This article has been accepted for publication in IEEE Intelligent Systems but has not yet been fully edited.
Some content may change prior to final publication.

Deductive and Inductive Stream Reasoning for Semantic Social Media Analytics

Davide Barbieri[1], Daniele Braga[1], Stefano Ceri[1], Emanuele Della Valle[1], Yi Huang[2], Volker Tresp[2], Achim Rettinger[3], and Hendrik Wermser[4]

[1] Dip. di Elettronica e Informazione, Politecnico di Milano, Milano, Italy
[2] Siemens AG, Corporate Technology, Munich, Germany
[3] Karlsruhe Institute of Technology, Karlsruhe, Germany
[4] Technical University of Munich, Munich, Germany

Abstract. Knowledge evolution is a major challenge in knowledge management. When knowledge evolution is conveyed in the form of data streams, a combined approach of deductive and inductive reasoning can leverage the clear separation between the *evolving* (streaming) and the *static* parts of the knowledge at the conceptual and technological level. In particular, the notion of *RDF streams* is the "glue" for the interoperability of inductive and deductive reasoning techniques with methods and systems for processing data streams. In this paper, we show our combined approach applied to social network analysis, we give experimental evidence of good performances, and demonstrate the effectiveness of the approach for extracting trends from micro-blogs and feeds.

Keywords: stream reasoning, online learning, social media analytics, deductive reasoning, inductive reasoning, RDF streams, SPARQL, C-SPARQL

1 Introduction

What are the hottest topics under discussion on Twitter? Which topics have my close friends discussed in the last hour? Which movie is my friend most likely to watch next? Which Tuscany red wine should I recommend to them? The information required to answer these queries is becoming available on the Web, as many popular social networks publish micro-blogs and feeds. This trend is often referred to as the "Twitter phenomenon". These feeds are "continuous" flows of information where recent items are typically more relevant than older ones, properly representing a stream. However, their interpretation requires rich background knowledge to fulfill meaningful reasoning tasks, beyond standard stream processing capabilities.

Stream processing has been studied by both, the database [dsmsbook] and data mining communities. Specialized Data Stream Management Systems (DSMS) are available on the market and DSMS features are appearing in major database products, such as Oracle and DB2. Online stream mining is applied in many contexts, e.g. to computer network traffic for intrusion detection, to Web searches for online recommendations [SuYYT10], and to sensor data for automated real-time decision making. These applications represent a *paradigm change* in information processing techniques, as data streams are processed on the fly, without being stored, and processing units produce their results without explicit invocation.

DSMS are designed to process real-time parallel queries over possibly bursty data, but they cannot perform reasoning tasks as complex as those required to answer the queries raised in the first paragraph. Understanding and interpreting the Twitter phenomenon (and many other data streams) requires

Digital Object Indentifier 10.1109/MIS.2010.111 0885-9000/$26.00 2010 IEEE

This article has been accepted for publication in IEEE Intelligent Systems but has not yet been fully edited.
Some content may change prior to final publication.

connecting quick and concise streams, representing changes occurring in the real world, to rich background knowledge bases. This "join" is crucial, for instance, for (1) understanding the topics discussed in the streams with the help of topic taxonomies, (2) recommending most attractive movies to particular user profiles and (3) predicting future behaviors based on the analysis of past behaviors (movie attendance). These examples show the need for connecting data stream processing techniques with reasoning methods, of both inductive and deductive nature, in order to support social media analytics. We believe that this is a good example of how a novel and high-impact research area, that we called **Stream Reasoning** [IEEE-IS-SR], enables the merging of data streams and rich background knowledge.

Extending reasoning methods to support changing knowledge is a known challenge for the reasoning community. In *deductive* reasoning, various methods exist to revise beliefs based on recent information. In *inductive* reasoning, a body of research in data mining and machine learning already supports online data analysis. However, little work has been done on the application of machine learning to streams as rich and structured as those considered here. We believe that data streams are an ideal model for changes occurring in the real world, as well as a suitable means to delimit the source and the nature of change, clearly separating the static part of knowledge from that part of knowledge that changes in real time.

This separation is both conceptual and technological, and allows the use of existing systems for data stream management on one hand and for inductive or deductive reasoning on the other. Also, data streams easily allow a novel interconnection of deductive and inductive reasoning. In our approach, the "glue" for this interconnection is the notion of *RDF streams*, together with an extension of the SPARQL language for continuous queries.

The next section introduces the notion of *stream reasoning* and briefly describes our previous work on the development of a stream reasoning platform. Section 3 describes the data streams generated by the social network Glue[1], used for our experiments. Section 4 shows examples of applications of stream reasoning to the running example. An evaluation of our approach is presented in Section 5, while Section 6 draws some conclusions and provides an outlook on future work.

2 Stream Reasoning

Deductive and inductive stream reasoning, described next in more detail, extend the following notion.

> **Stream Reasoning**: reasoning in real time on huge and possibly noisy data streams, to support a large number of concurrent decision processes.

We now characterize stream reasoning with respect to the notions of *streams*, *windows*, and *continuous processing*, three key concepts in stream processing [babu01continuous].

- **Streams** Data streams are unbounded sequences of time-varying data elements that form a "continuous" flow of information. Recent elements are more relevant as they describe the current state of a dynamic system. Being RDF the data interchange format for reasoners, we move from the notion of *RDF streams* as the fuel for stream reasoning. RDF streams are defined as ordered sequences of pairs, made of RDF triples and their timestamps T_i:

  $(<subj_i , pred_i , obj_i>, T_i)$
  $(<subj_{i+1} , pred_{i+1} , obj_{i+1}>, T_{i+1})$

[1] http://www.getglue.com

Digital Object Indentifier 10.1109/MIS.2010.111 0885-9000/$26.00 2010 IEEE

This article has been accepted for publication in IEEE Intelligent Systems but has not yet been fully edited.
Some content may change prior to final publication.

Timestamps can be considered as annotations of RDF triples. They are monotonically non-decreasing in the stream ($T_i \leq T_{i+1}$), and adjacent triples may have the same timestamp, if "occurring" at the same time.

