BOTTARI: an Augmented Reality Mobile Application to deliver Personalized and Location-based Recommendations by Continuous Analysis of Social Media Streams

Marco Balduini[a], Irene Celino[b], Daniele Dell'Aglio[b], Emanuele Della Valle[a,b,*], Yi Huang[c], Tony Lee[d], Seon-Ho Kim[d], Volker Tresp[c]

[a]*Dip. Elettronica e Informazione – Politecnico di Milano, via Ponzio 34/5, 20133, Milano, Italy*
[b]*CEFRIEL – ICT Institute, Politecnico di Milano, via Fucini 2, 20133, Milano, Italy*
[c]*Siemens AG, Corporate Technology, Otto-Hahn-Ring 6, 81739 München, Germany*
[d]*Saltlux, 7F, Deokil Building, 967, Daechi-dong, Gangnam-gu, Seoul 135-848 Korea*

Abstract

In 2011, an average of three million tweets per day was posted in Seoul. Hundreds of thousands of tweets carry the live opinion of some tens of thousands of users about restaurants, bars, coffees and many other semi-public points of interest (POIs) in the city. Trusting this collective opinion to be a solid base for novel commercial and social services, we conceived BOTTARI: an augmented reality application that offers personalized and localized recommendation of POIs based on the temporally-weighted opinions of the social media community.

In this paper, we present the design of BOTTARI, the potentialities of semantic technologies like inductive and deductive stream reasoning and the lesson learnt in experimentally deploying BOTTARI in Insadong – a popular tourist area in Seoul – for which we have been collecting tweets for three years to rate the few hundreds of restaurants in the district. The results of our study show to demonstrate the feasibility of BOTTARI and encourage its commercial spreading.

Keywords: social media analysis, mobile app, personalized recommendation, location-based recommendation, stream reasoning

1. Introduction

Imagine you are a tourist in Seoul. You would like to dine out. You prefer to avoid the *tourist's traps* and dine where the locals do. You have been told that Insadong district would be the perfect place; it offers a choice of more than a hundred restaurants in two square kilometres and most of the district is reserved for pedestrians.

When you reach Insadong-gil (the main street of the district), you find yourself surrounded by hundreds of restaurant advertisements (see Figure 1). You know you can still open the guide book and choose one of the few restaurants listed there, but you definitely want a place where the locals go. You take out your mobile and check various apps that recommend restaurants based on users' reviews. The number of user rated restaurants is smaller than you expected: only ten restaurants where rated more than ten times. This is probably because you are in Seoul, one of the cities world wide where people *tweets* more[1]. You wish a service exists that continuously analyses the social media streams and that can recommend you how the locals have been rating Insadong's restaurants for the last months.

This is exactly what we designed BOTTARI for. BOTTARI is an augmented reality application for personalized and localized restaurant recommendations, experimentally deployed in the Insadong district of Seoul. At a first look, it may appear like other mobile apps that recommend restaurants, but BOTTARI is different: BOTTARI uses inductive and deductive stream reasoning [1] to continuously analyse social media streams (specifically Twitter) to understand how the social media users collectively perceive the points of interest (POIs) in a given area, e.g., Insadong's restaurants.

In this paper, we describe the choices we made in designing BOTTARI and the lessons we learned by experimentally deploying it in Insadong. The paper is organized as follows. Section 2 introduces relevant background. Section 3 illustrates BOTTARI mobile app from the user's point of view, i.e., the main task being pursued. Section 4 allow to understand the data used in experimentally deploying BOTTARI in Insadong district. Section 5 briefly illustrates the ontology at the core of BOTTARI that was used to integrate the available information. Section 6 presents BOTTARI back-end. Sections 7 and 8 report our evaluation results both in terms of quality of recommendations and scalability of BOTTARI back-end. Finally, Section 9 concludes the paper discussing the lessons we learnt and sketches our future works.

*Corresponding author
Email addresses: marco.balduini@polimi.it (Marco Balduini), irene.celino@cefriel.it (Irene Celino), daniele.dellaglio@cefriel.it (Daniele Dell'Aglio), emanuele.dellavalle@polimi.it (Emanuele Della Valle), yihuang@siemens.com (Yi Huang), tony@saltlux.com (Tony Lee), shkim@saltlux.com (Seon-Ho Kim), volker.tresp@siemens.com (Volker Tresp)

[1] An average of 3 million tweets were posted each day in Seoul in 2011

Figure 1: A picture of Insadong district: the density of restaurants is very high (cf. also Section 4).

2. Background Work

In this section, we briefly illustrate the context in which BOTTARI idea was conceived and the technological *ingredients* we used in implementing it.

2.1. How Tourists Locate Points of Interest in the 2010s

In the first half of the 2000s, the popularity of tourist guide books faded out. The rise of Web 2.0 impacted the tourist market: like virtual travel agencies replaced physical ones, so collaborative tourist guides such as TripAdvisor[2], Yelp[3] or Qype[4] eroded the market of guide books. Today tourists commonly refer to those web sites when planning a travel.

In the second half of the 2000s, the search engine industry intercepted this trend by introducing *local search*: search facilities to find POIs described on the Web while on the go [2, 3, 4]. In local searches, users' preferred POIs are shown: those located close to the user current position [5, 6, 7], those for which ratings are available [8, 7] and those that match their preferences [9, 10]. Nowadays, a large number of tourists searches the Web for POIs while on the go.

In the 2010s, the increasing availability of GPS-enabled smart phones allowed the wide-spread of Location-based Services (LBSs), i.e., mobile and Web applications that ask users to *check-in* and, in response, provide context-dependent information and services. Examples of LBSs are foursquare[5], Gowalla[6] and Facebook Places[7]. LBSs can almost *listen to the pulse of the city*, therefore many tourists, when choosing which museum to visit or where to dine out, look up the POIs with the highest amount of check-ins on LBSs.

[2]Cf. `http://www.tripadvisor.com/`
[3]Cf. `http://www.yelp.com/`
[4]Cf. `http://www.qype.com/`
[5]Cf. `http://foursquare.com/`
[6]Cf. `http://gowalla.com/`
[7]Cf. `http://www.facebook.com/about/location`

2.2. Stream Reasoning

In 2008, Della Valle et al. in [11] called the Semantic Web community for techniques able to *reason upon rapidly changing information*. When reasoning on massive data streams, such as those characterizing BOTTARI, well known artificial intelligence techniques have the right level of expressivity, but their throughput is not high enough to keep pace with the stream (e.g., belief revision [12]). The only technological solutions with the right throughput are Data Stream Management Systems (DSMS) [13] and Complex Event Processing [14], but, on the other hand, they are not expressive enough. A new type of inference engines is thus needed to reason on streams. Della Valle et al. named them *stream reasoners*.

