Semantic Complex Event Processing for Social Media Monitoring - A Survey

Robin Keskisärkkä and Eva Blomqvist

Linköping University, 581 83 Linköping, Sweden,
{robin.keskisarkka|eva.blomqvist}@liu.se

Abstract. Semantic Complex Event Processing (CEP) is a promising approach for analysing streams of social media data in crisis situations. Traditional CEP approaches lack the capability to semantically interpret and analyse data, which *Semantic* CEP attempts to address, but current approaches have a number of limitations. In this paper we survey four semantic stream processing engines, and discuss them with the specific requirements of CEP for social media monitoring in mind. Current approaches assume well-structured data, known streams and vocabularies, and mainly static event patterns and ontologies, neither of which are realistic assumptions in our scenario. Additionally, the languages commonly used for event pattern detection, i.e., SPARQL extensions, lack several important features that would facilitate more advanced statistical and textual analyses, as well as adequate support for temporal and spatial reasoning. Being able to utilize external tools for processing specific tasks would also be of great value in processing social data streams.

Keywords: Complex Event Processing, Stream Reasoning, Social Media Monitoring

1 Introduction

To achieve situation awareness in crisis situations, traditionally, both authorities (including emergency services) and citizens rely mainly on official communication channels. Today, citizens as well as authorities are increasingly using new, and more direct, communication channels, such as social news media, e.g., Facebook, Twitter etc. [11, 23]. Analysing such media have proved useful for detecting earthquakes [20], assessing the spread of influenza [12], and assisting disaster relief [21], just to name a few examples. However, so far, "monitoring" has either been done more or less manually, e.g., by using native search interfaces of social media sites, or by using highly specialized analysis models for mining data about specific types of situations, e.g., earthquakes [20] or influenza outbreaks [12].

Although there are a few examples of recent systems that utilize Semantic Web (SW) technologies, e.g., for named entity recognition and classification [2], or geo-location and information extraction [13], these are usually highly dependent on user intervention and hand-crafted rules (or queries) for the event type at hand. SW technologies can support the *analysis* of data, and have the potential to boost the effectiveness of current social media monitoring methods.

Complex Event Processing (CEP) [17], e.g., for recognizing and analysing the current situation, is a task that the human mind performs with high precision, due to our inherent pattern-recognition capabilities. However, when the amount of data increases, e.g., on a Web scale, it is no longer feasible for a person to perform "situation recognition", without some automated pre-processing for finding the interesting patterns in data. The term Complex Event Processing originates in the abstraction task, where sets of "low-level" events are aggregated (possibly through several steps), by means of generalized event pattern descriptions, into abstracted "high-level" events (i.e., complex events) that make sense to a user. In this paper we use the term "event" in the sense of CEP events, i.e., elements of a data stream, rather than denoting situations in the outside world.

Classical CEP technologies are not targeted at *semantically* interpreting and analysing data, but commonly work on streams of highly structured data with well-known interpretations. Recently, this challenge has been picked up by the SW community [25, 4], in order to enhance CEP systems with SW technologies. In this paper we focus on CEP approaches utilizing SW technologies for analysing data, which we call *Semantic CEP*, and we review the current state of the art in this field, with specific focus on social media monitoring requirements.

The paper starts with an analysis of social media monitoring requirements in Section 2. In Section 3 we give an overview of the functionalities that the surveyed stream processing engines provide, including their SPARQL extensions. Section 4 then provides a discussion of the engines and their capabilities, and analyses their performance against our requirements from Section 2. Finally, in Section 5 we draw some conclusions and outline future challenges.

2 Social Media Monitoring Requirements

While CEP (and Semantic CEP) has so far mainly been employed on highly structured data streams, such as sensor data, in social media monitoring we are using humans as sensors, and the streams are beyond our control since they are produced and made available on the Web. This presents several new challenges, and results in a number of requirements that we summarize in Table 1.

Table 1. Semantic CEP requirements for social media monitoring scenarios.