- **Windows** Traditional reasoning problems assume that all the available information should be considered for solving a problem. Stream reasoning, instead, restricts processing to a certain window of concern, focusing on a subset of recent statements in the stream, while previous statements are ignored. However, the cumulative effect of past windows (processed in the past) and present windows can be taken into account.

- **Continuous Processing** Traditional reasoning approaches have well-defined beginnings and ends for reasoning tasks, respectively when tasks are presented to the reasoner and when results are delivered. Stream reasoning moves from this processing model to a continuous model, where tasks are registered and continuously evaluated reasoner against flowing data.

We are pushing our Stream Reasoning vision within the LarKC[2] project [ICSC08], whose main goal is to develop a platform for reasoning on massive heterogeneous information such as social media data. The platform has a pluggable architecture to exploit techniques and heuristics from diverse areas such as databases, machine learning, and Semantic Web.

Deductive Stream Reasoning. In [EDBT2010, ESWC2010], we specified a general flexible architecture for reasoning over data streams and rich background knowledge, within the LarKC conceptual architecture [ICSC08], leveraging existing DSMS and SPARQL engines. We introduced C-SPARQL (Continuous SPARQL) as an extension to SPARQL to query RDF streams [WWW2009,EDBT2010], and in [ESWC2010] we elaborated on the deductive reasoning support to C-SPARQL, proposing an efficient incremental technique that exploits the transient nature of streams for maintaining the materialization of their ontological entailments.

Inductive Stream Reasoning. The challenges here are, first, the large amount of information, that needs to be processed in a given time window, second, the structured multi-relational nature of the data, third, the sparsity of the typically high-dimensional data, and, fourth, the fact that the data is often incomplete. In [IRMLeS2009] a machine learning approach has been described, that is suitable for this challenging data situation and that has been termed the *Statistical Unit Node Set* (SUNS) learning approach. In [ILP2010], the approach was extended for on-line inductive reasoning.

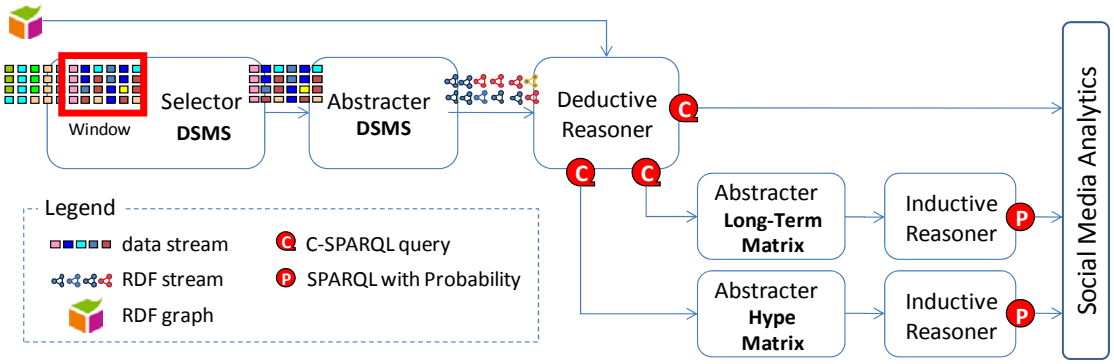

Figure 1. Architecture of a simple Stream Reasoner applied to Social Media Analysis.

[2] http://www.larkc.eu

Digital Object Indentifier 10.1109/MIS.2010.111 0885-9000/$26.00 2010 IEEE

This article has been accepted for publication in IEEE Intelligent Systems but has not yet been fully edited.
Some content may change prior to final publication.

Figure 1 shows the architecture of a simple Stream Reasoner consisting of streamlining within the LarKC platform a set of specialized plug-ins. A *selection* plug-in extracts the relevant data in each input stream by exploiting the *window-processing* ability of a DSMS. The window content is fed into a second plug-in that *abstracts* from fine grain data streams into aggregated events and produces RDF streams as output. A SPARQL engine able to operate under different entailment regimes constitutes the *deductive reasoner* plug-in. C-SPARQL queries are directly registered in the deductive reasoner. The results can be of immediate use or used by two more sub-workflows, both constituted by an abstracter and an inductive reasoner. As explained in Section 4.3, the inferences of the two inductive reasoners can be queried using an extended version of SPARQL which supports probabilities [IRMLeS2009]. Note, that this simple setup can be extended by arbitrarily combining and iterating the deductive and inductive reasoners. E.g., it might be helpful to feed the findings of the inductive reasoner back to the deductive reasoner to deduce further knowledge.

3 Experimental Data

We have based our experiments on Glue[3], a social network that allows users to connect to each other and share Web navigation experiences. In addition, Glue uses semantic recognition techniques to identify books, movies, and other similar topics, and publishes them in the form of data streams. Users can observe the streams and receive recommendations on interesting findings by their friends.

Both the social network data and the real-time streams are accessible via REST APIs. Our experiments build on adapters [SDOW2009] that export Glue data as RDF streams. The entities and relationships considered in the experiments are described in UML in Figure 2.

Users have *names*, and *know* and *follow* other users. Well-known Semantic Web vocabularies [BojarsBPTD08] are used: the Friend of a Friend vocabulary (FOAF) for the user names and the *knows* relationship, and the Semantically-Interlinked Online Communities (SIOC) for the *follows* relationship. *Objects* represent entities of the real world (such as movies or books), with name and category. *Resources* represent sources of information that *describe* the actual objects, such as Web pages about a particular movie or book. As vocabularies, we used rdfs:label for the names and skos:subject to link an object to its category, by means of the *subject* attribute. Moreover, categories identifiers are taken from YAGO.