In the following years, a number of stream reasoning approaches have been developed ([15, 16, 17, 18]). They share three main concepts: *a*) they logically model the information flow as an RDF stream, i.e. a sequence of RDF triples annotated with one or more non-decreasing timestamps, *b*) they process the RDF streams "on the fly", often by re-writing queries to the raw data streams, and *c*) they exploit the temporal order of the streaming data to optimize the computation.

BOTTARI uses both a deductive and an inductive stream reasoner. The *deductive stream reasoner* is based on Continuous SPARQL (C-SPARQL) [15] – an extension of SPARQL that continuously processes RDF streams observed through windows (as done in DSMS). The syntax and semantics of C-SPARQL were described in [19]. The C-SPARQL execution engine and its optimization techniques were illustrated in [20]. The optimization needed for high-throughput RDFS++ reasoning are described in [21]. The approach to publish RDF stream as Linked Data (namely Streaming Linked Data) is based on [22].

The *inductive stream reasoner* is based on SUNS (Statistical Unit Node Set) approach [23, 24] – a scalable machine learning framework for predicting unknown but potentially true statements by exploiting the regularities in structured data. SUNS employs a modular regularized multivariate learning approach able to deal with very high-dimensional data [25] and to integrate temporal information using a Markov decomposition [26].

2.3. The LarKC platform

The *Large Knowledge Collider* (LarKC) is an EU FP7 Integrated Project [27] that aimed at massive distributed incomplete reasoning. The LarKC platform [28] is one of the main results of the project. It is a pluggable Semantic Web framework that can be deployed on a high-performance computing cluster. It allows for orchestrating multiple heterogeneous units for data processing and reasoning (named plug-ins), and for exposing their aggregated capabilities as a SPARQL endpoint.

3. The BOTTARI Mobile App

As shown in Figure 2, BOTTARI is an Android application (for smart phones and tablets) in augmented reality (AR) that directs the users' attention to restaurants and dining places in the neighbourhood of their position.

Figure 2: Some screenshots of the BOTTARI Android application.

In Korean language, "bottari" is a cloth bundle that carries a person's belongings while travelling. BOTTARI carries the collective perceptions of social media users about POIs in an area and uses them to recommend POIs. As shown in the upper left corner of the screenshot in Figure 2(a), BOTTARI users can search POIs in their proximity using four buttons:

1. *For me* that emphasises the personalization of POI suggestions as in local search studies like [9, 10];
2. *Popular* that emphasises the presence of positive ratings of social media users as in local search studies like [8, 7];
3. *Emerging* that focuses on the most recent ratings posted on social media that capture the seasonal effects (e.g., Insadong people seems to prefer meat restaurants in winter rather than in summer) or the POIs "on fashion" only for a limited period; and
4. *Interesting* that returns the POIs described with a category of interest for the user.

By pointing the device to frame the surrounding environment, the users see in AR the recommended POIs, as shown in Figure 2(a). In this view, the POIs are indicated with different icons (e.g., restaurants with or snack bars with) and their reputation is indicated by thumb-up and thumb-down icons, which means that the POI is collectively perceived positively or negatively.

Moreover, given the importance of the distance between the user and the recommended POIs [5, 6, 7], BOTTARI offers a functionality for distance-based filtering of the recommended POIs; see the circles in the right-upper side of Figure 2(a).

The user can learn more about a POI by touching its icon. As shown in Figure 2(b) and 2(c), BOTTARI can display the POI's details. Figure 2(d) shows a peculiar feature of BOTTARI: the trend over time of the POI reputation as collectively perceived on social media.

A video displaying BOTTARI at work, on a mobile phone and on a tablet, is available on YouTube at `http://www.youtube.com/watch?v=c1FmZUz5BOo`.

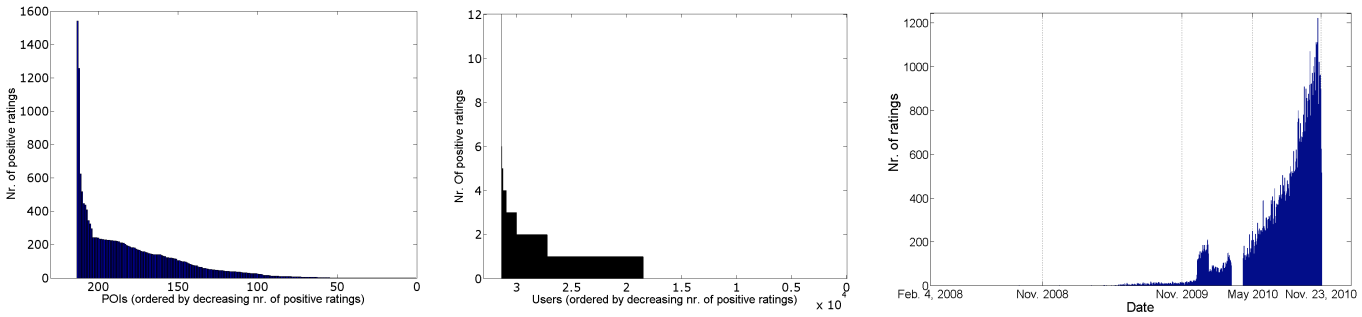
4. Datasets Used in BOTTARI

BOTTARI is built on two types of data: the *static* descriptions of the POIs and the social media *streams*.

4.1. Static Descriptions of the POIs

Insadong is a 2 km^2 district with a high density of restaurants. For BOTTARI, the information about the 319 restaurant of Insadong was collected with a considerable manual effort from Yelp[8], PoiFriend[9], Yahoo! Local[10], TrueLocal[11], several

[8] Cf. `http://www.yelp.com/`
[9] Cf. `http://www.poifriend.com/`
[10] Cf. `http://local.yahoo.com/`
[11] Cf. `http://www.truelocal.com.au/`



(a) (b) (c)

Figure 3: Dataset statistics: (a) positive ratings for POIs, (b) positive ratings by users and (c) tweet distribution over time.

Korean restaurant Web sites and a few Korean portals. The result is a manually curated high quality geo-referenced knowledge base where each restaurant is described by 44 attributes (e.g., name, images, position, address, ambiance, specialities, categories, etc.).

