Undefined 1 (2009) 1–5
IOS Press

Reality Mining on Micropost Streams

Deductive and Inductive Reasoning for Personalized and Location-based Recommendations

Editor(s): Matthew Rowe, Lancaster University, UK; Milan Stankovic, Universit Paris-Sorbonne, France; Aba-Sah Dadzie, Sheffield University, UK
Solicited review(s): Jelena Jovanovic, University of Belgrade, Serbia; José Morales del Castillo, Universidad de Granada, Spain; Pablo Mendes, Wright State University, USA

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

[a] *Dip. Elettronica e Informazione – Politecnico di Milano, via Ponzio 34/5, 20133, Milano, Italy*
E-mail: balduini@elet.polimi.it,emanuele.dellavalle@polimi.it,daniele.dellaglio@mail.polimi.it
[b] *CEFRIEL – ICT Institute, Politecnico di Milano, via Fucini 2, 20133, Milano, Italy*
E-mail: irene.celino@cefriel.it
[c] *Siemens AG, Corporate Technology, Otto-Hahn-Ring 6, 81739 München, Germany*
E-mail: {yihuang,volker.tresp}@siemens.com
[d] *Saltlux, 7F, Deokil Building, 967, Daechi-dong, Gangnam-gu, Seoul 135-848 Korea*
E-mail: {shkim,tony}@saltlux.com

Abstract.
The rapid growth of personal opinions published in form of microposts, such as those found on Twitter, is the basis of novel emerging social and commercial services. In this paper, we describe BOTTARI, an augmented reality application that permits the personalized and localized recommendation of points of interest (POIs) based on the temporally-weighted opinions of the community. The technological basis of BOTTARI is the highly scalable LarKC platform for the rapid prototyping and development of Semantic Web applications. In particular, BOTTARI exploits LarKC's deductive and inductive stream reasoning. We present an evaluation of BOTTARI based on a three year collection of tweets about 319 restaurants located in the 2 km^2 district of Insadong, a popular tourist area of the South Korean city of Seoul. BOTTARI is the winner of the 9th edition of the Semantic Web Challenge, co-located with the 2011 International Semantic Web Conference. BOTTARI is currently field tested in Korea by Saltlux.

Keywords: reality mining, stream reasoning, personalized recommendation, social media analysis, mobile app

1. Introduction

First the Web and then mobile devices changed the way we all consume information. Both are becoming our daily information channels by progressively subtracting market shares from traditional press; at the same time, as interfaces to social media, they are also giving a voice to the people.

In the tourism market, this trend appeared in two waves. The first wave started in 2000, when TripAdvisor[1] was founded, and continued with the rise of user review sites and local search Web sites, like Yelp[2] and Yahoo! Local[3] (both launched in 2004), Google Maps[4]

*Corresponding author. E-mail: irene.celino@cefriel.it.
**Corresponding author. E-mail: emanuele.dellavalle@polimi.it.

[1] TripAdvisor (`http://www.tripadvisor.com/`)
[2] See `http://www.yelp.com/`.
[3] See `http://local.yahoo.com/`.
[4] See `https://maps.google.com/`.

Fig. 1. A picture of a typical side street in Insadong district: the density of restaurants is very high.

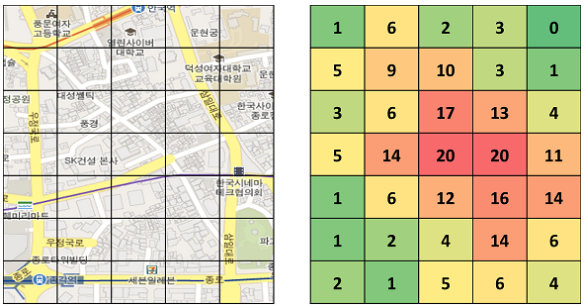

Fig. 2. Restaurants' distribution in the 2 km^2 Insadong district.

(2005) and Qype[5] (2006). All those Web sites assist tourists in gathering travel information by collecting reviews and opinions of travel-related content.

The second wave was fostered by the increasing availability of GPS-enabled smart phones that paved the way for the success of the so-called Location-based Services (LBSs), i.e., mobile and Web applications that provide context-dependent information and functionalities. The best known example of LBS is the *foursquare* social network[6] launched in 2009 that allows users to "check-in" points of interest (POIs) and to share their location with friends. Its rapid success pushed social network actors to add location-based features to their systems. For instance, in 2010, Facebook launched a service to let its user share their position[7], and Twitter launched the *places feature* with foursquare integration.

While the first wave was centred on the user as an individual, the second one is pivoting around communities. For instance, foursquare proposes the number of "check-ins" and the number of unique users as proxies for a POI popularity. The fact that LBSs mediate interactions between communities and POIs, and make them public, calls for *reality mining* [11], i.e., the interpretation of human social behaviour by sensing throughout large communities of individuals.

In this paper we propose BOTTARI,[8] a *reality mining* application that listens to social media (specifically Twitter), and uses deductive and inductive stream reasoning [4] to perceive the collective reputation of POIs with an emphasis on recent ratings. BOTTARI has been experimentally employed to mine the reputation of restaurants in a district of Seoul named Insadong. As data sources we used a manually curated knowledge base about 319 restaurants located in the 2 km^2 district of Insadong and a three year collection of 109,390 tweets that rates Insadong's restaurants. The mined reputation has been used to offer personalized and localized restaurant recommendations to mobile users through an augmented reality Android application [2].

The paper is organized as follows. Section 2 describes the data used in the application. Section 3 presents the BOTTARI app. Section 4 decribes the pluggable system architecture and the components of BOTTARI's back-end. The inductive and deductive stream reasoning techniques that realize BOTTARI recommendations are illustrated in Section 5. Sections 6 and 7 report our experimental results and analyse the scalability of the proposed scenarios with regards to future large scale deployments of BOTTARI. Finally, Section 8 concludes the paper and sketches our future works.

2. Data Collection

In this section, we describe first the Insadong district of Seoul with its restaurants and the manually curated knowledge base that describes those restaurants, and then the automatically collected tweets about the restaurants and their semantic interpretation as restaurants' ratings.

2.1. Insadong's Restaurants

Figure 2 illustrates the Insadong area, a 2 km^2 district with a high density of restaurants: the left side shows the topology of Insadong and the right side

[5] See http://www.qype.com/.
[6] See http://foursquare.com/.
[7] See http://www.facebook.com/about/location/.
[8] In the Korean language, "bottari" is a cloth bundle that carries a person's belongings while traveling. BOTTARI mobile app lets the user "transport" the location-specific knowledge, mined from the local social media, when moving in the geographic space.

shows a heat-map displaying the number of restaurants in a 5-by-7 grid. As the reader can see in Figure 1, in some areas, restaurants are so dense that finding a restaurant entrance (not known in advance) requires both high GPS accuracy and an image of the restaurant.

A description of the Insadong district is contained in most tourist guide books. Mobile apps can be used to perform local search for eating places, but usually information is available only for a small fraction of the existing 319 restaurants. For instance, the popular RoughGuides[9] which have an edition dedicated to South Korea, lists only 8 restaurants in Insadong. A user searching for a restaurant in Insadong using TripAdvisor would find 26 results, with a maximum of 18 reviews; on average there are 2.7 ratings per restaurant.

Gathering from the Web a comprehensive list of the 319 Insadong restaurants is not impossible, but it requires to merge the results of several local search services and dedicated Korean restaurant Web sites and portals. In developing BOTTARI, we performed those tasks and the result is a high-quality geo-referenced knowledge base in which each restaurant is described by 44 attributes (e.g., name, images, position, address, ambiance, specialities, categories, etc.).

2.2. *Tweets about Insadong's Restaurants*

South-Koreans are active Twitter users. Each day, 3.4 millions tweets are posted from Seoul. Saltlux commercially offers business intelligence services based on an analysis of micro-posts. In BOTTARI, Saltlux's proprietary micro-post crawling and opinion mining solution is used. The crawling component collects tweets that either contain a name of one of Insadong's restaurants or tweets that were posted in the Insadong area. A subsequent analysis discards tweets that cannot be related with high precision to one of the restaurants. Finally, the opinion mining component detects if a micro-post expresses a positive, negative or neutral opinion about a particular restaurant.