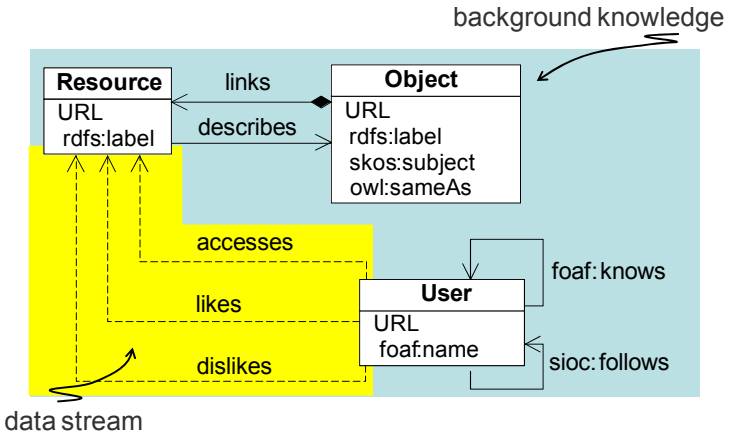

Figure 2. Entities and relationships in our experiments.

[3] http://getglue.com

Digital Object Indentifier 10.1109/MIS.2010.111 0885-9000/$26.00 2010 IEEE

This article has been accepted for publication in IEEE Intelligent Systems but has not yet been fully edited.
Some content may change prior to final publication.

The information described so far is static background knowledge, i.e., in the experiments we assumed that it is stable in a period comparable with the size of a window. Of course updates of this information are also allowed, but do not interfere with window processing. We also have streaming information, namely the notifications of the behavior of users with respect to resources (and, transitively, to objects). The *accesses*, *likes*, and *dislikes* relationships represent the events occurring when users access resources or express opinions about them. In the rest of this paper, we refer to this vocabulary with the prefix *sd*. Quite straightforwardly, all interaction of a generic users U with a resource R generates a triple of the form <U, sd:accesses, R>, and selected interactions generate triples of the form <U, sd:likes, R> and <U, sd:dislikes, R >. An example of possible triples in a stream is:

```
(<:Giulia, sd:accesses, :Avatar>, 2010-02-12T13:18:05)
(<:John, sd:accesses, :Twilight>, 2010-02-12T13:36:23)
(<:Giulia, sd:likes, :Avatar>,    2010-02-12T13:42:07)
```

4 Stream Reasoning at Work

In this section, we progressively apply the simple Stream Reasoner presented in Section 2 to the experimental data described in Section 3. We start with an example of social media analysis performed by a C-SPARQL query under simple RDF entailment. Then, we explain how complex conditions can be expressed using OWL-RL and how our Deductive Stream Reasoner can efficiently answer C-SPARQL queries under OWL2 RL entailment. The last part of this section is dedicated to Inductive Stream Reasoning performed on top of RDF streams generated by C-SPARQL queries registered in the Deductive Stream Reasoner.

4.1 C-SPARQL Under Simple RDF Entailment

C-SPARQL [WWW2009,EDBT2010] is an extension of SPARQL for expressing continuous queries over RDF graphs and RDF streams. Like SPARQL, it can be executed under multiple entailment regimes[4]. C-SPARQL under simple RDF entailment does not require reasoning, but it is already very useful. For instance, it can be used to discover causal relationships between different users' actions in Glue. The following query identifies users who are *opinion makers*, i.e., who are likely to influence the behavior of their followers.

```
1.  REGISTER STREAM OpinionMakers COMPUTED EVERY 5m AS
2.  CONSTRUCT { ?opinionMaker sd:about ?resource }
3.  FROM STREAM <http://streamingsocialdata.org/interactions>  [RANGE 30m STEP 5m]
4.  WHERE {
5.          ?opinionMaker ?opinion ?resource .
6.          ?follower sioc:follows ?opinionMaker.
7.          ?follower ?opinion ?resource.
8.          FILTER ( cs:timestamp(?follower) > cs:timestamp(?opinionMaker)
9.                   && ?opinion != sd:accesses )
10. }
11. HAVING ( COUNT(DISTINCT ?follower) > 3 )
```

Lines 1 and 3 tell the C-SPARQL engine to register the continuous query on the stream of interactions generated by Glue considering a window of 30 minutes that slides every 5 minutes. Line 2 tells the engine to generate an RDF stream as output. The basic triple pattern (BTP) at line 5 matches interactions of potential opinion makers with the resources. Line 6 matches the followers of the opinion makers, and line 7 matches their interactions with the resources. The FILTER clause uses the custom value testing function

[4] http://www.w3.org/TR/sparql11-entailment/

Digital Object Indentifier 10.1109/MIS.2010.111 0885-9000/$26.00 2010 IEEE

This article has been accepted for publication in IEEE Intelligent Systems but has not yet been fully edited.
Some content may change prior to final publication.

`cs:timestamp` that returns the timestamp of the RDF triple producing the binding[5]. It checks whether the interactions of the followers occur on the same resource *after* those of the opinion maker. Note that timestamps are taken from variables that occur only once in patterns applied to streaming triples, thus avoiding ambiguity. Also, the query filters out actions of type “accesses", that are normally required before expressing an opinion such as “like" or “dislike". Finally, the HAVING clause distinguishes *potential* opinion makers from *actual* opinion makers, checking that at least three followers imitated their behavior.

Two approaches, alternative to C-SPARQL, exist: Streaming SPARQL [StreamingSPARQL] and Time-Annotated SPARQL [TA-SPARQL]. Both languages introduce the concept of windows, but only C-SPARQL brings the notion of continuous processing, typical of stream processing, into the language. All other proposals rely on permanent storing of the stream and process it by one-shot queries. Moreover, only C-SPARQL proposes an extension to SPARQL to support aggregates. This extension permits optimizations [EDBT2010] that push, whenever possible, aggregates computation as close as possible to the raw data streams.