4.2. Social Media Streams

The social media streams are gathered from the Web (in particular from Twitter) and converted into an RDF stream using the proprietary crawling and sentiment mining infrastructure of Saltlux. The data used for the experiments (cf. also Section 8) were collected in 3 years, from February 4^{th}, 2008 to November 23^{rd}, 2010 (1,023 days). 200 million tweets were analysed and, as a result, 109,390 tweets posted by more than 31,369 users were discovered to positively, neutrally or negatively talk about 245 restaurants.

		Tweet	POI	User	Sparsity
Ratings	Positive	19,045	213	12,863	99.30%
	Negative	14,404	181	10,448	99.24%
	Neutral	75,941	245	28,056	98.90%
	Total	109,390	245	31,369	98.58%

Table 1: Statistics of the data set

Table 1 illustrates some statistics of the collected data set:

- *high sparsity* – defining sparsity as $1 - \frac{\#Ratings}{\#POIs \times \#Users}$, for instance, the sparsity of the positive ratings is 99.3%;
- *incompleteness* – only 41% of users positively rated at least one POI (see also Figure 3(a) and (b));
- *inconsistencies* – the same user can rate a particular POI several times expressing different opinions;
- *exponential growth of micro-posts in time* – Figure 3(c) shows the exponential growth in the usage of Twitter in Korea starting from 2009[12]; and
- *long-tail distribution* – Figures 3(a) and 3(b) show the long-tail distribution of positive ratings over POIs and users respectively.

[12]The peak in winter 2010 and the missing data in spring 2010 are due to changes and improvements on the crawling algorithm.

4.3. Comparison with Traditional Channels

BOTTARI information is very rich if compared to what a tourist can obtain from tourist guides and Web 2.0 sites.

Guide books in average list ten restaurants in Insadong and they provide a professional review. For instance, the popular RoughGuides series[13], which has an edition dedicated to South Korea, lists only 8 restaurants and one of them is even outside the Insadong district. BOTTARI knowledge base contains all of them; they are classical tourist places with a medium-low number of positive ratings.

Web 2.0 sites in average list twenty restaurants reviewed by 3-4 users. For instance, TripAdvisor has information about 13 restaurants in Insadog district reviewed on average by 3 users and maximum by 12 users. BOTTARI knowledge base contains 12 restaurant present in TripAdvisor; on average they have been rated positively by tens of users, only one received a hundred positive ratings.

5. Ontology Used in BOTTARI

We designed BOTTARI following an ontology-based information access architecture [29]. BOTTARI ontology is represented in Figure 4. It extends the SIOC vocabulary [30] defining `TwitterUser` as a special case of `UserAccont` and the concept of `Tweet` as being equivalent to `Post`. It models the notion of POI as `NamedPlace` extending `SpatialThing` from the W3C WGS-84 vocabulary[14]. A `NamedPlace` is enriched with a categorization (e.g., the ambience describing the atmosphere of a restaurant) and the count of positive/negative/neutral ratings. The most distinctive feature of BOTTARI ontology is the object property `talksAbout` – and its sub-properties for positive, negative and neutral opinions – that allows to state that a `Tweet` (positively, negatively or neutrally) talks about a `NamedPlace`.

6. Architecture and Components

BOTTARI architecture is illustrated in Figure 5. It consists of three parts: *a*) a client (described in Section 3) that interacts with the user and communicates to the back-end sending

[13]Cf. `http://www.roughguides.com/`

[14]Cf. `http://www.w3.org/2003/01/geo/`

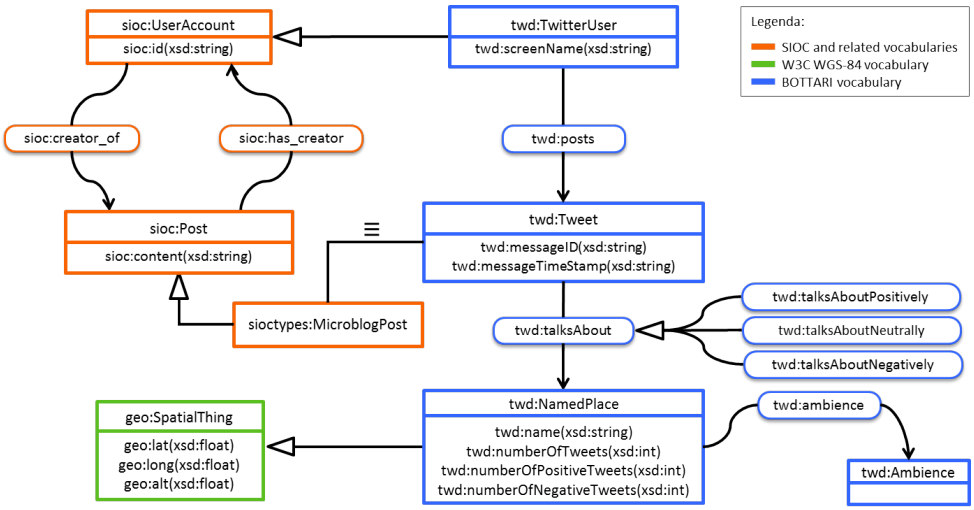


Figure 4: Ontology modelling of BOTTARI data.

SPARQL queries, *b*) a data initiated segment (PUSH) that continuously analyses the social media streams, and *c*) a query initiated segment (PULL) that uses the LarKC platform to answer the SPARQL queries of the client by combining several forms of reasoning.

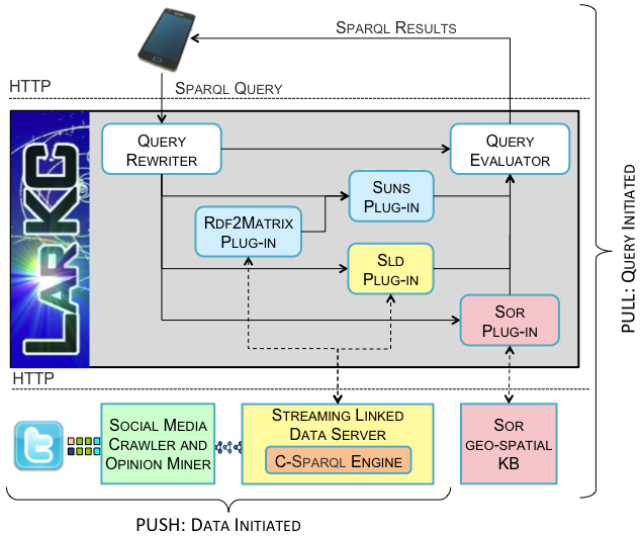


Figure 5: Architecture of BOTTARI back-end. A data initiate (i.e., PUSH) part of the system continuously analyse microposts, while BOTTARI client can issue SPARQL queries to the query initiated (i.e., PULL) part of the system. The plug-able Semantic Web platform LarKC couples the PULL part of the system with the PUSH one.