This component is made of three modules (see Figure 4). The first is a Korean Morphological Analyzer that tries to parse the text of the tweet. If it succeeds, the text is analysed using a rule-based module; a machine learning module based on Support Vector Machines (SVM) [20] is used otherwise.

The crawling and opinion miner component was tuned to guarantee high precision even if this implies

[9] See `http://www.roughguides.com/`.

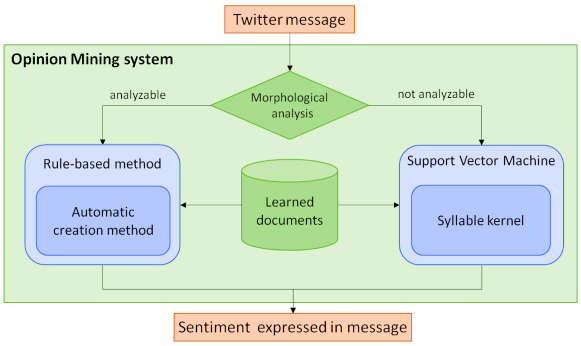

Fig. 4. Internal architecture of BOTTARI's Opinion Miner.

a low recall. In total, 200 million tweets somehow related to Insadong restaurants were gathered in 3 years, from 2008/02/04 to 2010/11/23 (1,023 days). After discarding the tweets that do not relate to any POI, we were able to identify 109,390 tweets produced by 31,369 users that rated 245 restaurants. Table 1 illustrates the statistics of the collected data set.

The temporal distribution of tweets is shown in Figure 3(c). Clearly most tweets (85%) were collected in the last six months. The exponential growing trend of tweets over time is due to different reasons: *a*) the tweets from 2008 were collected one year later, in 2009, and it was difficult to gather those messages because of the "oblivion" in Twitter stores; *b*) the usage of Twitter became mainstream in Korea only in 2009; finally, *c*) the crawling algorithm was changed and improved in April 2010.

			# 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. The number of positive, negative and neutral ratings respectively and the number of users and restaurants involved in each kind of rating. For instance, 10,448 users give 14,404 negative ratings about 181 POIs. Thus, the data matrix representing these negative ratings is a 10,448-by-181 matrix where 14,404 entries are non-zero and the remaining 99.24% entries are zeros (sparsity).

Table 1 shows the number of positive/negative/neutral ratings in terms of the number of involved entities (users and POIs). Based on those statistics we can assert the following characteristics:

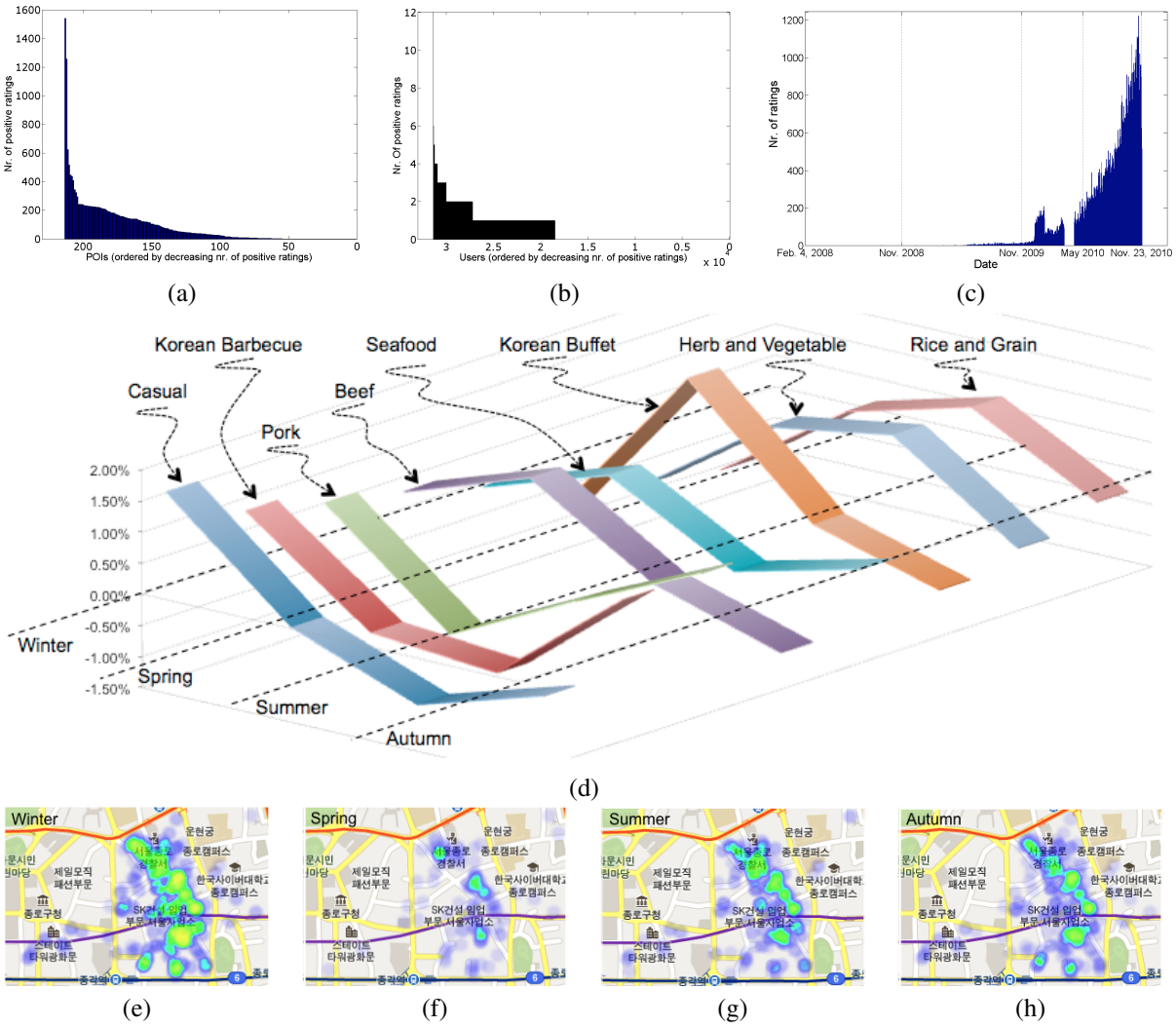

Fig. 3. Dataset statistics: (a) positive ratings by POIs; (b) positive ratings by users; (c) tweets' distribution over time; (d) seasonal effect on restaurants' preferences; and change in the location of top restaurants during (e) Winter, (f) Spring, (g) Summer and (h) Autumn.

- *High sparsity*: The sparsity of the positive ratings is higher than 99.3%, where the sparsity is defined as $sparsity = 1 - \frac{\#Ratings}{\#POIs \times \#Users}$.
- *Incompleteness*: While some POIs have neither positive nor negative ratings, a great number of users only provides either positive or negative ratings. For example, only 41% of users positively rated at least one POI.
- *Multiple ratings*: In BOTTARI, positive, negative and neutral ratings are represented as separate statements $positive(u,p)$, $negative(u,p)$ and $neutral(u,p)$. $positive(u,p) = 1$ means that at least one positive rating was obtained by user u on POI p. If there is no positive rating of user u for POI p we have $positive(u,p) = 0$. The same approach is used for $negative(u,p)$ and $neutral(u,p)$.

Besides the temporal distribution shown in Figure 3(c), statistics about ratings by users and POIs are useful to better understand the data distribution. The distribution of positive ratings over POIs and over users is especially interesting and is plotted in Figure 3(a) and 3(b), respectively. On average, a user gave 1.5 positive ratings and a POI was positively rated by 89 users on average. 31 POIs got more than 200 positive ratings; those POIs can be considered the most liked or the most popular ones.