Req#	Description
1	Consumption of online streams with evolving structures (vocabularies) and content that are not under our own control.
2	(Pre-)Processing of streams using external tools.
3	Filtering of data streams to focus CEP only on the relevant data.
4	Using information abundance to compensate for missing, incomplete, unreliable, or ambiguous information.
5	Handling streams of RDF-graphs, rather than single triples.
6	Temporal reasoning, including time intervals and temporal relations between events.
7	Spatial reasoning, including spatial relations between events and objects.
8	Reasoning over streams combined with background knowledge.
9	On-the-fly query generation, and (semi-)automatic evolution of event patterns and background knowledge.

The fact that streams are not under our own control pose challenges regarding structural changes, e.g., if vocabularies change, and requires interpretation of stream content at runtime (c.f. Req. 1). For pre-processing unstructured information we may need to use external tools (c.f. Req. 2) to process text, identify spatiotemporal anomalies [24], and deal with automated, relayed, and repeated messages [22]. The amount of data also poses a challenge, often requiring irrelevant messages to be filtered out (c.f. Req. 3), and structured information to be extracted before applying CEP methods.

When analysing sensor data, e.g., from physical sensors, we may be satisfied with reports from a single sensor, if the sensor provides accurate and unambiguous information. When dealing with social streams, on the other hand, data can be incomplete, unreliable, incorrect, and is often expressed in natural language. However, instead of one sensor we have millions, and the abundance of data can facilitate dealing with incompleteness, and unreliable information (c.f. Req 4).

After pre-processing, the social streams need to be broken down into RDF-data streams to allow for matching data to event patterns. As social media streams typically consist of messages coupled with metadata, such as message author, creation time, and geo-spatial information, each message results in multiple triples. This means that each message will be passed to the Semantic CEP engine not as a single RDF-triple (which is often the case for sensor data) but as an RDF-graph (c.f. Req 5). In order to detect sequences of events, compare states evolving with time, or determine the order in which events occurred, the ability to reason about temporal relations between events is essential (c.f. Req 6). In situation monitoring geo-spatial information is also of great importance, since it enables locating events in the real world (c.f. Req 7).

In order to successfully monitor a situation it is typically necessary to integrate streaming data, static data, and background knowledge, e.g., ontologies, and to be able to reason on these sources in combination (c.f. Req 8). The changing nature of social streams additionally requires event patterns and queries that can evolve or be replaced as needed. To support this type of dynamic queries it must be possible to generate queries on the fly, based on rich descriptions of event patterns, e.g., from an ontology, rather than using fixed hand-crafted queries (c.f. Req 9).

3 Semantic CEP - State of the Art

SW technologies have until recent years operated mainly on *static* data and ontologies, despite the fact that the Web is a rapidly changing source of data [25]. Traditional SW techniques, e.g., query languages and automated reasoning, commonly assume that a dataset is reset when updated with new data, and that reasoning is performed from scratch before queries are executed again. However, for streaming data, reasoning and result delivery are performed continuously rather than in a "batch mode". Also, traditional methods have not focused on reasoning over time and space, which is necessary to capture some of the important characteristics of streaming data and events.

Commonly streams are handled by observing them through *windows*. Triples inside the window are assumed to be relevant, while triples outside the window are ignored. The window size is either defined in terms of the amount of data (logical windows), or as physical windows defined in terms of time [5]. The predominant approach to Semantic CEP handle streams where individual RDF-triples constitute "low-level" events, which are then matched to SPARQL queries, i.e., "event patterns". The output is a stream of query results (RDF-triples), that may represent complex events.

For the purpose of this survey, we have selected four recent approaches to focus on. The approaches described in the paper were chosen because they represent stream processing engines that are publicly accessible, make extensive use of SW technologies, and can be regarded as representative of the state of the art in the field. Other approaches exist, e.g., Streaming SPARQL [7], and classical CEP tools, such as ESPER [1], but these will not be discussed in this paper.