6.1. The PUSH segment

The PUSH segment continuously analyses the social media streams crawled from the Web. The SEMANTIC MEDIA CRAWLER AND OPINION MINER crawl 3.4 million tweets/day related to Seoul, identifies the subset related to the Insadong restaurants (thousands per day) and extracts the users' opinions[15]. The result is an RDF stream of positive, negative and neutral ratings of the restaurants of Insadong. Listing 1 shows a segment of this RDF stream: two tweets are represented by four triples using the vocabulary of Section 5.

```
(<:Alice  :posts  _:t1 >,                    2011-10-12T13:34:41)
(<_:t1  :talksPositivelyAbout :TheVolga >, 2011-10-12T13:34:41)
(<:Bob :posts _:t2 >,                        2011-10-12T13:35:07)
(<_:t2  :talksNegativelyAbout :TheVolga >, 2011-10-12T13:35:07)
```

Listing 1: Social media RDF stream

The RDF stream flows at an average rate of a hundred tweets/-day, peaking at tens of tweets/minute. The RDF stream is processed in real-time by the STREAMING LINKED DATA SERVER (SLD SERVER) by means of the network of C-SPARQL queries illustrated in Figure 6.

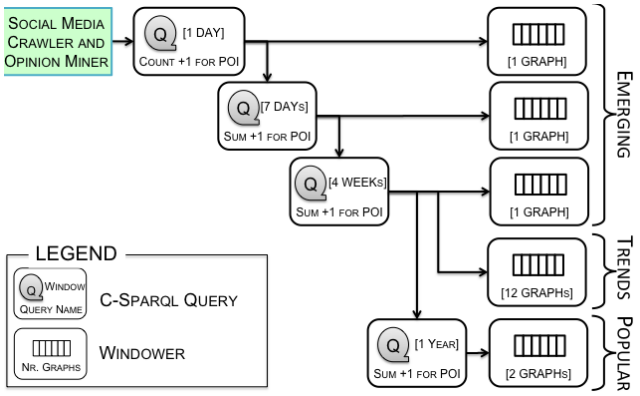


Figure 6: The network of C-SPARQL that continuously analyses the RDF stream produced by the SOCIAL MEDIA CRAWLER AND OPINION MINER and keeps up to date the data used for the *Emerging* and *Polular* recommendations

In particular, the query COUNT +1 FOR POI IN [1 DAY] counts the positive ratings for each POI in one day. The query

[15]The OPINION MINER is configured so to boost precision, which is 90%, at the expense of recall.

SUM +1 FOR POI IN [7 DAYS] aggregates the result of the previous one over a week and, similarly, the query SUM +1 FOR POI IN [4 WEEKS] computes this aggregation over a month.

The results of each query are published as linked data by the WINDOWERs. These counts will be used in the PULL segment to answer the SPARQL queries for the *Emerging* recommendations and to display the trend lines illustrated in Figure 2(d).

Moreover, the SUM +1 FOR POI IN [1 YEAR] query further aggregates the results of the queries upstream over one year. Its results, made available as linked data by another windower, are used to compute the *Popular* recommendations in the PULL segment.

The last component of the PUSH segment is BOTTARI inductive stream reasoner SUNS. The RDF2MATRIX PLUG-IN daily takes the results of the COUNT +1 FOR POI IN [1 DAY] query and updates the inductive materialization used for the *For me* recommendations.

6.2. The PULL segment

The PULL segment is based on the LarKC platform, which acts as an ontology-based information-integration platform [29]: BOTTARI ontology logically integrates the data models of the different plug-ins involved in computing a given type of recommendations. Whenever a user presses one of the four recommendation buttons in BOTTARI interface, the client issues a query using the BOTTARI ontology. When the QUERY REWRITER receives the query, it decomposes it into a set of queries, one for each plug-in. The plug-ins are executed in parallel. Each plug-in receives its re-written query and sends its partial results to the QUERY EVALUATOR. This plug-in joins the partial results and returns the complete answer to the client, as if the query had been evaluated on a single integrated knowledge base. Caching of entire queries and intermediate results is applied in order to minimize query latency.

The plug-ins involved in computing BOTTARI recommendations are three:

- Given a location, a spatial orientation and a POI category, the SOR PLUG-IN returns a list of POIs ordered by distance from the location. It delegates the query execution to SOR, the spatial-aware RDF store by Saltlux.
- Given a user, the SUNS PLUG-IN returns a list of POIs ordered by the estimated probability that the user will like them. It uses the inductive materialization maintained up-to-date by the RDF2MATRIX PLUG-IN.
- Given a period (i.e., a day, a week, a month, or two years), the SLD PLUG-IN returns a list of POIs ordered by the number of tweets that talk positively about the POI in that period. It uses the linked data published in SLD windowers (see Figure 6).

To better clarify how we configured the LarKC platform to evaluate the BOTTARI client requests, let us consider the query in Listing 2 that represent a mix of the queries the client sends for *Interesting* (lines 3-7), *For me* (lines 8-11) and *Emerging* (line 12) recommendations.

```
1. SELECT ?poi ?name ?lat ?long
2. WHERE {
3. ?poi a ns:NamedPlace ;  ns:name ?name ;
4.      geo:lat ?lat ;  geo:long ?long ;
5.      ns:category :InterestingForForeigners .
6. FILTER(:within_distance(37.5,126.9,?lat,?long,200))
7. FILTER(:dest_point_viewing(37.5,126.9,?lat,?long,90,200))
8. { :Alice sioc:creator_of ?tweet
9.   ?tweet twd:talksAboutPositively ?poi .
10.   WITH PROBABILITY ?prob
11.   ENSURE PROBABILITY [0.5..1)  }
12. ?poi twd:numberOfPositiveTweetsInTheMonth ?numPos .
13. }
14. ORDER BY DESC(?numPos * ?prob * distance(37.5,126.9,
                                              ?lat,?long,200))
15. LIMIT 10
```

Listing 2: Sample BOTTARI client query in SPARQL.

Lines 3-7 ask for POIs that may be of interest for foreigners (line 5), located within 200 meters from the user position (line 6), when looking east (line 7). This is the portion of query that the SOR PLUG-IN can evaluate. The QUERY REWRITER knows this: when rewriting the query, it extracts lines 3-7 and sends them to the SOR PLUG-IN. We do not show the rewritten query, because it is simply a subset of the query in Listing 2.