4.2 C-SPARQL and Deductive Stream Reasoning

Running C-SPARQL queries under expressive OWL reasoning regimes widens the spectrum of analysis that the Stream Reasoner can perform. For instance, we may define a “movie opinion maker” as an opinion maker who “recently” liked *only* movies. The definition in OWL of users who like only movies is

```
Class( sd:UserOnlyInterestInMovies complete
   intersectionOf(
      sd:User
      restriction(sd:likes allValuesFrom(yago:Movie))
   )
)
```

This ontological definition can be used in the following C-SPARQL query:

```
1. REGISTER STREAM MovieOpinionMakers COMPUTED EVERY 5m AS
2. CONSTRUCT { ?opinionMaker sd:about ?resource }
3. FROM STREAM <http://streamingsocialdata.org/interactions>  [RANGE 30m STEP 5m]
4. WHERE {
5.         ?opinionMaker a sd:UserOnlyInterestInMovies .
6.         ?opinionMaker ?opinion ?resource .
7.         ?follower sioc:follows ?opinionMaker.
8.         ?follower ?opinion ?resource.
9.         FILTER ( cs:timestamp(?follower) > cs:timestamp(?opinionMaker)
10.                 && ?opinion != sd:accesses )
11. }
12. HAVING (COUNT(DISTINCT ?follower) > 3)
```

For instance, if a window contains the triples shown below, then Giulia is an instance of `UserOnlyInterestInMovies`, while John is not (he also liked a book).

```
(<:Giulia, sd:likes, :Avatar>,              2010-02-12T13:18:05)
(<:John, sd:likes, :StarWars>,              2010-02-12T13:36:23)
(<:John, sd:likes, :WutheringHeights>,      2010-02-12T13:38:07)
(<:Giulia, sd:likes, :AliceInWonderland>,   2010-02-12T13:42:07)
```

[5] If the variable gets bound multiple times, the function returns the most recent timestamp value relative to the query evaluation time.

Digital Object Indentifier 10.1109/MIS.2010.111 0885-9000/$26.00 2010 IEEE

This article has been accepted for publication in IEEE Intelligent Systems but has not yet been fully edited.
Some content may change prior to final publication.

The evaluation of the query requires reasoning both on the triples in the window and on the background knowledge about objects described in Glue. In particular, the reasoner must check if users match the ontological definition *before* checking if they are opinion makers. Note that the Deductive Stream Reasoner must know the ontological definition, but also has to combine the RDF stream with relevant background knowledge about movies and books (i.e., it has to know that Wuthering Heights is a book while the other items are movies).

Performing this reasoning task efficiently is not obvious. Possible existing techniques are: incremental maintenance of materialized views in logic [VolzSM05], graph databases [CDE98], extensions of the RETE algorithm for incremental rule-based reasoning [VLDB93], recent attempts to incremental reasoning in description logics [ISWC2007]. All these methods operate incrementally, but none of them is explicitly dedicated to data stream. In [ESWC2010], we proposed a technique for efficiently computing this class of C-SPARQL queries, that incrementally maintains a materialization of ontological entailments exploiting the transient nature of streaming data. By adding expiration time information to each RDF triple, we show that it is possible to compute a new complete and correct materialization whenever the window slides, by dropping explicit statements and entailments that are expired, and then only adding the deductions that depend on the new triples that entered the window.

4.3 Inductive Stream Reasoning using C-SPARQL

Still wondering about Giulia, we may query which movies Giulia will like the most, even if she has not seen them yet. The answer is built for an ad-hoc query; the system uses the last window in the stream in order to determine such predicted probability.

```
1. SELECT ?movie ?prob
2. FROM STREAM <http://streamingsocialdata.org/interactions> [RANGE 30m STEP 5m]
3. WHERE { :Giulia sd:likes ?movie . WITH PROB ?prob
4.         ?movie a yago_Movie .
5.         FILTER ( ?prob > 0 && ?prob < 1 )
6. } ORDER BY ?prob
```

At line 3 we highlighted in bold the construct **`WITH PROB`** which extends SPARQL with the ability to query an inducted model. The variable ?prob assumes the value 1 for the movies she has watched and assumes the estimated probabilities between *zero* and *one* for next movies she would like to watch. The clause **`ORDER BY`** is used to return movies sorted with decreasing the probabilities. The query answer includes pairs of movie title and predicted likelihood as follows:

```
(:WutheringHeightsTvMovie,    0.8347)
(:StarWars,                   0.5693)
```

For running inductive reasoning on semantic data we use the SUNS learning approach mentioned in Section 2. In the approach, we first define the statistical unit, the population, the sampling procedure and the features. A statistical unit is an object of a certain type, e.g., a user. The population is the set of statistical units under consideration. For instance, in the experiments of described in this paper the population is defined as the users of Glue social network. For training models we sample a subset from the population. Then, based on the sample, the SUNS constructs data matrices by transforming the set of RDF-triples related to statistical units into matrices. The rows in the matrix stand for instances of a statistic unit and columns represent their features derived from the associated RDF graph. The binary entries *one* and *zero* represent the truth values “true” and “unknown” of the corresponding triples. Suppose that rows are users

Digital Object Indentifier 10.1109/MIS.2010.111 0885-9000/$26.00 2010 IEEE

This article has been accepted for publication in IEEE Intelligent Systems but has not yet been fully edited.
Some content may change prior to final publication.

and columns are movies. An *one* of the *(i,j)* entry in the matrix indicates that the *i*-th user rates the *j*-th movie as liked; otherwise it is unknown whether or not that user likes that movie.

After the transformation a multivariate analysis of the data matrices is performed. Multivariate prediction methods are especially suited for the challenging data situation: large scale, multi-relational, high-dimensional and highly sparse. The multivariate modeling problem can be solved via singular value decomposition (SVD), non-negative matrix factorization (NNMF) [NNMF], and latent Dirichlet allocation (LDA) [LDA]. All three approaches estimate unknown matrix entries via a low-rank matrix approximation. NNMF is a decomposition under the constraints that all terms in the factoring matrices are non-negative, while LDA is based on a Bayesian treatment of a generative topic model[6]. After matrix completion, the *zero* entries are replaced with certainty values representing the likelihood that the corresponding triples are true. In [ILP2010] we have investigated the performance of these methods in off-line as well as in on-line settings, following different sampling strategies. In this context, on-line setting means that the trained model is applied to predict relationships between entities at query time, including the new entities unseen in the training data set.