Finally, let us highlight that the tweets recorded a *seasonal* effect, i.e. people dining in Insadong's restau-

Fig. 5. Some screenshots of the BOTTARI Android application: the AR interface with the four types of recommendations and the possibility to filter by distance is shown in (a); further details about a selected POI are shown in (b) and (c) where the similar restaurants and the map with the POI description are visualized, respectively; finally, the trend over time of user opinions about the POI are graphically drawn in (d).

rants have different habits in different seasons. As Figure 3(d) shows, restaurants categorized as casual, Korean-barbecue and pork in the BOTTARI knowledge base are positively rated more in Autumn/Winter then in Spring/Summer, and symmetrically, those categorized as beef, sea-food, herb&vegetable and rice&grain are more appreciated in Spring/Summer than in Autumn/Winter. This effect can also be visualized in terms of usage patterns with heatmaps (see Figure 3(e-h)) that highlight the location of the top restaurants in the different seasons.

3. BOTTARI

As we stated in Section 1, BOTTARI aims at recommending POIs nearby the location of its mobile users by aggregating the ratings extracted from social media (specifically Twitter).

When a user starts the app, BOTTARI retrieves the manually curated knowledge base of Insadong's restaurants and shows in augmented reality (AR) the restaurants and dining places in the proximity of the user's position (see Figure 5(a)). Given the importance of the distance user from the recommended POIs, demonstrated by several large scale studies [14,7,18], BOTTARI allows the user to filter the results by distance (see the circles at the top-right of Figure 5(a)).

A user can immediately distinguish between restaurants appreciated and not appreciated by the community, since they are indicated by thumb-up and thumb-down icons in Figure 5(a) and 5(b)). Detailed information about each restaurant is available by clicking on the icons. Figure 5.(c) illustrates what a user can see when pressing the Details button (top-right of Figure 5(b)), while Figure 5(d) presents the positive and negative reputation trends in the last 12 months.

Four different types of recommendations are offered to the user (see top-left of Figure 5(a)). The *For me* button returns restaurants that the user may appreciate based on the tweets the user posted in the past about In-

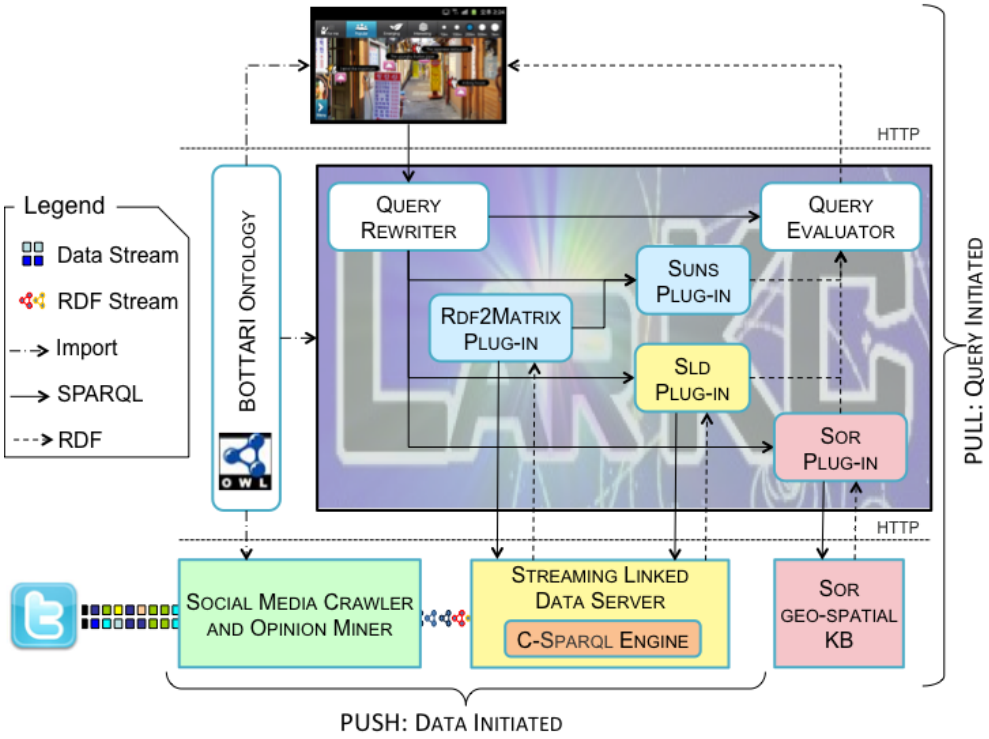

Fig. 6. The Architecture of BOTTARI.

sadong restaurants[10]. The *Popular* button recommends the top restaurant considering the whole 3 years of ratings extracted from the microposts. The *Emerging* button, instead, asks for restaurants top-rated by the community in general in the last three months. Finally, the *Interesting* button returns the restaurants described with a category of interest to the user, e.g., *to tourists*.

4. System Architecture

BOTTARI is an ontology-based information integration systems [17]. It uses the LarKC platform [1] – a pluggable Semantic Web framework for massive distributed incomplete reasoning – to interconnect the various components necessary to perform the recommendations. Figure 6 illustrates the system architecture. The innovative aspect is in integrating data-driven stream reasoning components [4] (namely, the PUSH segment) with more traditional query driven ones (namely, the PULL segment).

[10]Several studies [21,10] show that personal preferences play a more important role in mobile search applications compared to Web searches.

The core of the system is the ontology described in Section 4.1. Tweets are continuously processed by the PUSH segment presented in Section 4.2. When a user of the BOTTARI mobile application pushes one of the recommendation buttons (see upper-left corner of Figure 5(a)), the app sends a SPARQL query expressed in the BOTTARI ontology to LarKC, which then executes the PULL segment as explained in Section 4.3.

4.1. BOTTARI Ontology

The BOTTARI Ontology is an extension of Semantically-Interlinked Online Communities (SIOC) [8] – a popular vocabulary for expressing user-generated content of online community. The BOTTARI ontology takes from SIOC the relation between Twitter users and their tweets, and enriches it by further describing the relation between a tweet and its topic (i.e., the POI cited in the tweet[11]). To this end, the BOTTARI ontology adds the opinion expressed by the user about the topic by means of the `twd:talksAbout` property – and its sub-properties for positive, negative and neutral opinions.

[11]Only very few tweets talk about more than one POI.

Moreover, the BOTTARI ontology adds a geographical perspective by modelling the POIs as *spatial things* (according to the WGS84 vocabulary[12]) and enriching them with specific categorizations (e.g., the ambience describing the atmosphere of a restaurant) and the dynamic count of positive/negative/neutral ratings.

4.2. PUSH Segment

The PUSH segment continuously analyses the stream of microposts extracted from the Web by the SEMANTIC MEDIA CRAWLER AND OPINION MINER (cf. Figure 6). Following the Stream Reasoning [12] approach, the PUSH segment models the information flow as an RDF stream (i.e., a sequence of RDF triples annotated with a non-decreasing timestamp); and the STREAMING LINKED DATA SERVER (SLD SERVER) efficiently processes it "on the fly" by exploiting the temporal order of the data elements in the stream. The SLD SERVER computes the information required by *Popular* and *Emerging* recommendations.

An example of an RDF stream that flows through the BOTTARI PUSH segment is shown in Listing 1. Each RDF triple uses the ontology described above and is annotated with the tweet's timestamp. The notation used here is defined in [3].

```
(<:Giulia :posts _:1 >,             20111012T13:34:41)
(< _:1 :talksPositivelyAbout :r1 >, 20111012T13:34:41)
(<:Leo :posts _:2 >,                20111012T13:37:12)
(< _:2 :talksNegativelyAbout :r1 >, 20111012T13:37:12)
```

Listing 1: Social media RDF stream

SUNS [23,15] – BOTTARI's inductive stream reasoner – is responsible for the *For me* recommendations. It periodically reads the results of the SLD SERVER through the RDF2MATRIX PLUG-IN and applies its scalable machine learning framework to predict unknown but potentially true statements by exploiting regularities in the RDF stream.