Lines 8-11 add the additional constraint that the requesting user has not yet talked positively about these POIs, but may do it in future (i.e., it is probable that the user will like them). Lines 10-11 makes use of *SPARQL with probability* [24]. The triple pattern at lines 8-9 matches triples in the inductive materialization asserting that a user will likely positively talk about a POI. The `WITH PROBABILITY` clause at line 10 extends SPARQL with the probability values associated to each triple in the inductive materialization. The variable `?prob` may assume values between 0 and 1, where 1 means that the user has already positively rated the POI. The `ENSURE PROBABILITY` clause at line 11 accepts a pattern solution to lines 8-9 only if its estimated probability is greater or equal to 0.5 but strictly less than 1. The QUERY REWRITER extracts lines 8-11 from the query in Listing 2 and rewrites them in a query for the SUNS PLUG-IN, as illustrated in Listing 3.

```
1. CONSTRUCT {
2.   :Alice twd:talksAboutPositively
3.     [ ns:about ?poi ;  ns:withProbability ?prob ]  }
4. WHERE {
5.   :Alice sioc:creator_of ?tweet
6.   ?tweet twd:talksAboutPositively ?poi .
7.  WITH PROBABILITY ?prob
8.  ENSURE PROBABILITY [0.5..1)  }
9. ORDER BY DESC(?prob)
```

Listing 3: The query in Listing 2 as rewritten for the SUNS PLUG-IN.

Notably, in Listing 3 the `WHERE` clause (lines 5-8) corresponds to lines 8-11 in Listing 2. The peculiarity of the query in Listing 3 is the `CONSTRUCT` clause. It allows to embed the probability values in the query results of the SUNS PLUG-IN without using annotations or reification. An example of this query results is illustrated in Listing 4.

```
:Alice ns:talksAboutPositively
   [ ns:about :TheBanqueting ;  ns:withProbability  "0.8" ] ,
   [ ns:about :JoteunSeed ;  ns:withProbability  "0.6" ] .
```

Listing 4: Sample results to the query in Listing 3.

The query in Listing 2 also contains a triple pattern that the QUERY REWRITER will use to prepare the rewritten query for the SLD PLUG-IN. Line 12 adds the additional requirement that a number of tweets in the last month must have talked positively about the POIs.

The three plug-ins are executed in parallel and produce a list of POIs each. The QUERY EVALUATOR takes care of computing the global answer to the original query by evaluating the query in Listing 5 on the combined results of the three rewritten queries. Notably, at lines 6-8, the QUERY EVALUATOR assumes the existence of the triples generated by the SUNS PLUG-IN with Listing 3.

```
1. SELECT ?poi ?name ?lat ?long
2. WHERE {
3. ?poi a ns:NamedPlace ;  ns:name ?name ;
4.   geo:lat ?lat ;  geo:long ?long ; ns:distance ?distance .
5. :Alice twd:talksAboutPositively
6.   [ ns:about ?poi ; ns:withProbability  ?prob ]
7. ORDER BY DESC(?numPos * ?prob / distance(37.5,126.9,
                                        ?lat,?long,200))
8. LIMIT 10
```

Listing 5: The query in Listing 2 as rewritten for the QUERY EVALUATOR.

A final remark on lines 14-15 in Listing 2 (reported in Listing 5 as lines 7-8). Line 14 asks for the POIs to be ordered: the first result must be the POI with the highest number of positive tweets, with the highest estimated probability for the user to talk positively about it and the closest one to the user position. Finally, line 15 limits the returned results to the first 10. The `ORDER BY` and `LIMIT` clauses qualify the query in Listing 2 as a top-k query. This type of queries is subject to optimizations [31] typically unavailable in SPARQL [32].

7. Evaluation

The quality and the efficacy of BOTTARI recommendations was comparatively evaluated using the data set described in Section 4.

7.1. Methodology

We decided to measure normalized discounted cumulative gain (NDCG) and accuracy at top 10 (ACC@N) averaged among all users. The *For me*, *Popular* and *Interesting* recommendations were compared with two baselines: random guess (Random) and k-nearest neighbour (KNNItem)[16]. The combination of *For me* and *Popular* recommendations was also considered. For all recommendations, the distance filter was not applied, because our data set does not contain the user position at twitting time.

Two types of ground truths were prepared for the evaluation. They correspond to two experimental settings:

- Setting 1: the standard method of splitting the data into a training set and a test set was used. In this case, a ground truth contains one positive rating for each user randomly withheld from the data set. We repeated this data split five times.
- Setting 2: specifically for the *Emerging* recommendations, which uses a time window, a set of ground truths was created by withholding the newest rating for each user. Different time frames were considered: 1 day, 2 days, 7 days, 30 days, 90 days and 180 days. Table 2 shows the number of ratings considered for the *Emerging* recommendations.

	Nr. of ratings	%
Last day	188	0.17
Last 2 days	703	0.64
Last 7 days	5,057	4.62
Last 30 days	27,049	24.73
Last 90 days	65,600	70.01
Last 180 days	93,696	85.65
Total	109,389	100.00

Table 2: Number of ratings with different time frames

7.2. Results

Figure 7 shows the results we obtained in the two settings. In Figure 7(a), the NDCG of the tested methods are plotted against the number of latent variables (an input of the learning model). Since the two baselines (Random and KNNItem) and the *Popular* recommendations are independent of this number, they produce three horizontal lines. We evaluated *For me* recommendations produced by SUNS with 20, 50, 100, 150 and 200 latent variables. As expected, Random was the worst. The *Popular* recommendations were slightly better than KNNItem. This might be due to the "bandwagon effect" that exists in many social communities. The *For me* recommendations significantly outperformed all the others after the number of the latent variables reached 100. The best ranking ever was produced by the combination of both *For me* and *Popular* recommendations. These results confirm the idea presented in [1] that a combined approach of deductive and inductive stream reasoning works best.

Figure 7(b) shows the accuracy of the top N (ACC@N) recommended POIs for $N = \{5, 10, 15, 20, 25, 30\}$. The quality of the *For me* recommendations was much higher than that of all other methods and, once again, the combination of *For me* and *Popular* recommendations is the best option overall.