C-SPARQL Continuous SPARQL (C-SPARQL) [6] was developed within the LarKC-project. C-SPARQL enables standing queries on streaming RDF-data and static background knowledge, using a SPARQL syntax extension. Both physical windows, defined in terms of time, and logical windows, defined in terms of number of triples, can be applied to individual streams. C-SPARQL reports duplicate triples when the windows of two executions of a query overlap; this has the benefit of always providing the full result of the query upon execution, rather than requiring the user to keep track of the state, but can also cause delays.

ETALIS The ETALIS engine is an open source system for complex event processing that uses two languages to describe event patterns: ETALIS Language for Events (ELE), and Event Processing SPARQL (EP-SPARQL) [8, 3]. ETALIS is based on declarative semantics, grounded in logic programming, and implemented in Prolog. The EP-SPARQL language supports both static background knowledge and streaming RDF-data. It extends the standard SPARQL syntax with a number of binary operators; namely, `SEQ`, `EQUALS`, `OPTIONALSEQ`, and `EQUALSOPTIONAL`. These operators work much like the standard operators `UNION` and `OPTIONAL` in SPARQL. The operators can be seen as left, right, or full joins based on temporal aspects. Also, functions can be used to access duration, start time, and end time for results. EP-SPARQL also supports `CONSTRUCT` queries, allowing for streams of data to be used in recursive production rules.

CQELS Continuous Query Evaluation over Linked Streams (CQELS) is an engine that focuses on continuous queries (expressed in a SPARQL extension), which executes every time new data is introduced that matches the query in question [16, 14]. Matching triples are returned immediately when they arrive, i.e., queries fire in a data driven fashion rather than based on predefined time intervals. This has an important implication; a query will not return duplicates

[1] http://esper.codehaus.org/

of results. The benefit of this type of implementation is that the processing can be spread out over time, and event patterns can be fired in near real-time, rather than being executed as a batch of queries at predefined time intervals.

CQELS was created to address the problem of scalable query processing of linked stream data, tightly integrated with traditional linked data [16]. It makes use of various optimizations techniques, such as caching of intermediate query results, and continuous reordering of operators according to heuristic rules. In the tests carried out in [14] the CQELS engine outperforms C-SPARQL and ETALIS in terms of speed, at least for certain query types.

INSTANS In contrast to the other three solutions, INSTANS [19, 1] is a query engine capable of continuous and incremental execution of standard-compliant SPARQL. Like CQELS and ETALIS it avoids repeated computation of the same data, and makes results available immediately when event patterns are matched. The engine supports SPARQL Update and can therefore use both `INSERT`, to add triples into existing graphs, and `DELETE`, to remove triples. INSTANS processes incoming triples based on the Rete-algorithm, which means that each triple is processed when it becomes available and intermediate matches are cached into the Rete-net. Identical parts of queries are merged in the Rete structure. INSTANS supports the detection of missing events by employing a timer that can be registered for events that are to be monitored. At present the engine does not directly support complex reasoning on data in the past. As suggested in [9] the modelling of time could be expressed completely in RDF, although the effects in terms of query complexity and performance in the case of streaming RDF-data has not been evaluated.

4 Discussion

In this section we analyse general properties of the engines, and proceed to discuss to what extent the engines support requirements of social media monitoring.

4.1 Comparison of the Approaches

Expressivity There are two primary ways of dealing with information that changes over time; either saving a snapshot of the graph every time it changes, or adding timestamps to the triples represented in the graph [9]. Storing snapshots of graphs every time they change is often not a scalable solution when dealing with streaming data. A variation of the snapshot approach is instead to keep only the most recent graph in memory, i.e., to update the current state with the new information only, but this approach greatly limits the possibilities for temporal reasoning. To instead annotate triples with timestamps increases the expressive power considerably.

All of the engines surveyed use the timestamp approach for handling streaming data, but for CQELS, C-SPARQL, and EP-SPARQL the timestamps are handled by the systems internally and not made available as RDF-triples. CQELS does not make the timestamp information available to the user at all from within queries. C-SPARQL allows timestamps to be accessed for individual triples, e.g., making it possible to determine the order of events, but windows in the past are not well supported. EP-SPARQL supports reasoning with time through functions that can be used inside queries to access timestamps and durations, and provides operators to define temporal ordering. EP-SPARQL additionally allows windows to be defined in the past.