In our example user is the main entity of interest (and the reasoner's statistical unit). Each user is involved in a number of relationships such as interests in movies, books and other items, the friendship relationships and the follows relationship. All data is referred to users and is described by RDF-triples, expressing that users "relate" to "objects". C-SPARQL continuously delivers new windows of (aggregated) features to the inductive reasoning and the results of C-SPARQL are transformed into a data matrix, which becomes the input for the inductive reasoner. At predefined time intervals, a learning module applies a multivariate analysis to the data matrices. A second learning module, called hype-model, monitors rapid changes (see Section 5.2). The two data matrices contain, respectively, the long-term information and the short-term trends or "hypes". The "hype matrix" is simply populated with the current window content, whereas the long-term matrix is continuously updated and evolves over time.

5 Evaluation

As evaluation, we first present a stress test to show the scalability of our approach, and then evaluate the application to a real case. We show that each architectural component separately applies orthogonal optimizations, yielding to an efficient solution when one system's output is fed as input to the next system.

5.1 Performance and Scalability Evaluation

As in [SDOW2009], we compared the execution time of a C-SPARQL query in our deductive stream reasoner to the execution time of an equivalent SPARQL query on ARQ[7] with inference support. We show the tests for the query presented in Section 4.2, run on a Pentium Core 2 Quad 2.0GHz with a 2GB RAM.

[6] Recently, we developed a regularized SVD which is insensitive on the rank used in the matrix factorization step.

[7] http://jena.sourceforge.net/ARQ/

Digital Object Indentifier 10.1109/MIS.2010.111 0885-9000/$26.00 2010 IEEE

This article has been accepted for publication in IEEE Intelligent Systems but has not yet been fully edited.
Some content may change prior to final publication.

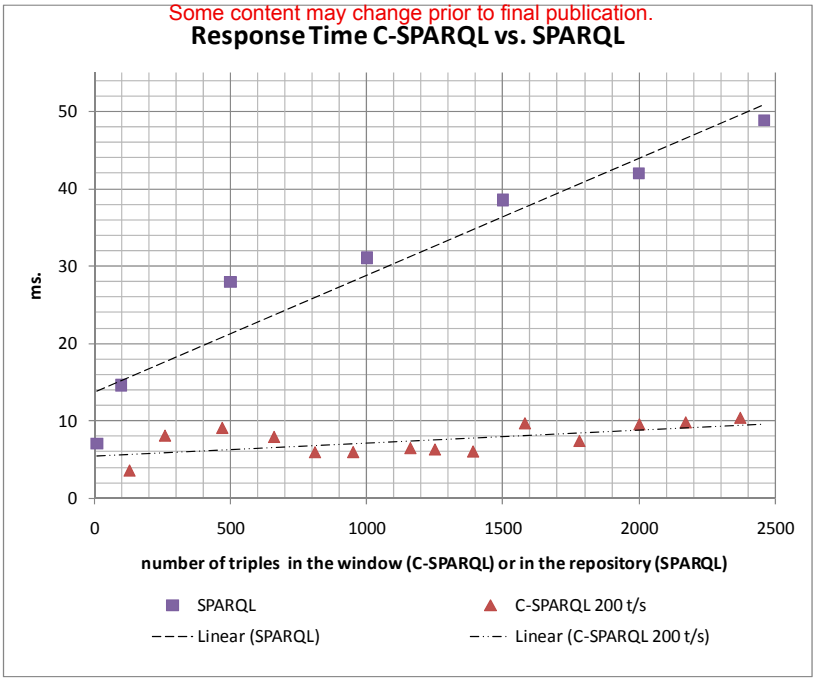

Figure 3. The window-based selection of C-SPARQL outperforms the FILTER-based selection of SPARQL.

A little change to the schema to represent interactions allows writing an equivalent SPARQL query.

```
1.  CONSTRUCT {?opinionMaker sd:about ?resource}
2.  FROM <http://streamingsocialdata.org/interactions>
3.  WHERE {?opinionMaker ?opinion [:about ?resource ;
4.                                 dc:created ?dateOpinionM .]
5.          ?opinionMaker a sd:UserOnlyInterestInMovies .
6.          ?follower sioc:follows ?opinionMaker .
7.          ?follower ?opinion     [:about ?resource ;
8.                                 dc:created ?dateOpinionF .]
9.          FILTER (?opinion != sd:accesses) &&
10.                 ?dateOpinionM > "2010-02-12T13:00:00Z"^^xsd:dateTime &&
11.                 ?dateOpinionM < "2010-02-12T13:30:00Z"^^xsd:dateTime &&
12.                 ?dateOpinionF > "2010-02-12T13:00:00Z"^^xsd:dateTime &&
13.                 ?dateOpinionF < "2010-02-12T13:30:00Z"^^xsd:dateTime &&
14.                 ?dateOpinionF > ?dateOpinionM)
15. }
16. HAVING (COUNT(DISTINCT ?follower) > 3)
```

The code in bold adds (a) two BTP (lines 4 and 8) that match the creation date of the interaction, and (b) four filter conditions (lines 10-13) that select the same time interval of the C-SPARQL query. Notably, the C-SPARQL syntax is more handy and terse.

We registered the C-SPARQL query in our engine, we fed RDF triples in our engine at a rate of 200 triple per second (t/s) and we measured the time required to compute the answer. Using ARQ, we executed the equivalent SPARQL query six times against repositories containing a growing number of triples and we again measured the time required to compute each answer.

The results are shown in Figure 3. By comparing the linear regressions of the two experiments, named Linear(SPARQL), and Linear(C-SPARQL 200 t/s), we see that the C-SPARQL window based selection always performs significantly better than the FILTER based selection of SPARQL in Jena.

Digital Object Indentifier 10.1109/MIS.2010.111 0885-9000/$26.00 2010 IEEE

This article has been accepted for publication in IEEE Intelligent Systems but has not yet been fully edited.
Some content may change prior to final publication.