[16] Let P be the number of POIs and U be the number of users. For the baseline KNNItem, we used the cosine as the similarity measure of POIs defined as $similarity(p_i, p_j) = \frac{<p_i, p_j>}{\|p_i\| * \|p_j\|}$, where p_i and p_j represent the vectors of ratings of the i-th and the j-th POI given by all users for $i, j \in \{1, \ldots, P\}$, and where $<\cdot, \cdot>$ is the scalar product of two vectors and $\| \cdot \|$ is the 2-norm of a vector. We set k to the total number of the POIs.

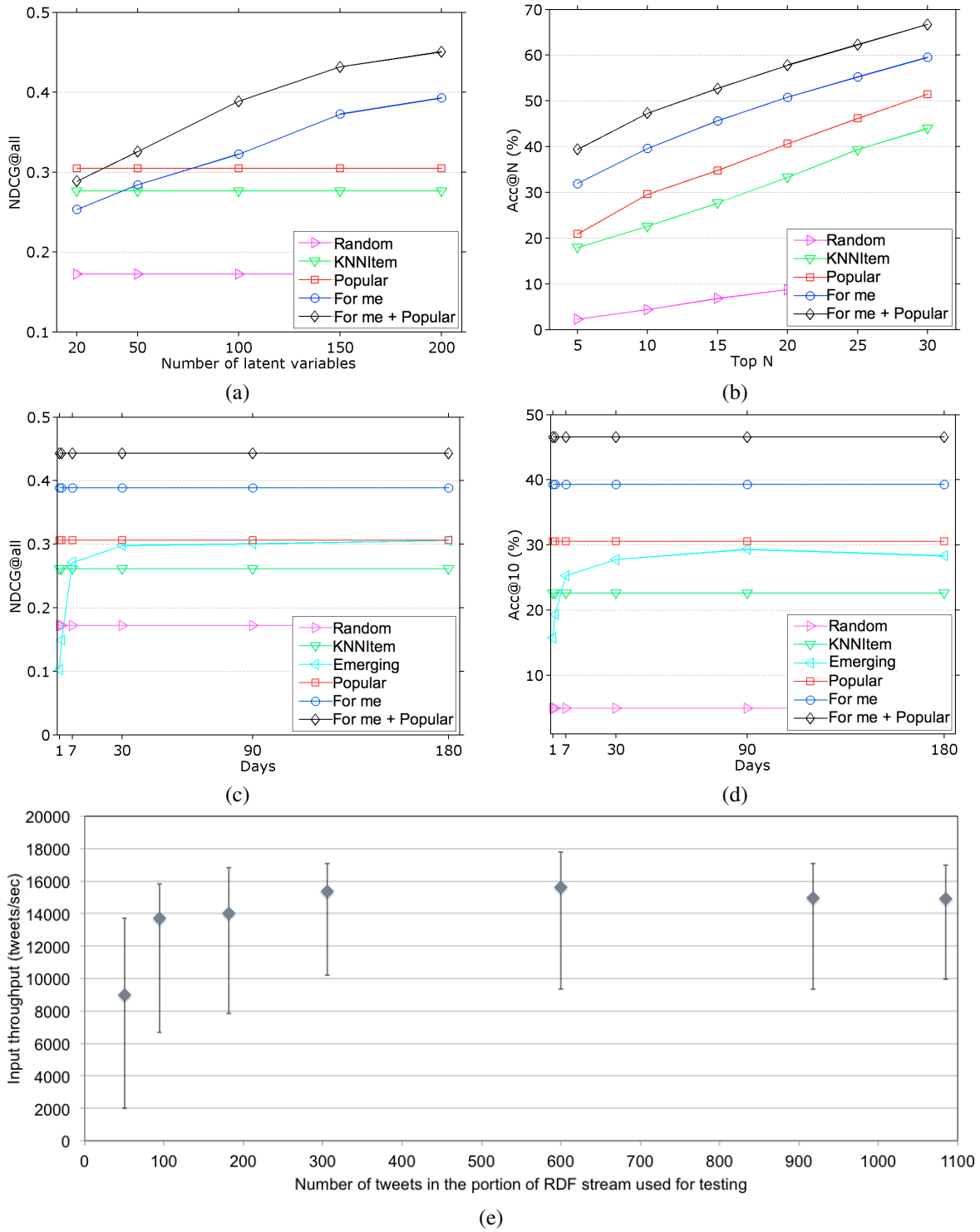


Figure 7: The figure shows the BOTTARI evaluation results. Sub-figures (a) and (b) respectively show the NDCG scores and the accuracy values at top N for all types of recommendation (except Emerging) in Setting 1. The combination of *For me* and *Popular* recommendations produces the best ranking. Sub-figures (c) and (d) focus on Setting 2, where a special emphasis is put on the dependency of *Emerging* recommendations on the window size (i.e., 1, 7, 30, 90 and 180 days). The nDCG scores and the accuracy values at top 10 shows that the *Emerging* recommendations can be nearly as effective as the *Popular* recommendations, keeping only a small fraction of the full history. Finally, sub-figure (e) shows the results of the scalability test on the PUSH segment of BOTTARI back-end, which is able to handle up to 15,000 tweets/second.

A key aspect of BOTTARI is the adoption of stream reasoning techniques that build on the hypothesis that a long enough window can capture all the information needed for a given task, while the rest can be forgotten. In Setting 2, we compared the *Emerging* recommendations, which use a time window, against the other approaches. We varied the size of the time window from 1 day to 180 days. Figures 7(c) and (d) plot the NDCG scores and the accuracy at top 10 (ACC@10) as a function of the length of the window. The *Emerging* recommendations, the only curve in the figure, catches the NDCG of *Popular* recommendations (which consider years of data) when using the last 30 days of ratings, and was very close to the accuracy at top-10 of the *Popular* recommendations when using the last 90 days of ratings.

This fact tells us that, in this setting, the *Emerging* recommendations with a 90 days window are nearly as effective as the *Popular* recommendations that keep the full history (i.e., two years of data).

8. Scalability

The SOCIAL MEDIA CRAWLER (cf. Figure 5) probes hundreds of thousand tweets/day, but the RDF stream produced by the OPINION MINER contains an average of 150 RDF triples/-day (corresponding to 75 tweets). The large majority of the crawled tweets are not related to Insadong's restaurants. The flow rate of this RDF stream does not stress the PUSH segment of BOTTARI back-end that runs on a laptop with CPU 2.8 GHz Intel Core i7 and 8 GB RAM DDR3, which corresponds to a 150 €/month share in a cloud environment[17].