INSTANS differs from the other approaches in that it handles RDF-streams using standard compliant SPARQL, and like EP-SPARQL, it does not rely on windows over individual streams for performing stream processing and event detection [19, 1]. It is not discussed explicitly how INSTANS would deal with events that, e.g., must be defined as a sequence of events, but since INSTANS supports both `DELETE` and `INSERT` it would be possible to use a model of time in RDF as suggested in [9].

Standard SPARQL queries miss the timeout aspect of events, that is, the detection of anticipated but absent events, since the language is designed for querying finite datasets. INSTANS supports the timeout aspect by employing a timer functionality, making it possible to generate events after a predetermined time interval. C-SPARQL has some support for the detection of missing events, since queries can be evaluated periodically.

Support for `OPTIONAL` constructs is important since all variables returned by a query are not always bound to values. Support for the features introduced in SPARQL 1.1 is limited in CQELS, C-SPARQL, and EP-SPARQL, which makes the function `BOUND` important in terms of expressive power, as it allows queries to express the absence of patterns using SPARQL 1.0. All the surveyed engines support both operators.

CQELS, C-SPARQL, and EP-SPARQL support SPARQL `CONSTRUCT` to produce new RDF-streams, which could be fed back into the engines to support recursive reasoning. However, neither C-SPARQL nor CQELS have built-in support for interconnected queries and layering, thus it would require some workaround. Also, there are no evaluations showing how well this approach would perform. Similar reasoning can be applied in INSTANS by using `INSERT` instead of `CONSTRUCT`. Table 2 summarizes some of the features of the systems.

Performance Measuring performance for the different engines is not completely straightforward. CQELS, ETALIS, and INSTANS use an eager execution strategy, where queries are executed as soon as new triples arrive. C-SPARQL queries could define windows that slide for every triple but it is not optimized to do so because it reports duplicates of results when windows overlap, and the execution rate of queries would still be handled internally.

In [15, 14] the performance of CQELS, ETALIS, and C-SPARQL was evaluated in terms of, e.g., execution throughput, and scalability in terms of number of queries and static data size. Their findings show that the periodical execution of C-SPARQL results in a considerably lower throughput compared to the other two engines. Only CQELS scaled well with growing static data, both because it is

Table 2. Summary of the expressiveness of the four engines.

	CQELS	**C-SPARQL**	**EP-SPARQL**	**INSTANS**
Static RDF-data	yes	yes	yes	yes
Streams of RDF-triples	yes	yes	yes	yes
Streams of RDF-graphs	no[1]	no	no[1]	no[3]
Windows over streams	yes	yes	yes[2]	yes[2,3]
Multiple windows over the same stream	yes	no[1]	no	yes[3]
Timestamps available in queries	no	for triples	for results	as triples
Windows in the past	no	limited	yes	yes[3]
Support for RDFS/OWL reasoning	no	no	no	no
Support for detection of missed events	no	limited	no	yes

[1] No explicit support, although workarounds may be possible.
[2] Not defined for individual streams.
[3] Must be defined "manually" within queries.

the only native implementation and because it pre-computes and indexes intermediate results. None of the engines employ optimization techniques for multiple queries, which could avoid redundancy of computations, and they therefore did not scale well with multiple parallel queries in this evaluation.

The performance of INSTANS was compared to C-SPARQL in [18]. Results show that while the average processing delay for INSTANS was around 12 ms, C-SPARQL needed between 12–253 ms for the same queries (windows sizes of 5–60 events). Higher window repetition rates favoured INSTANS even more.