4.3. PULL Segment

The PULL Segment works as an ontology-based information-integration system. Each plug-ins involved in generating the recommendations has its own data model and query language. The BOTTARI ontology is used to combine the recommendations generated by the multiple heterogeneous plug-ins. The data model, internally used by each plug-in, is declaratively mapped to the BOTTARI ontology using a global-as-view approach. The QUERY REWRITER uses these mappings to rewrite the queries it receives from the BOTTARI app and issues the rewritten queries to SUNS, SLD and SOR plug-ins. The plug-ins evaluate the respective queries in parallel and send their results to the QUERY EVALUATOR that joins the results and sends them back to the BOTTARI app. Both final and intermediate results are cached by LarKC platform to minimize latency.

The SOR PLUG-IN efficiently evaluates geo-spatial SPARQL queries (e.g., queries that contains geo-spatial functions such as `within_distance`), on SOR – Saltlux's geo-spatial enabled RDF store[13]. This plug-in is used in each type of recommendation, since BOTTARI always filters results by distance (see the circles at the top-right of Figure 5(a)). The *Interesting* recommendations use only this plug-in.

The SUNS PLUG-IN estimate probabilities of unknown statements of the form:

```
<user :talksPositivelyAbout POI>
```

and provide users with a list of POIs ranked by estimated probabilities (the higher the score, the more likely the statement). This plug-in is used in the *For me* recommendations.

The SLD PLUG-IN is used in both the *Popular* and *Emerging* types of recommendation. It relies on the aggregation and windowing mechanisms offered by the SLD SERVER that continuously evaluate a network of C-SPARQL queries. In the *Emerging* case the window is three months long, while in the *Popular* case its spans three years.

5. Recommendation Services in BOTTARI

In this section, we provide more details about the different kinds of recommendations that BOTTARI offers. We start by presenting the *Interesting* recommendations, because they involve only the SOR plug-in. Next, we illustrate the *Popular* and the *Emerging* recommendations which both rely on the SLD and the SOR plug-ins. Finally, we report on the *For me* recommendations that use the SUNS and the SOR plug-ins.

[12] See `http://www.w3.org/2003/01/geo/`.

[13] Cf. `http://bit.ly/saltlux-sor`.

5.1. Semantic Search for Interesting POIs

The *Interesting* recommendation is based on the SPARQL query in Listing 2. It asks for a list of POIs (lines 3 and 4) that belong to one of the category available in BOTTARI, e.g., "attractions of interest to foreign visitors" (line 5), that are located in a given distance from the user's current location (line 6), and that are in front of the user[14] (line 7). The results are ordered by growing distance and limited to the 10 closest ones (lines 8 and 9).

```
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,
                                        45,50,200))
8. ORDER BY :distance(37.5,126.9,?lat,?long)
9. LIMIT 10
```

Listing 2: Query for Interesting recommendations.

Note that *a*) the query highly relies on the geospatial features of the SOR GEO-SPATIAL KB (see Figure 6); *b*) the query returns not only POIs described as `InterestingForForeigners`, but also those described with a sub-category of `InterestingForForeigners`; and *c*) this query involves only the SOR PLUG-IN and requires no rewriting, because the BOTTARI ontology is an extension of the ontology used in SOR GEO-SPATIAL KB.

5.2. Stream Reasoning for Emerging/Popular POIs

The *Popular* recommendations are based on the top-rated POIs across the entire three year dataset. This type of recommendation is a standard base-line in recommendation engines. However, the opinions of users on POIs change over time. For instance, in different seasons Insadong's visitors prefer different categories of restaurants (see the seasonal effect illustrated in Figure 3). The *Emerging* recommendations consider the list of the top positively rated POIs in the last three months, thus aim at recommending POIs considering the seasonal effect.

[14] Filtering the POIs to load only those in front of the user is important, because BOTTARI is an augmented reality application and BOTTARI's users are supposed to look at the world around them through their device.

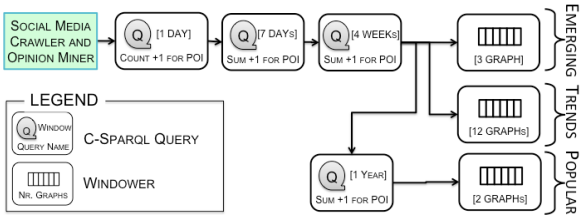

Fig. 7. The network of C-SPARQL that continuously analyses the RDF stream produced by the SOCIAL MEDIA CRAWLER AND OPINION MINER and keeps up to date the data used for the *Emerging* and *Polular* recommendations

Figure 7 illustrates how the SLD SERVER continuously elaborates the RDF stream produced by the SOCIAL MEDIA CRAWLER AND OPINION MINER to compute the aggregated information used for the *Emerging* and *Popular* recommendations. The SLD SERVER continuously runs a network of C-SPARQL queries that consume each other's results and publish their output as Streaming Linked Data [3]. From left to right in the figure, the COUNT +1 FOR POI IN [1 DAY] query counts the positive opinions about a POI for each day. The positive messages per day can be further aggregated over the week (in the SUM +1 FOR POI IN [7 DAYS] query), over the month (in the SUM +1 FOR POI IN [4 WEEKS] query), and over the year (in the SUM +1 FOR POI IN [1 YEAR] query).

Listing 3 shows the leftmost query in Figure 7 that counts the positive opinions on each POI per day; Listing 4 refers to the query that consumes the result stream of the previous one, summing the positive opinions for each POI over the last week.

```
REGISTER QUERY Count +1 For POI AS
CONSTRUCT { ?o twd:numberOfPositiveTweets ?n . }
FROM STREAM <#bottari> [RANGE 1d STEP 1d]
WHERE { { SELECT DISTINCT ?o (count(?s) as ?n)
          WHERE { ?s twd:talksPositivelyAbout ?o }
          GROUP BY ?o } }
```

Listing 3: Query to count positive opinions per day.

```
REGISTER QUERY Sum +1 For POIs AS
CONSTRUCT { ?s twd:numberOfPositiveTweetsInWeek ?n . }
FROM STREAM <#Count+1forPOI> [RANGE 7d STEP 7d]
WHERE { { SELECT ?s (sum(?o) as ?n)
          WHERE { ?s twd:numberOfPositiveTweets ?o . }
          GROUP BY ?s } }
```

Listing 4: Query to sum the positive opinions for each POI over the last week.

The results of each query are published as linked data by the WINDOWERS. The WINDOWERS are used to answer the BOTTARI query for the *Emerging* recommendations; the bottom-right one in Figure 7 is used to compute the *Popular* recommendations. The trend lines illustrated in Figure 5(d) are drawn using the 12 most recent answers of the SUM +1 FOR POI IN [4 WEEKS] published as Linked Data by a WINDOWER.

When BOTTARI clients require *Emerging* recommendations, they issues to the LarKC platform the SPARQL query in Listing 5.

```
1. SELECT ?poi ?name ?lat ?long
2. WHERE {
3. ?poi a ns:NamedPlace ;  ns:name ?name ;
4.      geo:lat ?lat ;  geo:long ?long .
5. FILTER(:within_distance(37.5,126.9,?lat,?long,200))
6. FILTER(:dest_point_viewing(37.5,126.9,?lat,?long,
                                        45,50,200))
7.  ?poi twd:numberOfPositiveTweetsInMonth ?n .
8. ORDER BY DESC(?n/:distance(37.5,126.9,?lat,?long))
9. LIMIT 10
```

Listing 5: Query for *Emerging* recommendations.

Lines 3 to 6 are sent by the QUERY REWRITER to the SOR PLUG-IN. Line 7 is rewritten in a query that runs against the latest result of the SUM +1 FOR POI IN [4 WEEKS] C-SPARQL query published as linked data by the SLD SERVER. The QUERY EVALUATOR receives the list of POIs returned by the two plug-ins and aggregates the ranking in accordance with the function at line 8, i.e., trendy and close-by restaurants first.

