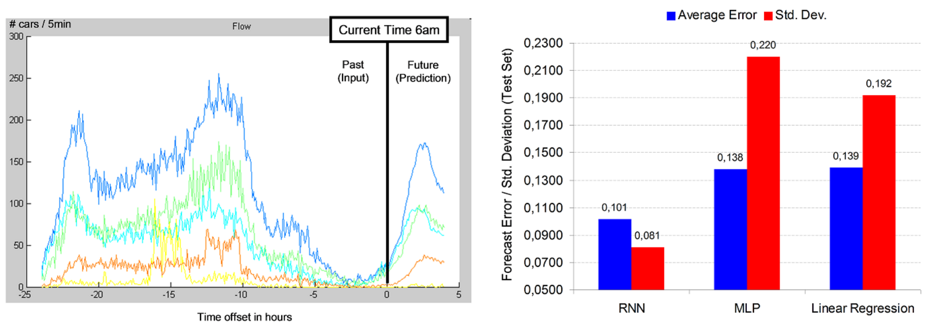


Fig. 5. RNN Traffic predictions: time-series with some example sensors (left), comparison of the time delay RNNs vs. feed-forward neural networks and linear regression (right)

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.

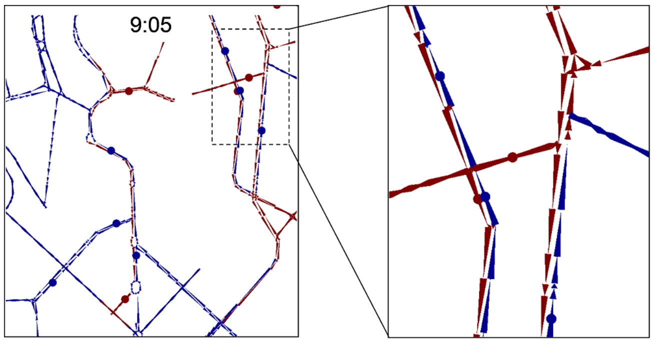
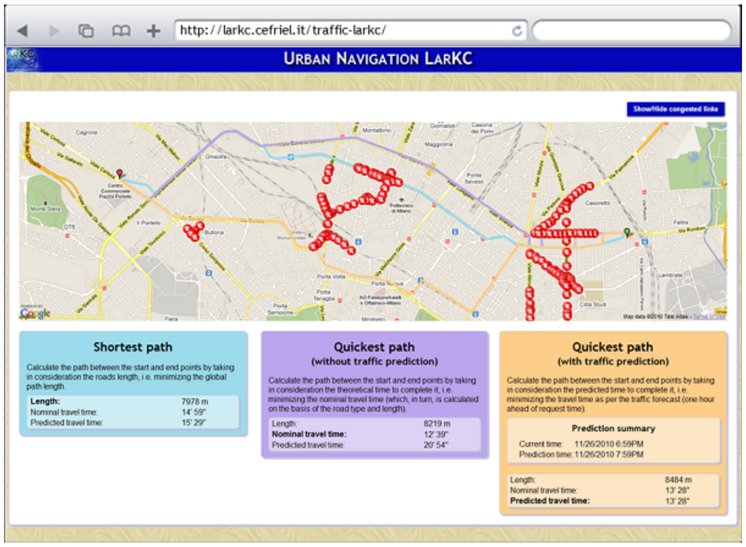


Fig. 6. Results of generalizing traffic condition predictions from sensor locations (dots) to all links of the road network via Bayesian Semi-Supervised Learning. Blue means normal condition, red congested.


http://larkc.cefriel.it/traffic-larkc/
URBAN NAVIGATION LARKC
Show/Hide congested links

Shortest path

Calculate the path between the start and end points by taking in consideration the roads length, i.e. minimizing the global path length.

Length:	7978 m
Nominal travel time:	14' 59"
Predicted travel time:	15' 29"

Quickest path
(without traffic prediction)

Calculate the path between the start and end points by taking in consideration the theoretical time to complete it, i.e. minimizing the nominal travel time (which, in turn, is calculated on the basis of the road type and length).

Length:	8219 m
Nominal travel time:	12' 39"
Predicted travel time:	20' 54"

Quickest path
(with traffic prediction)

Calculate the path between the start and end points by taking in consideration the predicted time to complete it, i.e. minimizing the travel time as per the traffic forecast (one hour ahead of request time).

Prediction summary

Current time: 11/26/2010 6:59PM
Prediction time: 11/26/2010 7:59PM

Length:	8484 m
Nominal travel time:	13' 28"
Predicted travel time:	13' 28"

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 seems 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[5], 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).

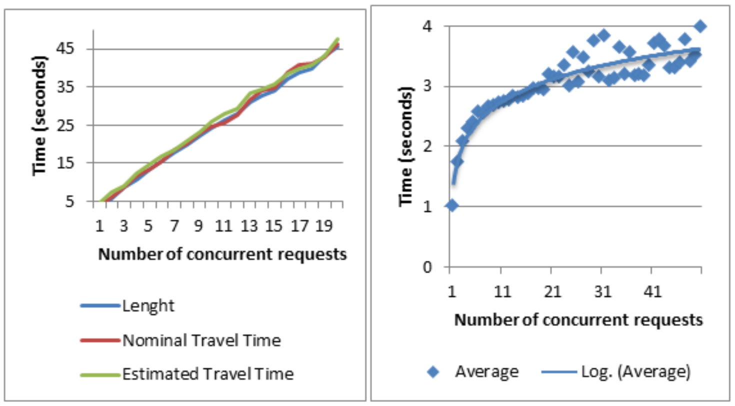


Fig. 8. Stress tests on the Traffic LarKC without caching (left) and with caching (right)

7 Conclusions

In this paper, we provided experimental proof that state-or-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

[5] Only one core was assigned to the LarKC virtual machine.

information in real time[6], 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'Aglio "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.

[6] 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.