Reasoning None of the engines reviewed here integrates a standard reasoning engine, i.e., for RDFS/OWL reasoning. Automated reasoning typically involves materializing graphs based on RDFS/OWL semantics. For streaming data this is usually not feasible, i.e., to perform the materialization from scratch every time. There are, however, alternative strategies that maintain a materialized graph by updating only those parts that are affected by a change. For example, the DRed algorithm tracks changes in materializations by storing expiration times for inserted triples [6]. In [6] the authors also suggest a different approach based on monitoring the triples inserted and keeping track of the entailments resulting from each inserted triples. A maintenance program would tag each inserted and entailed triple with an expiration time, and in addition every entailed triple is valid only if there still exists some justification for that particular triple. This scales well compared to the naive approach, i.e., to recompute the complete materialization, but as the percentage of background information subject to change increases the time needed to maintain the materialization increases and eventually favours the naive approach.

Handling unscheduled changes also needs to be dealt with when maintaining a materialization. If an event is triggered it will be valid for a predefined amount of time, but it may also be invalidated at any given time. A possible strategy is to remove entailed triples if no valid justification for the entailment remains, similar to the strategy used in the DRed algorithm [6].

Another way of handling reasoning is to use rewriting, to incorporate reasoning into queries instead of using RDFS/OWL reasoners. In CQELS, C-SPARQL, and EP-SPARQL recursive relationships could be described using `CONSTRUCT` to create new streams. ETALIS explicitly supports this type of recursive production rules. INSTANS could accomplish this in a similar way using `INSERT`. Another possible way of supporting reasoning has been implemented in a system called Sparkwave [10]. Similar to INSTANS, it is based on the Rete-algorithm, but supports partial RDFS/OWL reasoning by using a pre-processing epsilon network.

4.2 General Limitations and Challenges

Before proceeding to analyse specific requirements, we discuss some general challenges within Semantic CEP. First, we note that there are no standards, nor de-facto ones, for how data streams should be represented, accessed online, and referred to, nor are there any standards for the representation of timestamps. In the surveyed approaches streaming RDF-triples are usually represented as tuples (in all engines but INSTANS), consisting of an RDF-triple and a timestamp, but there is no agreement on how to handle such RDF-streams within queries.

All approaches, except INSTANS, extend the SPARQL query syntax, but as there are no predefined standards they differ somewhat. Each syntax has different expressive power, e.g., EP-SPARQL cannot refer to individual streams (true also for INSTANS that uses standard SPARQL), while CQELS cannot access timestamps for triples or windows, and neither C-SPARQL nor CQELS can adequately handle windows in the past. There are, however, many similarities between the extended query languages, e.g., they all include support for aggregate functions within query windows, which can significantly simplify some common query expressions. In INSTANS aggregates must be computed from scratch, using only the primitive mathematical functions allowed in SPARQL, which makes such operations very complex and require several aggregation steps. In our opinion, the specific aggregate functions that are particularly useful in the context of analysing streaming data should be proposed for inclusion in future standards. A standardized query language would also make it easier to compare the performance and limitations of different engines.

Being able to compare current and historical states is fundamental in many CEP applications. But querying past events has so far not received proper attention. EP-SPARQL supports states defined in the past primarily by allowing events to be ordered chronologically. In CQELS and C-SPARQL all windows are defined up to the current time, making it difficult to define windows in the past at all. In INSTANS information from past events is cached into the Rete-net, provided that there are queries utilizing it, and all removals of triples must be done explicitly using `DELETE` operations, making past events available by default. The authors of INSTANS have also used timestamps as triggers to inform the system that a certain triple is no longer valid [18]. Time represented in RDF could be used to reason over windows in the past without dedicated functions for handling time, but this requires that time is managed explicitly in queries. In [9] the authors present two ways in which time can be represented in RDF format using either point-based labelling, or interval-based labelling. Both solutions provide support for more complex reasoning than the solutions used in either of the engines surveyed in this paper, but scalability and performance of such a solution remains to be tested.

None of the engines surveyed support any RDFS or OWL reasoning, i.e., using a standard reasoning engine. This makes it difficult to formulate questions that rely on such things as subclass or equivalence axioms, and it considerably restricts the way in which background knowledge (in the form of ontologies and static data) can be exploited.