5.3. Inductive Stream Reasoning to get For-me POIs

Finally, BOTTARI can provide personalized recommendations using SUNS – the inductive stream reasoning solutions investigated in [23,15]. SUNS employs matrix factorization approaches [22,6] and tensor factorization approaches [19,24] to identify regularities in the data and to predict unknown, but potentially true, statements. In particular, BOTTARI benefits from the recently introduced modularized approach [24] that permits the easy integration of contextual information, such as temporal and sequence information.

Figure 8 sketches the architecture of the inductive reasoner. At training time, the RDF2MATRIX PLUG-IN transforms data and knowledge from RDF statements into the data matrix (required as input by machine learning approaches including SUNS), while the SUNS PLUG-IN trains the models. If necessary, external machine learning libraries can be invoked. This training process occurs as a batch process and the trained models are persistently stored internally. During the system startup, the models are loaded in memory so that at runtime the predictions can be carried out efficiently. Based on the models, the SUNS PLUG-IN estimates the truth value of unknown statements of interest. The SUNS PLUG-IN executes the training off-line and the estimation on-line. To facilitate the usability of the plug-ins, a default configuration is available. Machine learning experts can refine the setting of the plug-ins by modifying the default configuration.

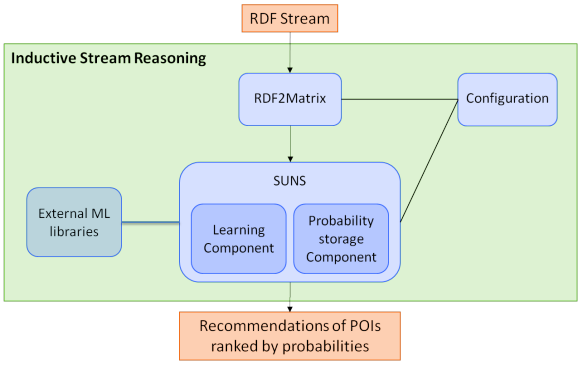

Fig. 8. Architecture of inductive stream reasoning

In BOTTARI, the RDF2MATRIX PLUG-IN reads the RDF streams about the opinions (see Table 1) provided by the SLD SERVER and transforms them into a data matrix, while the SUNS PLUG-IN learns the users' preferences and predicts the probabilities that can be then queried for the *For me* 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. FILTER(:within_distance(37.5,126.9,?lat,?long,200))
6. FILTER(:dest_point_viewing(37.5,126.9,?lat,?long,
                                        45,50,200))
7. { :user sioc:creator_of ?tweet
8.   ?tweet twd:talksPositivelyAbout ?poi .
9.   WITH PROBABILITY ?prob
10.  ENSURE PROBABILITY [0.5..1)  }
11. ORDER BY
       DESC(?prob/:distance(37.5,126.9,?lat,?long))
12. LIMIT 10
```

Listing 6: Query for "For me" recommendations.

When BOTTARI clients request *For me* recommendations, they issue the SPARQL query in Listing 6. Lines 7 to 10 makes use of SPARQL with probability [15]. The triple pattern at lines 7 and 8 matches POIs that a particular user might positively talk about. The `WITH PROBABILITY` clause at line 9 extends SPARQL by letting it query probabilities. The variable `?prob` may assume values between 0 and 1, where the value 1 refers to those POIs the user already positively rated. The `ENSURE PROBABILITY` clause at line 10 accepts only those solutions whose estimated probabilities are larger or equal to, in this example, 0.5.

The query in Listing 6 is rewritten by the QUERY REWRITER and produces the query for the SUNS PLUG-IN illustrated in Listing 7, the query for the SOR PLUG-IN (cf. Section 5.1) and the query for the QUERY EVALUATOR illustrated in Listing 9.

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

Listing 7: The query in Listing 6 as rewritten for the SUNS PLUG-IN.

Notably, in Listing 7 the `WHERE` clause (lines 6–9) corresponds to lines 7–10 in Listing 6. The peculiarity of the query in Listing 7 is the `CONSTRUCT` clause. It allows to embed the probability in the query results processed by the SUNS PLUG-IN without using annotations and reification. For instance, the results of the query in Listing 7 can be:

```
:user ns:mayTalkPositively [
     ns:about :TheVolga ;
     ns:withProbability  "0.8" ], [
     ns:about :TheVegetarian ;
     ns:withProbability  "0.6" ] .
```

Listing 8: Sample results of the query in Listing 7.

Note that the rewritten query for the QUERY EVALUATOR in Listing 9 at lines 6–8 assumes the presence of the triples generated by the SUNS PLUG-IN with Listing 7 query.