8.1. SUNS Scalability

On the deployment machine, the training of SUNS on the whole dataset takes approximately 86 seconds with 200 latent variables and the recommendation of POIs for a user costs on average less than 5 milliseconds. Internal studies have confirmed that, by exploiting sparsity, SUNS computational requirements scales linearly with the number of *known* ratings. In addition, we observed that SUNS is very robust and insensitive to the number of latent variables. This can be explained by the fact that SUNS, in contrast to other matrix factorization methods such as Singular Value Decomposition (SVD), is regularized. This property can simplify the use of SUNS, in particular for people without machine learning expertise.

8.2. SLD Scalability

To evaluate the scalability of SLD SERVER, we adopted a technique used in publish-subscribe systems [33]. We measured the *input throughput*, i.e. the ability to consume the stream inputs, computed as:

$$input\,throughput = \frac{size\,input}{time\,to\,process\,the\,input}$$

[17]The calculation of the cost per month was done using `https://www.gandi.net/hosting/vps`.

We measure the input throughput by sending a recorded portion of the RDF stream to the SLD SERVER and by measuring the time required to process it. We used seven portions of the recorded RDF stream of growing length: the shortest one contains 50 tweets recorded between May 1^{st} and May 2^{nd}, 2010; the longest one contains 1,085 tweets recorded between May 1^{st} and June 3^{th}, 2010. To improve the confidence, we repeated each experiment for 500 times and we measured average, minimum and maximum time required to process the RDF stream portion.

The results of these scalability tests on the deployment machine are illustrated in Figure 7(e). Initially, when the SLD SERVER is able to keep the pace of the growing rate of tweets/second in input, the input throughput increases with the number of tweets in the recorded segment. When the input throughput reaches 15,000 tweets/second, the SLD SERVER saturates the available computational resources and is no longer able to handle the whole input. This is a common behaviour in DSMSs, where this kind of tests allows to dimension the input queue to handle pick rates exceeding this saturation point.

9. Conclusions and Future Work

BOTTARI is a sophisticated application of semantic technologies that makes use of the rich and collective knowledge obtained by continuously analysing social media streams. We believe it was important to hide this complexity from the user using an intuitive and easy to use interface. The preliminary experiments we conducted show that BOTTARI can be more effective than guide books and Web 2.0 travel review sites.

Inspired by the literature on ontology-based information access, we design BOTTARI ontology as driver of both data and service integration. It allows for combining real data sources at real scale, i.e. location-specific static information about hundreds of POIs with the results of continuous analysis of dynamic social media streams. However, we believe that the BOTTARI ontology was also crucial in handling the heterogeneous data models of the plug-ins. For instance, the inductive reasoner annotates triples in the inductive materialization with their probability to be true, but the other plug-ins cannot understand these annotations, unless they are transformed into commonly described data (see Listings 3).

BOTTARI is engineered for scalability. Both SUNS and SLD show a scalability that goes largely beyond the actual needs of the BOTTARI deployment in Insadong. Training SUNS over two years of data takes 1.5 minutes. SLD can handle a flow of 15,000 tweets/second when the actual rate is tens of tweets/day. These results convinced Saltlux to start a large-scale deployment of BOTTARI in Korea.

Our future work will be devoted to extend BOTTARI functionalities, to reduce its production costs and to further improve the technological solution.

We intend to extend BOTTARI by identifying and recommending the "mavens of a POI" (i.e., social media users that post a large number of micro-posts related to that POI and are known to influence the opinion of other social media users) as

described in [34] and to cope with an evolving world (e.g., new restaurants open, some restaurants close, etc.). We would like to reduce the production costs of BOTTARI, in particular those associated to the manual creation of the spatial-aware knowledge base of the POIs. To this end, we look forward to integrating BOTTARI with LOD sources. We are aware that published data sets may not have sufficient quality, so a first step is to consider the LOD data available for Insadong and to check the differences on the metric results compared to the manually curated KB used for BOTTARI. If this step turns out to be successful, extending BOTTARI to bigger area, e.g., such as the full Seoul, is the necessary step to assess the feasibility of commercial spreading of BOTTARI.

Finally, we intend to improve BOTTARI back-end by studying different strategies to deal with the inconsistent ratings and opinion changes about a POI and by applying to SPARQL some optimizations studied for top-k queries in relational databases.

Acknowledgments

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

References

[1] D. F. Barbieri, D. Braga, S. Ceri, E. Della Valle, Y. Huang, V. Tresp, A. Rettinger, H. Wermser, Deductive and Inductive Stream Reasoning for Semantic Social Media Analytics, IEEE Intelligent Systems 25 (6) (2010) 32–41.

[2] M. Kamvar, S. Baluja, A large scale study of wireless search behavior: Google mobile search, in: Proceedings of the SIGCHI conference on Human Factors in computing systems, CHI '06, ACM, New York, NY, USA, 2006, pp. 701–709.

[3] M. Kamvar, S. Baluja, Deciphering trends in mobile search, IEEE Computer 40 (8) (2007) 58–62.

[4] J. Yi, F. Maghoul, J. O. Pedersen, Deciphering mobile search patterns: a study of yahoo! mobile search queries, in: J. Huai, R. Chen, H.-W. Hon, Y. Liu, W.-Y. Ma, A. Tomkins, X. Zhang (Eds.), WWW, ACM, 2008, pp. 257–266.

[5] J. Froehlich, M. Y. Chen, I. E. Smith, F. Potter, Voting with your feet: An investigative study of the relationship between place visit behavior and preference, in: P. Dourish, A. Friday (Eds.), Ubicomp, Vol. 4206 of Lecture Notes in Computer Science, Springer, 2006, pp. 333–350.

[6] K. Berberich, A. C. König, D. Lymberopoulos, P. Zhao, Improving local search ranking through external logs, in: W.-Y. Ma, J.-Y. Nie, R. A. Baeza-Yates, T.-S. Chua, W. B. Croft (Eds.), SIGIR, ACM, 2011, pp. 785–794.

[7] D. Lymberopoulos, P. Zhao, A. C. König, K. Berberich, J. Liu, Location-aware click prediction in mobile local search, in: C. Macdonald, I. Ounis, I. Ruthven (Eds.), CIKM, ACM, 2011, pp. 413–422.

[8] A. Amin, S. Townsend, J. van Ossenbruggen, L. Hardman, Fancy a drink in canary wharf?: A user study on location-based mobile search, in: T. Gross, J. Gulliksen, P. Kotzé, L. Oestreicher, P. A. Palanque, R. O. Prates, M. Winckler (Eds.), INTERACT (1), Vol. 5726 of Lecture Notes in Computer Science, Springer, 2009, pp. 736–749.