4.3 Analysis of Social Media Monitoring Requirements

The requirements for social media stream monitoring, discussed in Section 2, are here reviewed with regard to the approaches discussed in this survey. Overall results are summarized in Table 3.

All the surveyed approaches use local streams in their implementations, i.e., streams are not being retrieved and consumed online, and vocabularies are known beforehand (c.f. Req. 1). These approaches require new hand-crafted queries to be registered if vocabularies change. The (pre-)processing of streams using external tools (c.f. Req. 2) can be achieved if streams can be created and added to the engine on the fly from outside the engine environment, and if the system can handle several parallel streams. The engines discussed in this paper use two different approaches for handling streams; data from streams are added to a repository but the streams are never referenced directly, or streams are stored in a repository but must be referenced explicitly in queries. The first approach enables simple collaboration between queries, but keeping the contents of different streams apart can be difficult. The second approach enables referencing specific streams directly, but makes communication between queries complicated.

To the best of our knowledge, all current Semantic CEP approaches are based on SPARQL, which makes it easy to use triple patterns for filtering data (c.f. Req. 3). However, the engines lack support for combining SW technologies with external tools, making it difficult to analyse semi-structured data streams, e.g., textual information in Tweets, as well as statistically aggregating and combining data from a large numbers of streamed objects (c.f. Req. 4). Aggregate functions, such as MIN, MAX, and AVG, are supported in CQELS, C-SPARQL, and EP-SPARQL, and can to some extent also be calculated in INSTANS through various workarounds. However, new aggregate functions and operators should be considered, for handling important relationships when managing social data streams, e.g., for analysing geo-spatial relationships (c.f. Req. 7), or for applying statistical filters. Also, temporal reasoning is a common task when working with streaming data (c.f Req. 6). INSTANS makes timestamps available as triple patterns, which allows for complex temporal reasoning [9], but introducing operators to support the most commonly used tasks, as has been done in EP-SPARQL, can greatly simplify the process of writing queries involving temporal aspects.

All the surveyed systems process events represented as RDF-triples, but ensuring that the boundaries of events represented as RDF-graphs are respected is not straightforward (c.f. Req 5). One solution is to use a common triple subject for each event, but more elaborate structures are sometimes required, e.g., when

Table 3. Summary of current support and open challenges.

Requirement	**Current support**
1. Online streams and vocabularies	Systems assume local streams, mostly with a fixed vocabulary
2. External stream (pre-)processing	Diverse solutions (partial support), several steps of stream processing is a prerequisite
3. Filtering	Simple filters through SPARQL queries supported, statistical filtering and NLP filters largely unsupported
4. Using information abundance	Not natively supported, can be achieved to some extent through tailored queries
5. Streams of RDF-graphs	Not explicitly supported
6. Temporal reasoning	Partially supported (through diverse approaches)
7. Spatial reasoning	No native support, can partially be solved through queries
8. Reasoning with background knowledge	Background knowledge supported, but standard RDFS/OWL reasoning is not supported
9. Query generation	Event patterns represented as hand-crafted queries, query generation from declarative patterns not supported

a query results in a "false match" if triggered based on only a subset of the RDF-triples of that event. We may also receive data about an event from multiple sources, e.g., a Twitter stream may be delivered first in its original form, but additional triples could arrive later on, after messages have been analysed by an external service. The periodical execution strategy employed by C-SPARQL seems to make it more or less impossible to fully respect event boundaries. In addition, there are currently no standardized Web protocols for streaming RDF-data, making the task of appropriately handling RDF-streams more difficult.

None of the surveyed engines support RDFS/OWL reasoning (c.f. Req 8), a powerful tool that would allow Semantic CEP systems to take full advantage of ontologies as background knowledge, and to declaratively express event patterns. Reasoning capabilities could also increase the support for both temporal and spatial reasoning. However, RDFS/OWL reasoning comes with its own challenges of scalability and efficiency. Replacing queries at runtime allows for queries to be updated, however, today queries are mainly assumed to be hand-crafted. To define queries that evolve, based on changes in background knowledge or declarative descriptions of event patterns (c.f. Req 9), is currently not supported. In INSTANS there is no real difference between "facts" (static data) and events in the event stream. It would therefore be possible to update a static vocabulary based on queries over the event stream.