```
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:distance ?distance .
6. :user twd:mayTalkPositively [
7.     ns:about ?poi ;
8.     ns:withProbability  ?prob ]
9. ORDER BY DESC(?prob/?distance))
10. LIMIT 10
```

Listing 9: The query in Listing 6 as rewritten for the QUERY EVALUATOR.

Wrapping up, the SUNS PLUG-IN provides a generic way for probabilistic materialization. The predicted RDF triples are maintained in a compact internal representation and are serialized as RDF triples on demand. In BOTTARI, the SUNS PLUG-IN is applied to answer the ad-hoc query "Which POIs could be recommended for this particular user?". When a BOTTARI client issues the SPARQL query for the *For me* recommendations, the SUNS PLUG-IN is utilized to estimate the probability that a user might like a POI, based on the (known) opinions that the user expressed about all other POIs and based on the (observed) opinions that all other users expressed about all POIs including that one. In this sense, we provide a personalized and collective recommendation engine suggesting users the most interesting POIs with respect to their preferences.

6. Evaluation

We tested BOTTARI using the data set described in Section 2. The goal was to evaluate the quality and the efficacy of recommendations.

6.1. Baselines

In the evaluation, we applied three baselines: random guess (Random), k-nearest neighbour (KNNItem) and the most liked items (MostLiked).

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 POIs given by all users for $i, j \in$

$\{1, \ldots, P\}$ (being P the total number of POIs), 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 POIs.

The *MostLiked* baseline takes into account all positive ratings at all times. It recommends the most positively rated POIs to every user. This baseline was realized by the SLD SERVER as explained in Section 5.2.

6.2. Metrics

We used two evaluation metrics: accuracy at the top N POIs (Acc@N) and normalized discounted cumulative gain (nDCG). For both evaluation measures, we averaged the scores across all ranking lists (a list per user).

Acc@N is defined as the number of true positives in the top N recommendations. In our case, since we have just one test data point per user, Acc@N is either equal to one or zero.

nDCG is calculated by summing all the gains in the rank list R with a log discount factor as $\text{nDCG}(R) = \frac{1}{Z} \sum_k \frac{2^{r(k)}-1}{\log(1+k)}$, where $r(k)$ denotes the target label for the k-th ranked item in R, and r is chosen so that a perfect ranking obtains value 1. The normalization term Z is the DCG value of the ideal rank. To focus more on the top-ranked items, we also consider the *nDCG@n* which only counts the top n items in the rank list for a user. In our evaluation R is the rank list of POIs and we report nDCG@all.

nDCG [16] is a common measure in information retrieval and is used to evaluate the quality of the returned ranked results for a given query. It especially emphasizes the very top positions in R due to the log function on the position k, which means that the nDCG score increases much more when the position of a test item is lifted from the second position to the first one, rather than when its position is improved from 100 to 99. In our case, with only one test rating (correct answer) for each user, nDCG being 1 means that the test POI is ranked as the top one in the list of POIs, while an extremely low nDCG value indicates that the test POI is placed at bottom of the list.[15]

	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

6.3. Settings

The evaluation was performed in three different settings:

- Setting 1: We randomly withheld one positive rating for each user and treated it as a test data point. We trained the models using the remaining ratings and then estimated the values of the test data points. This corresponds to the common method of splitting the data into a training and a test subsets. We repeated this data split five times.
- Setting 2: We withheld the newest rating for each user as test data. As an additional baseline TIMEWINDOW was introduced which corresponds to MOSTLIKED with different time frames: 1 day, 2 days, 7 days, 30 days, 90 days and 180 days[16]. To this end, we removed the older ratings. Table 2 shows the number of ratings that remained for training.
- Setting 3: motivated by the good performances of the TIMEWINDOW baseline, we further investigated the seasonal effect by introducing the SEASON approach and repeating the experiments in Settings 1 and 2.

We implemented the baselines, SUNS and the evaluation methods in Matlab. We ran the evaluation on a laptop with Windows XP, CPU 2.1 GHz and 3.24 GB RAM, which corresponds to a 80 €/month share in a cloud environment[17].

[15] For example, in our application we define $r(\cdot) = \{0, 1\}$ without loss of generality. When *for every user* the test POI is the top one in the ranking list of recommendations, we obtain nDCG value 1; for the second place, we obtain 0.63; and for the third, fourth and fifth place, we have 0.5, 0.43 and 0.39 respectively.

[16] This baseline also is evaluated using the network of C-SPARQL queries continuously executed by the SLD SERVER as illustrated in Section 5.2.

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

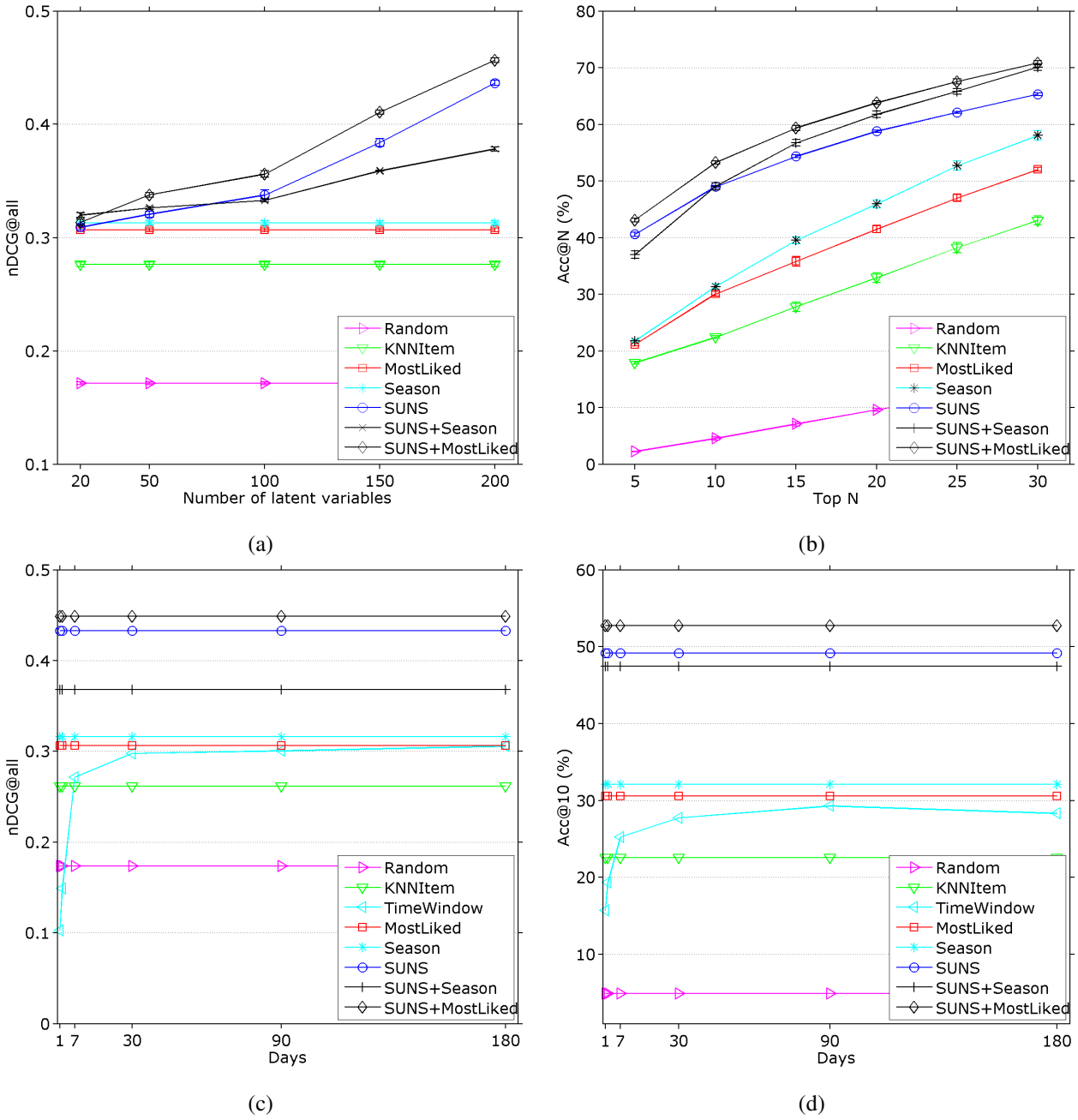

Fig. 9. A comparative evaluation of the different types of recommendation offered by BOTTARI against several baselines.

6.4. Results

Figure 9 shows the results that we obtained in Settings 1 and 3. In Figure 9(a) the nDCG scores of the tested methods are plotted against the number of latent variables. Since the baselines are independent of this number, they produced three horizontal lines for Random, MOSTLIKED and KNNItem. For the SEASON approach the latent variables are not influential either. We evaluated SUNS with 20, 50, 100, 150 and 200 latent variables.

As expected, Random was the worst. MOSTLIKED was slightly better than KNNItem. This might be due to the "bandwagon effect"[18] that exists in many social communities. Interestingly, the SEASON approach performed slightly better than MOSTLIKED, but its combination with SUNS did not bring any improvement when compared with SUNS only. With 20 latent variables, the SEASON approach shows the best performances. SUNS significantly outperformed all baselines after the number of latent variables exceeded 100.

SUNS appeared 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 with little machine learning expertise. The best ranking overall was produced by the combination of SUNS and MOSTLIKED. These results confirm the idea presented in [4] that a combined approach of deductive and inductive stream reasoning works best.

Figure 9(b) shows the accuracy[19] of the top N POIs, where $N = \{5, 10, 15, 20, 25, 30\}$. The quality of the recommendations made by SEASONS and by SUNS was much higher than that of all baseline recommendation types. The combination of SUNS and MOSTLIKED generated the overall best recommendations. Differently from the case of nDCG score, the combination with SUNS improved SEASONS to a great extent and it was very close to the combination of SUNS and MOSTLIKED at top 30.

Figure 9(c) and 9(d) shows the results obtained in Settings 2 and 3. As mentioned beforehand, we varied the size of the time window from 1 day to 180 days. Figure 9(c) plots the nDCG scores, while Figure 9(d) plots the accuracy at top 10. Naturally, only TIMEWINDOW is sensitive to the time axis and approaches the performance of MOSTLIKED with a time window of more than 90 days. This fact tells us that keeping a window of 90 days is nearly as effective as keeping the full history. The other approaches behave in a similar way to Figure 9(a) and 9(b).