5.2 Evaluation on a Real Case Scenario

To prove the effectiveness of stream reasoning for social media analytics, we evaluated the accuracy of top-N movie recommendations. First, we compared diverse inductive reasoning approaches with common recommendation methods, some of which were realized by deductive stream reasoning. Secondly, we examined the performance of the combination of both inductive and deductive streaming reasoning.

To gather a data set for the evaluation, we used a predefined C-SPARQL query and then transcoded and stored the output RDF streams into a data matrix. The matrix was continuously updated from February 19 to April 22, 2010. Finally, we selected 245860 interactions made by 2457 users. In particular, we examined the interactions of “Liked movies” relationship. The transformed data matrix (see Section 4.3) is extremely sparse with only 0.002% nonzero elements. To make statistically significant evaluations we removed all users with almost no interactions and items that were evaluated less than 5 times. After pruning, the resulting subset consists of 1455 users and 7724 features with sparsity of 0.02%. The item most specified by users is the "Liked movies"-relation with around 2500 movies. "Liked music" and "Liked recording_artists" was specified about 1300 times and "Liked movie-stars", "Liked tv_shows" and "Liked video_games" approximately 600 times. The remaining 18 features were mentioned less than 250 times.

Being most easily adaptable to the dynamic setting, we applied SVD and regularized SVD for movie recommendations. As baseline methods we utilized, first, a global liked movie list carried out by a simple registered C-SPARQL query, shown below.

```
1. REGISTER STREAM MostLiked COMPUTED EVERY 1d AS
2. SELECT ?movie (SUM(?user) AS ?noOfUser)
3. FROM STREAM <http://streamingsocialdata.org/interactions> [RANGE  XX STEP XX]
4. WHERE {?movie a yago_Movie .
5.         ?user sd:likes ?movie .}
6. GROUP BY ?movie
7. ORDER BY DESC(?nrOfUser)
```

Second, we extracted the most liked movies of a person’s friends, calculated also through a corresponding registered C-SPARQL query (not shown). Third, we applied the k-nearest neighbour (kNN) regression, using the same user-based and movie-based similarity measures defined in [IEEE-IC2003]. By means of cross validation we carefully tuned parameters of each method.

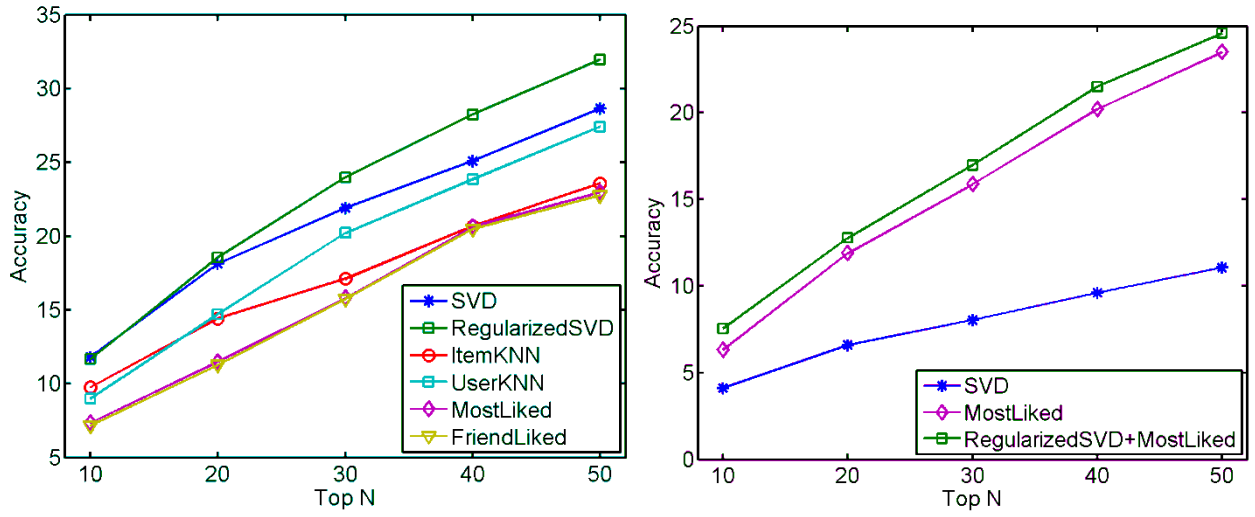

Figure 4. Accuracy of top N movie recommendations. Left: inductive stream reasoning and deductive stream reasoning separately. Right: combination of inductive and deductive stream reasoning

Digital Object Indentifier 10.1109/MIS.2010.111 0885-9000/$26.00 2010 IEEE

This article has been accepted for publication in IEEE Intelligent Systems but has not yet been fully edited.
Some content may change prior to final publication.

Figure 4 left shows the evaluation results, i.e. the percentage of truly liked movies in the top N recommendations where N = 10, 20, 30, 40 and 50. First, SVD and regularized SVD outperformed all baseline methods. In particular, the regularized SVD performed much better than any other method and was robust and insensitive on its parameters as expected. Secondly, both kNN lines are above the baselines. This means that the users and the items collected share some common regularity. For example, users who like the same actors are likely to watch movies in which those actors play. Taking such additional features into account improves significantly the accuracy of movie recommendations. The data regularity is exploited of course by SVD and regularized SVD as well. Thirdly, the two baseline methods almost completely overlapped. The reason might be that the most users would like to watch movies from the same gobal list of the most-popular movies rather than considering the preferences of their friends.

As the second part of the empirical study, we experimented with combining the output of the deductive and the inductive reasoning module. In this scenario the inductive module models the long-term preferences of users, which, in this experiment, is trained only on data that is older than 30 days. It is quite reasonable to assume that the long-term preference model is updated only at larger intervals since the required computations can be quite costly, if one wants to avoid sub-sampling. The deductive reasoning module contributes predictions in the form of “MostLiked”, which simply aggregates recent recommendations to capture the short-term trends of “hype”. Strictly speaking, the latter is also an inductive process, but aggregation is inherently supported by the deductive component as well.