5 Conclusions and Future Research Challenges

In this paper we have discussed Semantic CEP as a promising technology for analysing streams of social media data in crisis situations. Traditional CEP approaches, although usually very efficient, lack the capability to semantically interpret and analyse data. Semantic CEP adds such capabilities, but current approaches focus more on simple stream processing rather than identifying complex events, and therefore have a number of limitations; for example, they assume well-structured data, known vocabularies, and static queries and ontologies. Additionally, the languages commonly used for event pattern detection, i.e., SPARQL extensions, lack several important features that would facilitate more advanced statistical and textual analyses, as well as better support for temporal and spatial reasoning. RDFS/OWL reasoning is not supported by any of the current approaches, although inference rules can usually be expressed as queries if static background knowledge can be taken into account.

Future challenges for Semantic CEP, in general, include standardization efforts, concerning both how RDF-streams should be represented, advertised and accessed over the Web, as well as how event patterns should be expressed and executed, e.g., in a standardized query language, such as SPARQL or an extension thereof. More specifically for a social media monitoring scenario, future Semantic CEP engines must allow for accessing online streams, with vocabularies that evolve along with the outside world. Such systems also need to view the information abundance as an asset, which can be statistically analysed and from which new structured information can be retrieved.

References

1. Abdullah, H., Rinne, M., Törmä, S., Nuutila, E.: Efficient matching of SPARQL subscriptions using rete. In: Proceedings of the 27th Annual ACM Symposium on Applied Computing. pp. 372–377. ACM, New York, USA (2012)
2. Abel, F., Hauff, C., Houben, G.J., Stronkman, R., Tao, K.: Twitcident: fighting fire with information from social web streams. In: Proc. of the 21st international conf. companion on World Wide Web. pp. 305–308. ACM, New York, USA (2012)
3. Anicic, D., Fodor, P., Rudolph, S., Stojanovic, N.: EP-SPARQL: a unified language for event processing and stream reasoning. In: Proceedings of the 20th international conference on World Wide Web. pp. 635–644. ACM, New York, NY, USA (2011)
4. Barbieri, D., Braga, D., Ceri, S., Della Valle, E., Grossniklaus, M.: Stream Reasoning: Where We Got So Far. In: Proc. of the 4th Intl. Workshop on New Forms of Reasoning for the Semantic Web: Scalable and Dynamic (NeFoRS) (2010)
5. Barbieri, D., Della Valle, E.: A Proposal for Publishing Data Streams as Linked Data - A Position Paper. In: Proc. of the LDOW2010 workshop, co-located with WWW2010. CEUR Workshop Proceedings, vol. 628 (2010)
6. Barbieri, D.F., Braga, D., Ceri, S., Della Valle, E., Grossniklaus, M.: Incremental reasoning on streams and rich background knowledge. In: Proceedings of the7th Extended Semantic Web Conference, ESWC 2010, Heraklion, Crete, Greece, May 30 –June 3, 2010, Proceedings, Part I. pp. 1–15. LNCS, Springer (2010)
7. Bolles, A., Grawunder, M., Jacobi, J.: Streaming SPARQL - Extending SPARQL to Process Data Streams. In: 5th European Semantic Web Conference, ESWC 2008, Tenerife, Canary Islands, Spain, June 1-5, 2008 Proceedings. LNCS, vol. 5021, pp. 448–462. Springer (2008)
8. Fodor, P., Darko, A., Rudolph, S., Ding, J., Hafsi, A., Sthümer, R.: The etalis system version 1.1 manual (draft) (August 2010), http://ewl.cewit.stonybrook.edu/pfodor/misc/etalis_manual/
9. Gutierrez, C., Hurtado, C.A., Vaisman, A.: Introducing Time into RDF. IEEE Trans. on Knowl. and Data Eng. 19(2), 207–218 (2007)
10. Komazec, S., Cerri, D., Fensel, D.: Sparkwave: continuous schema-enhanced pattern matching over rdf data streams. In: Proc. of the 6th ACM Intl. Conference on Distributed Event-Based Systems. pp. 58–68. ACM, New York, USA (2012)