We observe that the resulting accuracy values are in general low and that the highest Acc@10 achieved by the combination of SUNS and MOSTLIKED reaches slightly more than 50%. This fact indicates that the recommendation on the POI data set is indeed a tough task. Looking at the top 10 recommendations, only 50% users can find restaurants that they are really interested in. In the data set (described in Section 2.2) every user has positively rated only 1.5 restaurants on average. A typical situation is that a couple of active users contribute a large amount of ratings, while many users own very few ratings. For the latter user group it is difficult to provide recommendations due to the very limited information on their preferences.

7. Scalability

In the evaluation, BOTTARI displayed excellent performance. Training SUNS took approximately 86 seconds with 200 latent variables. Recommendation of POIs for a user using the three parallel plug-ins costs on average less than 10 milliseconds. Internal studies have confirmed that, by exploiting sparsity, SUNS computational requirements scales linearly with the number of *known* ratings. Therefore, the only potential bottleneck of BOTTARI is the PUSH segment. In this section, we report the stress tests we conducted on the SLD SERVER to assess the scalability of the system in future scenarios, when BOTTARI will be deployed at larger scales (e.g., the entire Korea).

7.1. Methodology

An RDF stream, as explained in Section 4.2, is a sequence of triples annotated with a non-decreasing timestamp. The C-SPARQL query network illustrated in Figure 7 uses only logical tumbling windows[20], i.e., each query is evaluated when the difference between

[18] Bandwagon effect is a form of groupthink known in behavioral science and also a common phenomenon in the daily life: people follow majorities without caring for evidence and their individual preferences.

[19] Note, the standard errors shown by the error bars are so small that they lie within the shapes representing each approach.

[20] Interested readers can learn more about C-SPARQL windowing in Section 2.2 of [5].

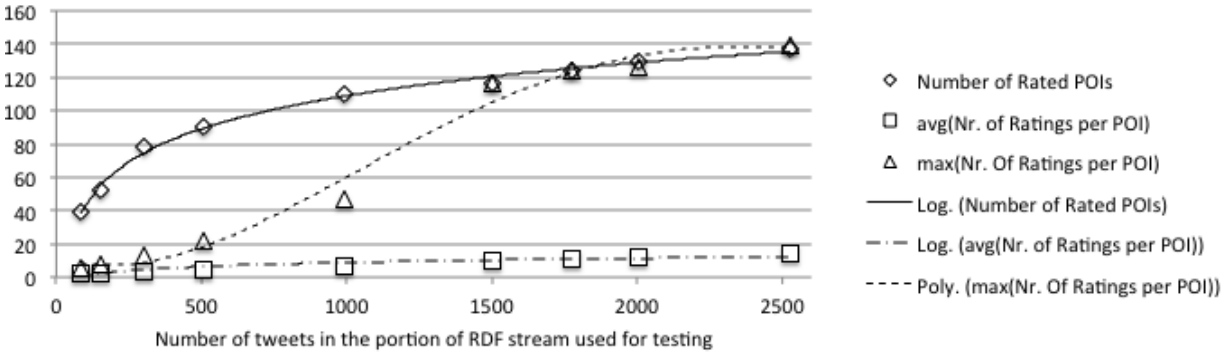

Fig. 10. Settings for SLD SERVER scalability tests

the timestamp of the most recent triple and that of the oldest triple is equal to or greater than the window size. This means that the SLD SERVER can be stressed with a variable speed test without the risk to alter the semantics of the C-SPARQL query network. In other words, the RDF stream can be replayed at a faster rated, stress-testing the SLD SERVER, but without altering the results of the queries. As in publish-subscribe systems [13], the most appropriate quantity to be measured is the *input throughput*, i.e., the ability to consume triples as inputs. It is computed as follows:

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

We measured input throughput by sending to the SLD SERVER a recorded portion of the RDF stream and by measuring the time required to process it, i.e., by computing all the answers to all the queries in the network for this portion of RDF stream.

To improve confidence, we repeated each experiment for 30 minutes and we measured average, minimum and maximum time required to process the portion of the RDF stream.

7.2. Settings

The stress test was conducted in nine different settings using a growing number of tweets in the corresponding segment of the RDF stream presented to BOTTARI. The first settings is a portion of the RDF stream recorded between May, 1^{st} and May, 3^{rd}, 2010. The ninth contains 45 days of tweets recorded between May, 1^{st} and June, 15^{th}, 2010. Figure 10 shows the characteristics of those nine settings. The number of rated POIs grows logarithmically in the number of tweets in the portion of RDF Stream. The same logarithmic dependency exists between the number of tweets and average number of ratings per POI. The maximum number of ratings per POI, instead, has an *S* shape. For small windows (e.g., the first setting containing only three days of tweets) the maximum number of rating per POI is a small fraction of the number of rated POIs, while for large windows (e.g., the sixth setting containing one month of tweets) it tends to become proportional to the number of rated POIs. This happens because the number of positive ratings per POI is distributed according to a power law (see Figure 3(a)).

The experiments were conducted on a laptop with CPU 2.2 GHz and 4 GB RAM, which corresponds to a 80 €/month share in a cloud environment[21].

7.3. Results

The results of the stress tests are illustrated in Figure 11. In the first six settings, the SLD SERVER keeps the pace of the growing input rate of tweets per second. The average latency remains stable at around 2 seconds and consequently the input-throughput linearly increases with the number of tweets in the segment of recorded RDF stream used in the settings. In the first setting, the segment of recorded RDF stream contains 85 tweets recorded in 3 days and the input throughput is 55 tweets per second, whereas in the sixth setting the segment of RDF streams contains 1.501 tweets recorded in 1 month (i.e., the data required by the *Emerging* recommendations) and the input throughput is 700 tweets per second.

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

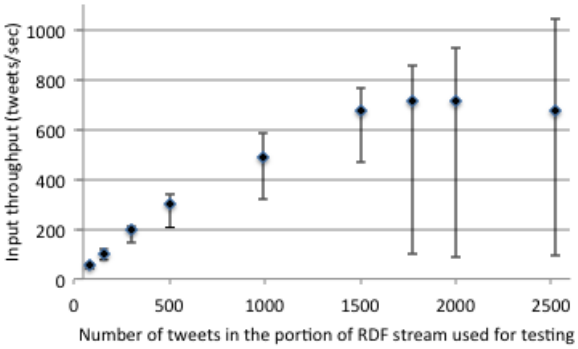

Fig. 11. The results of the scalability tests performed on SLD SERVER

From the seventh setting on, the SLD SERVER is no longer able to keep the pace. The average input throughput remains stable on 700 tweets per second, while the minimum throughput drops to 100 tweets per second (i.e., the maximum latency of the system grows to 25 seconds). This is normal in Data Stream Management Systems (DSMS): the system keeps the pace of the input data up to the moment where all computational resources are saturated. If pick rates above this saturation point are expected, queues are used to temporarily store the incoming data.

7.4. Interpretation of results

The most straightforward interpretation, that the SLD SERVER would be able to support only 700 tweets per second for Insadong, is erroneous. The correct interpretation, instead, is that the SLD SERVER could support the computation needed for the *Emerging* recommendation for tens of thousands[22] of areas like Insadong. This is under the following assumptions: these areas contain hundreds of POIs; POIs are discussed in a thousand of tweets per day; on average each POI is rated ten times per month; at least half of the POIs are rated once per month; and the maximum number of positive rates per month per POI is proportional to the number of POIs. The results show the possibility to deploy BOTTARI across the entire Korea where thousands of areas like Insadong can be located.

[22]The exact number is the proportion between the number of tweets per day about Insadong restaurant and the maximum throughput in tweets per day, which is 50,400 areas like Insadong. In our most recent crawls (see Figure 3(c)) 1,200 tweets per day was posted about Insadong's restaurants, SLD throughput of 700 tweets per second corresponds to 60,480,000 tweets per day, and, thus, the proportion between 60,480,000 and 1,200 is 50,400.

8. Conclusions and Future Work

In the last few years, we have been witnessing the increasing popularity and success of Location-based Services (LBS), especially of those with a Social Networking flavor. Typical services bring a wide range of useful information about tourist attractions, local businesses and points of interest (POIs) in the physical world. Although these services are enormously popular, users still suffer from a number of shortcomings. The overwhelming information flow from those channels often confuses users; it is also very difficult to distinguish between a fair personal opinion and a malicious or opportunistic advice.

In this paper, we presented our work to design and develop BOTTARI, a location-based Android application for mobile users, that recommends POIs in the Insadong district of Seoul by making use of the rich and collective knowledge obtained by mining the urban reality as sensed via micropost streams. BOTTARI won the first prize at the 9th Semantic Web Challenge[23].