Note that the short-term trend could have been predicted by a multivariate analysis similar to the long-term module and combined in a comparable way to the output of the deductive module. However, Figure 4 right already shows clearly that the combination of the long-term inductive model and the “MostLiked” deductive model outperforms both separated methods. Experiments with more sophisticated “hype-modules” and the exploration of different combination schemes are part of future work.

6 Conclusion and Outlook

We have shown that RDF streams can serve as an interconnecting mechanism to support change management within deductive and inductive reasoners. Suitable type extensions provide both timestamps and probabilities to RDF triples, thereby injecting the required ingredients in the basic RDF model. Extensions to the SPARQL language make RDF streams first-class-citizens, allowing static and streaming RDF data to be used within queries and allowing several window definition options. C-SPARQL embedded within the entailment of a deductive reasoner can be used to generate an output stream, which can then feed an inductive reasoner. Conversely, results of inductive reasoners, described by RDF triples tagged with probabilities, can be inspected by SPARQL engines.

This paper has illustrated a sequential integration between two existing stream reasoning environments within the LarKC platform. However, the pluggable architecture of LarKC allows also for other form of integration with reasoners sharing the same RDF resources, freely reacting to RDF streams, and mutually interacting.

Acknowledgements

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

Digital Object Indentifier 10.1109/MIS.2010.111 0885-9000/$26.00 2010 IEEE

This article has been accepted for publication in IEEE Intelligent Systems but has not yet been fully edited.
Some content may change prior to final publication.

References

[babu01continuous] S. Babu and J. Widom, “Continuous Queries over Data Streams,” *SIGMOD Rec.*, 2001, 30(3):109-120.

[BojarsBPTD08] U. Bojārs, J.G. Breslin, V. Peristeras, G. Tummarello, S. Decker, "Interlinking the Social Web with Semantics," *IEEE Intelligent Systems*, 23(3), 2008, pp. 29-40.

[CDE98] Y. Zhuge, and H. Garcia-Molina, “Graph structured views and their incremental maintenance,” *Proc. of the 14th Int. Conf. on Data Engineering*, 1998, pp. 116-125.

[dsmsbook] M. Garofalakis, , et al., “Data Stream Management: Processing High-Speed Data Streams (Data-Centric Systems and Applications),” *Springer-Verlag New York, Inc*., 2007.

[EDBT2010] D. Barbieri, et al., “An Execution Environment for C-SPARQL Queries”, *Proc. Intl. Conf. on Extending Database Technology (EDBT 2010)*, 2010.

[ESWC2010] D. Barbieri, et al., “Incremental reasoning on streams and rich background knowledge,” *Proc. of the Extended Semantic Web Conf. (ESWC 2010)*, 2010.

[ICSC08] D. Fensel et al., “Towards LarKC: A Platform for Web-Scale Reasoning,” *Proc. IEEE Int’l Conf. Semantic Computing (ICSC 08)*, IEEE Press, 2008.

[IEEE-IC2003] G. Linden, et al., “Amazon.com Recommendations: Item-to-Item Collaborative Filtering,” *IEEE Internet Computing*, vol. 7, 2003, pp. 76–80.

[IEEE-IS-SR] E. Della Valle, et al., “It's a Streaming World! Reasoning upon Rapidly Changing Information,” *IEEE Intelligent Systems*, 24(6), 2009, pp. 83-89.

[ILP2010] Y. Huang, et al., "Multivariate Prediction for Learning on the Semantic Web," *Proc. of the 20th Int. Conf. on Inductive Logic Programming (ILP 2010)*, 2010.

[IRMLeS2009] V. Tresp, et al., "Materializing and querying learned knowledge," *Proc. of the 1st ESWC Workshop on Inductive Reasoning and Machine Learning on the Semantic Web (IRMLeS 2009)*, 2009.

[ISWC2007] B. Cuenca-Grau, et al., “History matters: Incremental ontology reasoning using modules,” *Proc. of the 6th Int. Semantic Web Conf. (ISWC 2007)*, Springer, 2007, pp. 183-196.

[LDA] D.M. Blei, et al., “Latent dirichlet allocation,” *J. Mach. Learn. Res.*, 3, 2003.

[NNMF] D. D. Lee and H. S. Seung, “Learning the parts of objects by non-negative matrix factorization,” *Nature*, 1999.

[SDOW2009] D.F. Barbieri, et al, "Continuous Queries and Real-time Analysis of Social Semantic Data with C-SPARQL," *Proc. of Social Data on the Web Workshop at ISWC 2009*, 2009.

[StreamingSPARQL] A. Bolles, et al., “Streaming SPARQL: Extending SPARQL to Process Data Streams,” *Proc. of European Semantic Web Conf. (ESWC 2008)*, 2008, pp. 448-462.

[SuYYT10] J.H. Su, H.H. Yeh, P.S. Yu, V.S. Tseng, "Music Recommendation Using Content and Context Information Mining," *IEEE Intelligent Systems*, 25(1), 2010, pp. 16-26.

Digital Object Indentifier 10.1109/MIS.2010.111 0885-9000/$26.00 2010 IEEE

This article has been accepted
Some content may change

[TA-SPARQL] A. Rodriguez, et al., "Semantic Management of Streaming Data," *Proc. Intl. Workshop on Semantic Sensor Networks (SSN)*, 2009.

[VLDB93] F. Fabret, et al., "An adaptive algorithm for incremental evaluation of production rules in databases," *Proc. of the 19th Int. Conf. on Very Large Data Bases (VLDB 1993)*, 1993, pp. 455-466.

[VolzSM05] R. Volz, et al. "Incrementally maintaining materializations of ontologies stored in logic databases," *J. Data Semantics*, 2, 2005, pp. 1-34.

[WWW2009] D.F. Barbieri et al., "C-SPARQL: SPARQL for Continuous Querying," *Proc. 18th Int'l World Wide Web Conf. (WWW 09)*, ACM Press, 2009, pp. 1061–1062.