11. Kwak, H., Lee, C., Park, H., Moon, S.: What is Twitter, a social network or a news media? In: Proceedings of the 19th international conference on World wide web. pp. 591–600. ACM, New York, NY, USA (2010)
12. Lampos, V., De Bie, T., Cristianini, N.: Flu detector: tracking epidemics on twitter. In: Proc. of the 2010 European conference on Machine learning and knowledge discovery in databases: Part III. pp. 599–602. Springer (2010)
13. Lanfranchi, V.: Realtime situation awareness via social media streams (2012), demo session at ISCRAM2012, Vancouver, Canada
14. Le-Phuoc, D., Dao-Tran, M., Parreira, J.X., Hauswirth, M.: A native and adaptive approach for unified processing of linked streams and linked data. In: Proc. of the 10th International Semantic Web Conference, Bonn, Germany, October 23-27, 2011, Part I. pp. 370–388. Springer (2011)
15. Le-Phuoc, D., Dao-Tran, M., Pham, M.D., Boncz, P.A., Eiter, T., Fink, M.: Linked stream data processing engines: Facts and figures. In: The Semantic Web - ISWC 2012 - 11th International Semantic Web Conference, Boston, MA, USA, November 11-15, 2012, Proceedings, Part II. LNCS, vol. 7650, pp. 300–312. Springer (2012)
16. Le-Phuoc, D., Parreira, J.X., Hausenblas, M., Hauswirth, M.: Unifying stream data and linked open data. Tech. rep., Digital Enterprise Research Institute (DERI) (August 2010)
17. Luckham, D.C.: The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (2001)
18. Rinne, M., Abdullah, H., Törmä, S., Nuutila, E.: Processing Heterogeneous RDF Events with Standing SPARQL Update Rules. In: On the Move to Meaningful Internet Systems: OTM 2012. LNCS, vol. 7566, pp. 797–806. Springer (2012)
19. Rinne, M., Nuutila, E., Törmä, S.: INSTANS: High-Performance Event Processing with Standard RDF and SPARQL. In: Proceedings of the ISWC 2012 Posters and Demonstrations Track. Boston, US (2012)
20. Sakaki, T., Okazaki, M., Matsuo, Y.: Earthquake shakes Twitter users: real-time event detection by social sensors. In: Proceedings of the 19th international conference on World Wide Web. pp. 851–860. ACM, New York, USA (2010)
21. Slagh, C.L.: Managing Chaos, 140 Characters at a Time: How the Usage of Social Media in the 2010 Haiti Crisis Enhanced Disaster Relief, vol. 9781243460929. Proquest, Umi Dissertation Publishing (2011)
22. Stringhini, G., Kruegel, C., Vigna, G.: Detecting spammers on social networks. In: Proceedings of the 26th Annual Computer Security Applications Conference. pp. 1–9. ACSAC '10, ACM, New York, NY, USA (2010)
23. Teevan, J., Ramage, D., Morris, M.R.: #twittersearch: a comparison of microblog search and web search. In: Proceedings of the fourth ACM international conference on Web search and data mining. pp. 35–44. ACM, New York, NY, USA (2011)
24. Thom, D., Bosch, H., Koch, S., Wörner, M., Ertl, T.: Spatiotemporal anomaly detection through visual analysis of geolocated twitter messages. In: 2012 IEEE Pacific Visualization Symposium, PacificVis 2012, Songdo, Korea (South), February 28 - March 2, 2012. pp. 41–48. IEEE (2012)
25. Valle, E.D., Ceri, S., Harmelen, F.v., Fensel, D.: It's a Streaming World! Reasoning upon Rapidly Changing Information. IEEE Intelligent Systems 24(6), 83–89 (Nov 2009)