From the user's point of view, BOTTARI is an augmented reality application whose interface is designed to be intuitive and easy to use, hiding the complexity of the technology behind a simple interaction. BOTTARI is an effective business analysis and market scanning tool; it also promises to be more effective than guide books and Web 2.0 travel review sites that often suffer from very low recall.

BOTTARI combines location-specific static information about POIs in Insadong (restaurants' description) with dynamic data coming from public micropost streams. Our dataset thus includes heterogeneous information from real sources at a real scale; we employed Semantic Web technologies to allow for a seamless integration of heterogeneous components within an ontology-based information access framework: both POI descriptions and microposts are represented and stored in RDF and described by OWL ontologies; in the data processing, SPARQL and its extensions are heavily used.

BOTTARI's effectiveness and efficiency is the result of an interdisciplinary approach. The data processing to provide POIs recommendations is based on a novel mix of technologies: besides a more traditional location-based information retrieval, we are applying opinion mining and stream reasoning (both deductive and inductive) to the micropost stream to obtain a full reality mining result.

[23]Cf. `http://challenge.semanticweb.org/2011/`.

The presented evaluation demonstrates that our results are very good in terms of efficiency and quality of the produced recommendations and displays a significant better performance with respect to the baselines. Moreover, our techniques demonstrate the capability to efficiently deal with the data scale of social media; indeed, BOTTARI is able to process a significantly large flow of microposts.

The efficacy of recommendations depends on BOTTARI's ability to keep into consideration the user context, in terms of time, location and feeling. It is worth noting, however, that, in general, the efficiency of recommender systems decreases when the user context increases, while, in BOTTARI, the window-based approach, offered by our stream reasoning, allows for deciding to forget past information, thus offering a way to find the most appropriate trade-off between efficacy and efficiency.

Even if BOTTARI is currently just a research prototype, its potential as a commercial product is clear and encouraging and Saltlux decided to continue the BOTTARI development for its Korean customers.

Our future work will be devoted to identifying and recommending POI "mavens" as described in [9]; we also intend to study different strategies to aggregate positive and negative ratings, beyond the straightforward approach we are currently adopting. The possibility to integrate other social media sources, especially those with a location-based flavour (e.g., foursquare), is also under investigation. Finally, we would like to better leverage the promises of a Linked Data-based approach: while in its current implementation, BOTTARI adopts Linked Data only for streaming data, we look forward to integrating BOTTARI with LOD sources: currently, the open linked datasets available turned out to be of too low quality for effective usage.

Acknowledgments

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

References

[1] Assel, M., Cheptsov, A., Gallizo, G., Celino, I., Dell'Aglio, D., Bradesko, L., Witbrock, M., & Della Valle, E. (2011). Large Knowledge Collider: a service-oriented platform for large-scale semantic reasoning. In R. Akerkar, editor, *WIMS*, page 41. ACM.

[2] Balduini, M., Celino, I., Dell'Aglio, D., Della Valle, E., Huang, Y., il Lee, T. K., Kim, S.-H., & Tresp, V. (2012). BOTTARI: An augmented reality mobile application to deliver personalized and location-based recommendations by continuous analysis of social media streams. *J. Web Sem.*, **16**, 33–41.

[3] Barbieri, D. F. & Della Valle, E. (2010). A Proposal for Publishing Data Streams as Linked Data - A Position Paper. In C. Bizer, T. Heath, T. Berners-Lee, and M. Hausenblas, editors, *LDOW*, volume 628 of *CEUR Workshop Proceedings*. CEUR-WS.org.

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

[5] Barbieri, D. F., Braga, D., Ceri, S., Della Valle, E., & Grossniklaus, M. (2010b). Querying RDF streams with C-SPARQL. *SIGMOD Record*, **39**(1), 20–26.

[6] Bell, R. M. & Koren, Y. (2007). Lessons from the Netflix prize challenge. *SIGKDD Explorations*, **9**(2), 75–79.

[7] Berberich, K., König, A. C., Lymberopoulos, D., & Zhao, P. (2011). Improving local search ranking through external logs. In W.-Y. Ma, J.-Y. Nie, R. A. Baeza-Yates, T.-S. Chua, and W. B. Croft, editors, *SIGIR*, pages 785–794. ACM.

[8] Berrueta, D., Brickley, D., Decker, S., Fernández, S., Görn, C., Harth, A., Heath, T., Idehen, K., Kjernsmo, K., Miles, A., Passant, A., Polleres, A., Polo, L., & Sintek, M. (2007). SIOC Core Ontology Specification. W3C Member Submission, W3C.

[9] Celino, I., Dell'Aglio, D., Della Valle, E., Huang, Y., il Lee, T. K., Kim, S.-H., & Tresp, V. (2011). Towards BOTTARI: Using Stream Reasoning to Make Sense of Location-Based Microposts. In R. Garcia-Castro, D. Fensel, and G. Antoniou, editors, *ESWC Workshops*, volume 7117 of *Lecture Notes in Computer Science*, pages 80–87. Springer.

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

[11] Eagle, N. & Pentland, A. (2006). Reality mining: sensing complex social systems. *Personal and Ubiquitous Computing*, **10**(4), 255–268.

[12] Emanuele Della Valle and Stefano Ceri and Frank van Harmelen and Dieter Fensel (2009). It's a Streaming World! Reasoning upon Rapidly Changing Information. *IEEE Intelligent Systems*, **24**(6), 83–89.

[13] Fabret, F., Jacobsen, H.-A., Llirbat, F., Pereira, J., Ross, K. A., & Shasha, D. (2001). Filtering Algorithms and Implementation for Very Fast Publish/Subscribe. In *SIGMOD Conference*, pages 115–126.

[14] Froehlich, J., Chen, M. Y., Smith, I. E., & Potter, F. (2006). Voting with Your Feet: An Investigative Study of the Relationship Between Place Visit Behavior and Preference. In P. Dourish and A. Friday, editors, *Ubicomp*, volume 4206 of *Lecture Notes in Computer Science*, pages 333–350. Springer.

[15] Huang, Y., Tresp, V., Bundschus, M., Rettinger, A., & Kriegel, H.-P. (2010). Multivariate Prediction for Learning on the Semantic Web. In P. Frasconi and F. A. Lisi, editors, *ILP*, volume 6489 of *Lecture Notes in Computer Science*, pages 92–104. Springer.

[16] Järvelin, K. & Kekäläinen, J. (2000). IR evaluation methods for retrieving highly relevant documents. In *SIGIR*, pages 41–48.

[17] Lenzerini, M. (2002). Data Integration: A Theoretical Perspective. In L. Popa, editor, *PODS*, pages 233–246. ACM.

[18] Lymberopoulos, D., Zhao, P., König, A. C., Berberich, K., & Liu, J. (2011). Location-aware click prediction in mobile local search. In C. Macdonald, I. Ounis, and I. Ruthven, editors, *CIKM*, pages 413–422. ACM.

[19] Rendle, S., Freudenthaler, C., & Schmidt-Thieme, L. (2010). Factorizing personalized Markov chains for next-basket recommendation. In M. Rappa, P. Jones, J. Freire, and S. Chakrabarti, editors, *WWW*, pages 811–820. ACM.

[20] Schölkopf, B. & Burges, C. J. (1999). *Advances in Kernel Methods: Support Vector Learning*. MIT press.

[21] Shen, X., Tan, B., & Zhai, C. (2005). Context-sensitive information retrieval using implicit feedback. In R. A. Baeza-Yates, N. Ziviani, G. Marchionini, A. Moffat, and J. Tait, editors, *SIGIR*, pages 43–50. ACM.

[22] Takács, G., Pilászy, I., Németh, B., & Tikk, D. (2007). Major components of the gravity recommendation system. *SIGKDD Explorations*, **9**(2), 80–83.

[23] Tresp, V., Huang, Y., Bundschus, M., & Rettinger, A. (2009). Materializing and querying learned knowledge. In *Proc. of the 1st International Workshop on Inductive Reasoning and Machine Learning for the Semantic Web 2009*.

[24] Tresp, V., Huang, Y., Jiang, X., & Rettinger, A. (2011). Graphical Models for Relations - Modeling Relational Context. In J. Filipe and A. L. N. Fred, editors, *KDIR*, pages 114–120. SciTePress.