This article has been accepted for publication in IEEE Intelligent Systems but has not yet been fully edited.
Some content may change prior to final publication.

Davide Barbieri Davide Francesco Barbieri is a PhD student at DEI, Politecnico di Milano, since 2008. His research interests address several issues related to stream processing. C-SPARQL will be the main topic of his PhD dissertation.

Dip. di Elettronica e Informazione,
Politecnico di Milano,
P.za L. Da Vinci, 32,
20133 Milano, Italy

Tel.: +39 02 2399 3655
Fax: +39 02 2399 3411
Email: davide.barbieri@polimi.it

Daniele Braga is an Assistant Professor at DEI, Politecnico di Milano, since 2006. He also got his PhD in Computer Science at Politecnico di Milano. His research interests address several issues related to the manipulation of semi-structured data (visual languages and advanced processing for XML data), service integration (with specific reference to complex queries over heterogeneous Web data sources), and Stream Reasoning (C-SPARQL and reasoning over streaming data).

Dip. di Elettronica e Informazione,
Politecnico di Milano,
P.za L. Da Vinci, 32,
20133 Milano, Italy

Tel.: +39 02 2399 3661
Fax: +39 02 2399 3411
Email: daniele.braga@polimi.it

Stefano Ceri is professor of Database Systems at the Dipartimento di Elettronica e Informazione (DEI), Politecnico di Milano. He was visiting professor at the Computer Science Department of Stanford University (1983-1990). His research work covers over three decades (1976-2010) and has been generally concerned with extending database technologies to incorporate new features: distribution, object-orientation, rules, streaming data; with the advent of the Web, research has been targeted towards the engineering of Web-based applications and complex search systems. He is editor in chief of the book series "Data Centric Systems and Applications" (Springer-Verlag), and author of about 250 articles on International Journals and Conference Proceedings and of nine international books. He has been awarded an IDEAS Advanced Grant, funded by the European Research Council (ERC), on Search Computing (2008-2013).

Dip. di Elettronica e Informazione,
Politecnico di Milano,
P.za L. Da Vinci, 32,
20133 Milano, Italy

Tel.: +390223954324
Fax: +390223954524
Email: stefano.ceri@polimi.it

Digital Object Indentifier 10.1109/MIS.2010.111 0885-9000/$26.00 2010 IEEE

This article has been accepted for publication in IEEE Intelligent Systems but has not yet been fully edited.
Some content may change prior to final publication.

Emanuele Della Valle is an Assistant Professor of "Software Project Management " at the Department of Electronics and Information of the Politecnico di Milano since July 2008. He started CEFRIEL 's Semantic Web Activities in 2001 and he coordinated the Semantic Web group until June 2008. His current interest is on scalable processing of information at semantic level. Currently he is focusing on stream reasoning.

Dip. di Elettronica e Informazione,
Politecnico di Milano,
P.za L. Da Vinci, 32,
20133 Milano, Italy

Tel.: +390223954324
Fax: +390223954524
Email: emanuele.dellavalle@polimi.it

Yi Huang is a stuff scientist in Siemens Corporate Technology since 2008 and he is finishing Ph.D. in Ludwig Maximilian University of Munich, Germany, where he received his Diploma degree in Computer Science in 2005. His research interests focus on Statistical Machine Learning, Text Mining, Information Retrieval and Semantic Web.

Siemens AG
Corporate Technology - Corporate Research and Technologies
CT T DE TC3
Otto-Hahn-Ring 6
81739 Munich, Germany

Tel.: +49 (89) 636-47926
Fax: +49 (89) 636-49767
Email: yihuang@siemens.com

Volker Tresp received a Diploma degree from the University of Göttingen, Germany, in 1984, and a Ph.D. from Yale University in 1989. Since 1989 he is the head of a research team in machine learning at Siemens, Corporate Research and Technology. In 1994 he was a visiting scientist at the Massachusetts Institute of Technology's Center for Biological and Computational Learning. Each summer (since 2003) he gives a lecture on machine learning and data mining at the University of Munich. He has been involved in all leading program committees in machine learning and is on the organizing committee of the annual Learning Workshop. He has served on numerous industrial and academic advisory boards.

Siemens AG
Corporate Technology - Corporate Research and Technologies
CT T DE TC3
Otto-Hahn-Ring 6
81739 Munich, Germany

Tel.: +49 (89) 636-49408
Fax: +49 (89) 636-49767
Email: volker.tresp@siemens.com

Digital Object Indentifier 10.1109/MIS.2010.111 0885-9000/$26.00 2010 IEEE

This article has been accepted for publication in IEEE Intelligent Systems but has not yet been fully edited.
Some content may change prior to final publication.

Achim Rettinger is a project manager and assistant professor at the Institute of Applied Informatics and Formal Description Methods (AIFB) at the Karlsruhe Institute of Technology (KIT, Germany). His research interests and publications are in combining machine learning, knowledge discovery and human computer systems with semantic technologies. Achim Rettinger did his PhD studies at the Technische Universtät München (TUM, Germany) and at the Siemens AG in Munich, Germany. He studied and worked on research projects at the University of Koblenz-Landau (Germany), Osaka University (Japan), University of Bath (UK), University of Alberta (Canada) and University of Georgia (USA).

Institute AIFB - Building 11.40
KIT-Campus Süd
D-76128 Karlsruhe, Germany

Tel.: +49 721 608 7363
Fax: +49 721 608 6580
Email: rettinger@kit.edu

Hendrik Wermser did his Bachelor's thesis at Technische Universität München (TUM) and at Siemens AG, Munich, Germany and is currently pursuing his Master's degree at TUM. His research interests are in information retrieval, recommender systems and machine learning.

Technical University of Munich
Theresienstr. 106
D-80333 München, Germany

Phone: +49 171 4255513
Fax: +49 5331 900989-199
Email: wermser@cs.tum.edu

Digital Object Indentifier 10.1109/MIS.2010.111 0885-9000/$26.00 2010 IEEE