[9] X. Shen, B. Tan, C. Zhai, Context-sensitive information retrieval using implicit feedback, in: R. A. Baeza-Yates, N. Ziviani, G. Marchionini, A. Moffat, J. Tait (Eds.), SIGIR, ACM, 2005, pp. 43–50.

[10] Z. Dou, R. Song, J.-R. Wen, A large-scale evaluation and analysis of personalized search strategies, in: C. L. Williamson, M. E. Zurko, P. F. Patel-Schneider, P. J. Shenoy (Eds.), WWW, ACM, 2007, pp. 581–590.

[11] E. Della Valle, S. Ceri, F. van Harmelen, D. Fensel, It's a Streaming World! Reasoning upon Rapidly Changing Information, IEEE Intelligent Systems 24 (6) (2009) 83–89.

[12] P. Gaerdenfors (Ed.), Belief Revision, Cambridge University Press, 2003.

[13] M. Garofalakis, J. Gehrke, R. Rastogi, Data Stream Management: Processing High-Speed Data Streams, Springer-Verlag New York, Inc., 2007.

[14] D. Luckham, The power of events: An introduction to complex event processing in distributed enterprise systems, in: N. Bassiliades, G. Governatori, A. Paschke (Eds.), Rule Representation, Interchange and Reasoning on the Web, Vol. 5321 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2008, pp. 3–3.

[15] D. F. Barbieri, D. Braga, S. Ceri, E. Della Valle, M. Grossniklaus, Querying rdf streams with c-sparql, SIGMOD Record 39 (1) (2010) 20–26.

[16] D. Anicic, P. Fodor, S. Rudolph, N. Stojanovic, Ep-sparql: a unified language for event processing and stream reasoning, in: S. Srinivasan, K. Ramamritham, A. Kumar, M. P. Ravindra, E. Bertino, R. Kumar (Eds.), WWW, ACM, 2011, pp. 635–644.

[17] T. Do, S. Loke, F. Liu, Answer set programming for stream reasoning, in: C. Butz, P. Lingras (Eds.), Advances in Artificial Intelligence, Vol. 6657 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2011, pp. 104–109.

[18] J.-P. Calbimonte, Ó. Corcho, A. J. G. Gray, Enabling ontology-based access to streaming data sources, in: P. F. Patel-Schneider, Y. Pan, P. Hitzler, P. Mika, L. Zhang, J. Z. Pan, I. Horrocks, B. Glimm (Eds.), International Semantic Web Conference (1), Vol. 6496 of Lecture Notes in Computer Science, Springer, 2010, pp. 96–111.

[19] D. F. Barbieri, D. Braga, S. Ceri, E. Della Valle, M. Grossniklaus, C-sparql: a continuous query language for rdf data streams, Int. J. Semantic Computing 4 (1) (2010) 3–25.

[20] D. F. Barbieri, D. Braga, S. Ceri, M. Grossniklaus, An execution environment for c-sparql queries, in: I. Manolescu, S. Spaccapietra, J. Teubner, M. Kitsuregawa, A. Léger, F. Naumann, A. Ailamaki, F. Özcan (Eds.), EDBT, Vol. 426 of ACM International Conference Proceeding Series, ACM, 2010, pp. 441–452.

[21] D. F. Barbieri, D. Braga, S. Ceri, E. Della Valle, M. Grossniklaus, Incremental Reasoning on Streams and Rich Background Knowledge, in: Proc. of ESWC2010, 2010.

[22] D. F. Barbieri, E. Della Valle, A proposal for publishing data streams as linked data - a position paper, in: C. Bizer, T. Heath, T. Berners-Lee, M. Hausenblas (Eds.), LDOW, Vol. 628 of CEUR Workshop Proceedings, CEUR-WS.org, 2010.

[23] V. Tresp, Y. Huang, M. Bundschus, A. Rettinger, Materializing and querying learned knowledge, in: Proc. of IRMLeS 2009, 2009.

[24] Y. Huang, V. Tresp, M. Bundschus, A. Rettinger, H.-P. Kriegel, Multivariate prediction for learning on the semantic web, in: P. Frasconi, F. A. Lisi (Eds.), ILP, Vol. 6489 of Lecture Notes in Computer Science, Springer, 2010, pp. 92–104.

[25] Y. Huang, M. Nickel, V. Tresp, H.-P. Kriegel, A scalable kernel approach to learning in semantic graphs with applications to linked data, in: Proc. of the 1st Workshop on Mining the Future Internet, 2010.

[26] V. Tresp, Y. Huang, X. Jiang, A. Rettinger, Graphical models for relations - modeling relational context, in: International Conference on Knowledge Discovery and Information Retrieval, 2011.

[27] D. Fensel, F. van Harmelen, B. Andersson, P. Brennan, H. Cunningham, E. Della Valle, F. Fischer, Z. Huang, A. Kiryakov, T. K. il Lee, L. School, V. Tresp, S. Wesner, M. Witbrock, N. Zhong, Towards LarKC: a Platform for Web-scale Reasoning, in: Proc. of ICSC 2008, 2008.

[28] A. Cheptsov, et al., Large Knowledge Collider. A Service-oriented Platform for Large-scale Semantic Reasoning, in: Proceedings of WIMS 2011, 2011.

[29] M. Lenzerini, Data integration: A theoretical perspective, in: L. Popa (Ed.), PODS, ACM, 2002, pp. 233–246.

[30] D. Berrueta, et al., SIOC Core Ontology Specification, W3C Member Submission, W3C (2007).

[31] I. F. Ilyas, G. Beskales, M. A. Soliman, A survey of top-k query processing techniques in relational database systems, ACM Comput. Surv. 40 (4).

[32] A. Bozzon, E. Della Valle, S. Magliacane, Towards and efficient SPARQL top-k query execution in virtual RDF stores, in: 5th International Workshop on Ranking in Databases (DBRANK 2011), 2011.

[33] F. Fabret, H.-A. Jacobsen, F. Llirbat, J. Pereira, K. A. Ross, D. Shasha, Filtering algorithms and implementation for very fast publish/subscribe, in: SIGMOD Conference, 2001, pp. 115–126.

[34] I. Celino, D. Dell'Aglio, E. Della Valle, Y. Huang, T. Lee, S. Park, V. Tresp, Towards BOTTARI: Using Stream Reasoning to Make Sense of Location-based Micro-Posts, in: R. G.-C. et al (Ed.), ESWC 2011 Workshops, LNCS 7117, Springer-Verlag Berlin Heidelberg, 2011, pp. 80–87.